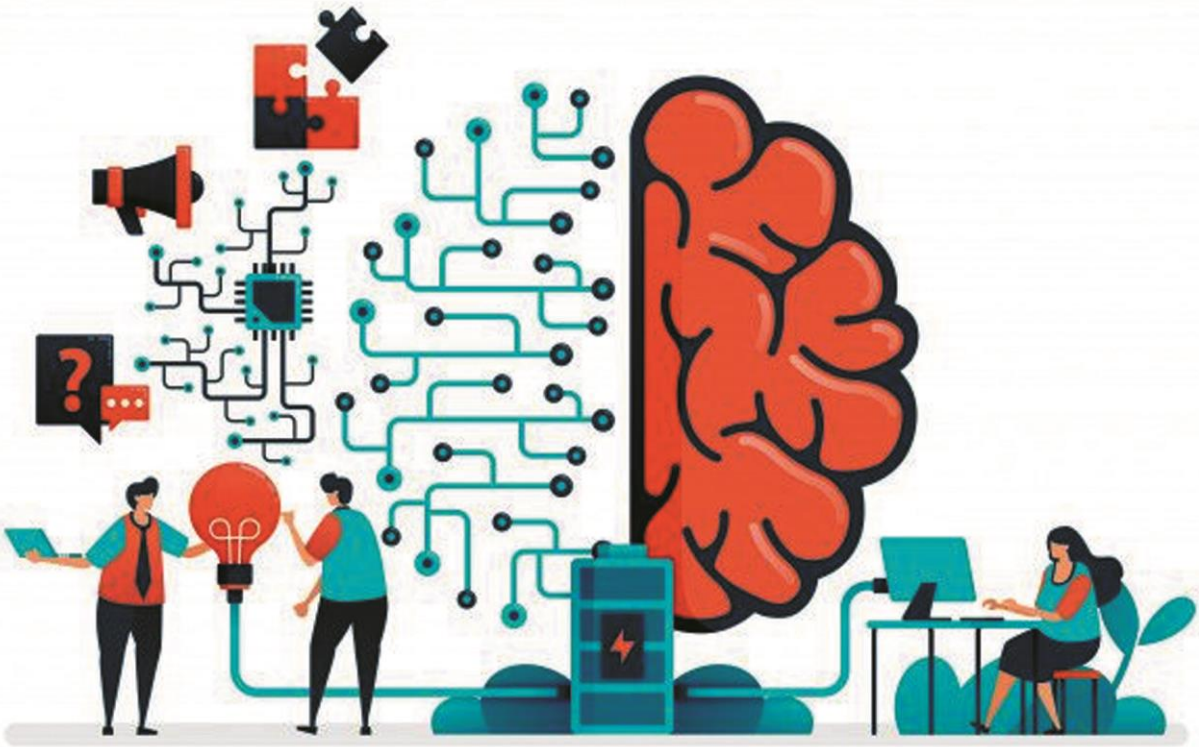
التعلم الآلي

عن طريق الامثلة

٥٠ مشروع تعلم آلي تم حلها وشرحها باستخدام بايثون

ترجمة واعداد: د. علاء طعيمة



بـــسم تعالى

التعلم الآلي: عن طريق الامثلة

50 مشروع تعلم آلي تم حلها وشرحها باستخدام بايثون

ترجمة واعداد:

د. علاء طعيمة

مقدمة المؤلف

يعمل التعلم الآلي على إحداث ثورة تدريجيةفي كل مجال من مجالات الذكاء الاصطناعي، مما يجعل تطوير التطبيقات أسهل.

في هذا الكتاب، تنقل مشاريع التعلم الآلي باستخدام بايثون كل المعرفة اللازمة لتنفيذ مشاريع التعلم الآليفي مختلف مجالات التعلم الآلي. كل مشروع من هذه المشاريع فريد من نوعه، مما يساعدك على إتقان الموضوع تدريجيًا. ستتعلم كيفية تشخيص مجموعة متنوعة من الامراض باستخدام التعلم الآلي وكذلك ستتعلم الكثير من المشاريع الجذابة الاخرى الي ستساعدكفي اكتساب المعرفة لتطوير أنظمة التعلم الآلي الخاصة بك بطريقة مباشرة وفعالة.

لقد حاولت قدر المستطاع ان اترجم المشاريع الأكثر طرحاًفي مجال التعلم الآلي مع الشرح المناسب والكافي، ومع هذا يبقى عملاً بشرياً يحتمل النقص، فاذاكان لديك أي ملاحظات حول هذا الكتاب، فلا تتردد بمراسلتنا عبر بريدنا الالكتروني alaa.taima@qu.edu.iq .

نأمل ان يساعد هذا الكتاب كل من يريد ان يدخلفي مجالات التعلم الآلي والتعلم العميق وعلم البيانات ومساعدة القارئ العربي على تعلم هذا المجالات. اسأل الله التوفيقفي هذا العمل لأثراء المحتوى العربي الذي يفتقر أشد الافتقار إلى محتوى جيد ورصينفي مجال التعلم الآلي والتعلم العميق وعلم البيانات. ونرجو لك الاستمتاع مع الكتاب ولا تنسونا من صالح الدعاء.

د. علاء طعيمة

كلية علوم الحاسوب وتكنولوجيا المعلومات

جامعة القادسية

العراق

المحتويات

0) مشروعك الأول للتعلم الآلي في بايثون خطوة بخطوة Your First Machine Learning Project in Python Step-By-Step

هل تريد تعلم الآلة باستخدام بايثون، لكنك تواجه مشكلةفي البدء؟

في هذا المنشور، ستكمل مشروعك الأول للتعلم الآلي باستخدام بايثون.

في هذا البرنامج التعليمي خطوة بخطوة سوف:

1. نقوم بتنزيل Python SciPy وتثبيته واحصل على الحزمة الأكثر فائدة للتعلم الآلي في بايثون.
2. نقوم بتحميل مجموعة بيانات وافهم هيكلها باستخدام الملخصات الإحصائية ورسم البيانات.
3. نقوم بإنشاء 6 نماذج للتعلم الآلي، واختيار الأفضل وبناء الثقةفي أن الدقة معتمدة.

إذاكنت مبتدئًافي تعلم الآلة وتتطلع إلى البدء أخيرًافي استخدام بايثون، فقد تم تصميم هذا البرنامج التعليمي من أجلك.

كيف تبدأ التعلم الآلي في بايثون؟

أفضل طريقة لتعلم التعلم الآلي هي من خلال تصميم واستكمال المشاريع الصغيرة.

يمكن أن تكون لغة بايثون مخيفة عند البدء.

بايثون هي لغة مفسرة شائعة وقوية. على عكس R، فإن بايثون هي لغة ومنصة كاملة يمكنك استخدامها للبحث والتطوير وتطوير أنظمة الإنتاج.

هناك أيضًا الكثير من الوحدات النمطية والمكتبات للاختيار من بينها، مما يوفر طرقًا متعددة للقيام بكل مهمة. يمكن أن تشعر بالإرهاق.

أفضل طريقة لبدء استخدام بايثون للتعلم الآلي هي إكمال مشروع.

- سيجبرك على تثبيت وبدء مترجم بايثون (على الأقل).
- سوف يمنحك نظرة عامة على كيفية تنفيذ مشروع صغير.
- سيمنحك هذا الثقة، وربما تستمرفي مشاريعك الصغيرة.

يحتاج المبتدئون إلى مشروع صغير شامل

الكتب والدورات محبطة. يعطونك الكثير من الوصفات والمقتطفات، لكنك لن ترى أبدًا كيف تتلاءم جميعًا معًا.

عندما تقوم بتطبيق التعلم الآلي على مجموعات البيانات الخاصة بك، فأنت تعملفي مشروع.

قد لا يكون مشروع التعلم الآلي خطيًا، ولكن يحتوي على عدد من الخطوات المعروفة جيداً:

- تحديد المشكلة Define Problem.
- تحضير البيانات Prepare Data.
- تقييم الخوارزميات Evaluate Algorithms.
- تحسين النتائج Improve Results.
- تقديم النتائج Present Results.

أفضل طريقة للتصالح مع نظام أساسي أو أداة جديدة هي العمل من خلال مشروع التعلم الآلي من البداية إلى النهاية وتغطية الخطوات الرئيسية. أي من تحميل البيانات وتلخيص البيانات وتقييم الخوارزميات وعمل بعض التنبؤات.

إذا كان بإمكانك القيام بذلك، فلديك نموذج يمكنك استخدامهفي مجموعة البيانات بعد مجموعة البيانات. يمكنك ملء الفجوات مثل إعداد المزيد من البيانات وتحسين مهام النتائج لاحقًا، بمجرد أن تكتسب المزيد من الثقة.

hello world للتعلم الآلي

أفضل مشروع صغير للبدء به على أداة جديدة هو تصنيف زهور iris (مثل the iris dataset).

هذا مشروع جيد لأنه مفهوم جيداً.

- السمات رقمية لذا عليك معرفة كيفية تحميل البيانات ومعالجتها.
- إنها مشكلة تصنيف، مما يسمح لك بالتدرب على نوع أسهل من خوارزمية التعلم الخاضع للإشراف.
- إنها مشكلة تصنيف متعددة الفئات (متعددة الأسماء) قد تتطلب بعض المعالجة المتخصصة.
- يحتوي فقط على 4 سمات و150 صفًا، مما يعني أنه صغير ويسهل وضعهفي الذاكرة (وشاشة أو صفحة A4).
- جميع السمات الرقمية موجودةفي نفس الوحدات ونفس المقياس، ولا تتطلب أي تحجيم أو تحويلات خاصة للبدء.

لنبدأ بمشروع hello world للتعلم الآليفي بايثون.

تعلم الآلة في بايثون: تعليمي خطوة بخطوة

في هذا القسم، سنعمل من خلال مشروع صغير للتعلم الآلي من البداية إلى النهاية.

فيما يلي نظرة عامة على ما سنقوم بتغطيته:

1. تثبيت منصة بايثون وSciPy.

2. تحميل مجموعة البيانات.
3. تلخيص مجموعة البيانات.
4. رسم مجموعة البيانات.
5. تقييم بعض الخوارزميات.
6. عمل بعض التوقعات.

خذ وقتك. اعمل من خلال كل خطوة.

حاول كتابة الأوامر بنفسك أو نسخ الأوامر ولصقها لتسريع الأمور.

1. تنزيل وتثبيت وبدء Python SciPy

قم بتثبيت نظامي بايثون وSciPy على نظامك إذا لم يكن مثبتًا بالفعل.

لا أريد أن أغطي هذا بتفصيل كبير، لأن الآخرين فعلوا ذلك بالفعل. هذا بالفعل واضح ومباشر، خاصة إذا كنت مطورًا.

1.1 تثبيت مكتبات SciPy

يفترض هذا البرنامج التعليمي إصدار Python 2.7 أو 3.6+.

هناك 5 مكتبات رئيسية ستحتاج إلى تثبيتها. فيما يلي قائمة بمكتبات Python SciPy المطلوبة لهذا البرنامج التعليمي:

- scipy
- numpy
- matplotlib
- pandas
- sklearn

توجد طرق عديدة لتثبيت هذه المكتبات. أفضل نصيحتي هو اختيار طريقة واحدة ثم تكون متسقةفي تثبيت كل مكتبة.

توفر صفحة التثبيت scipy إرشادات ممتازة لتثبيت المكتبات المذكورة أعلاه على العديد من الأنظمة الأساسية المختلفة، مثل Linux وmac OS X وWindows. إذا كانت لديك أية شكوك أو أسئلة، ارجع إلى هذا الدليل، فقد اتبعه آلاف الأشخاص.

في نظام التشغيل Mac OS X، يمكنك استخدام macports لتثبيت Python 3.6 وهذه المكتبات. لمزيد من المعلومات حول macports ، انظر الصفحة الرئيسية.

على Linux، يمكنك استخدام مدير الحزم الخاص بك، مثل yum على Fedora لتثبيت RPMs.

إذا كنت تستخدم نظام Windows أو لم تكن واثقًا، فإنني أوصي بتثبيت الإصدار المجاني من Anaconda الذي يتضمن كل ما تحتاجه.

ملاحظة: يفترض هذا البرنامج التعليمي أن لديك الإصدار 0.20 من scikit-Learn مثبتًا أو إصدارًا أعلى.

هل تريد المزيد من المساعدة؟ شاهد أحد هذه الدروس:

- كيفية إعداد بيئة Python للتعلم الآلي باستخدام Anaconda
- كيفية إنشاء آلة افتراضية على نظام Linux للتعلم الآلي باستخدام Python 3

2.1 ابدأ بايثون وتحقق من الإصدارات

إنها لفكرة جيدة أن تتأكد من أن بيئة بايثون الخاصة بك قد تم تثبيتها بنجاح وأنها تعمل كما هو متوقع.

سيساعدك السكريبت أدناه على اختبار بيئتك. يقوم باستيراد كل مكتبة مطلوبةفي هذا البرنامج التعليمي ويطبع الإصدار.

افتح سطر أوامر وابدأ تشغيل مترجم Python:

```
python
```

أوصي بالعمل مباشرةفي المترجم الفوري أو كتابة السكريبتات الخاصة بك وتشغيلهافي سطر الأوامر بدلاً من المحررين الكبار وIDEs. اجعل الأمور بسيطة وركز على التعلم الآلي وليس سلسلة الأدوات.

اكتب أو انسخ وألصق السكريبت التالي:

```
# Check the versions of libraries
 # Python version
import sys
print('Python: {}'.format(sys.version))
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
# scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.__version__))
```

هذا هو الناتج الذي أحصل عليه على التيرميتال لـ OS X الخاصة بي:

```
Python: 3.6.11 (default, Jun 29 2020, 13:22:26)
```

```
[GCC 4.2.1 Compatible Apple LLVM 9.1.0 (clang-902.0.39.2)]
scipy: 1.5.2
numpy: 1.19.1
matplotlib: 3.3.0
pandas: 1.1.0
sklearn: 0.23.2
```

قارن الناتج أعلاه بإصداراتك.

من الناحية المثالية، يجب أن تتطابق إصداراتك أو تكون أحدث. لا تتغير واجهات برمجة التطبيقات بسرعة، لذلك لا تقلق كثيرًا إذا كنت متأخرًا عن إصدارات قليلة، فمن المحتمل جدًا أن يظل كل شيءفي هذا البرنامج التعليمي مناسبًا لك.

إذا تلقيت خطأ، توقف. حان الوقت لإصلاحه.

إذا لم تتمكن من تشغيل السكريبت أعلاه بشكل نظيف، فلن تتمكن من إكمال هذا البرنامج التعليمي.

أفضل نصيحتي هي البحثفي Google عن رسالة الخطأ أو نشر سؤال على Stack Exchange.

2. تحميل البيانات

سنستخدم مجموعة بيانات زهور iris. مجموعة البيانات هذه مشهورة لأنها تُستخدم كمجموعة بيانات "hello world"في التعلم الآلي والإحصاءات من قِبل الجميع تقريبًا.

تحتوي مجموعة البيانات على 150 ملاحظة لزهور iris. هناك أربعة أعمدة لقياسات الزهور بالسنتيمتر. العمود الخامس هو نوع الزهرة التي تمت ملاحظتها. تنتمي جميع الزهور التي تمت ملاحظتها إلى واحد من ثلاثة أنواع.

يمكنك معرفة المزيد حول مجموعة البيانات هذه على ويكيبيديا.

في هذه الخطوة، سنقوم بتحميل بيانات iris من عنوان URL لملف CSV.

1.2 استيراد المكتبات

أولاً، دعنا نستورد جميع الوحدات والدوال والكائنات التي سنستخدمهافي هذا البرنامج التعليمي.

```
# Load libraries
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
...
```

يجب أن يتم تحميل كل شيء بدون أخطاء. إذا كان لديك خطأ، توقف. أنت بحاجة إلى بيئة عمل SciPy قبل المتابعة. راجع النصيحة أعلاه حول إعداد بيئتك.

2.2 تحميل مجموعة البيانات

يمكننا تحميل البيانات مباشرة من مستودع UCI Machine Learning.

نحن نستخدم pandas لتحميل البيانات. سنستخدم أيضًا pandas بعد ذلك لاستكشاف البيانات مع كل من الإحصاءات الوصفية ورسم البيانات.

لاحظ أننا نحدد أسماء كل عمود عند تحميل البيانات. سيساعد هذا لاحقًا عندما نستكشف البيانات.

```
...
# Load dataset
url =
"https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
dataset = read_csv(url, names=names)
```

يجب تحميل مجموعة البيانات دون وقوع حوادث.

إذا كانت لديك مشكلاتفي الشبكة، فيمكنك تنزيل ملف iris.csv إلى دليل العمل الخاص بك وتحميله باستخدام نفس الطريقة، وتغيير عنوان URL إلى اسم الملف المحلي.

3. تلخيص مجموعة البيانات

حان الوقت الآن لإلقاء نظرة على البيانات.

في هذه الخطوة، سنلقي نظرة على البيانات بعدة طرق مختلفة:

1. أبعاد مجموعة البيانات.
2. ألق نظرة خاطفة على البيانات نفسها.
3. ملخص إحصائي لجميع الميزات.
4. تفصيل البيانات حسب متغير الفئة.

لا تقلق، فكل نظرة على البيانات هي أمر واحد. هذه أوامر مفيدة يمكنك استخدامها مرارًا وتكرارًا في المشاريع المستقبلية.

1.3 أبعاد مجموعة البيانات

يمكننا الحصول على فكرة سريعة عن عدد الحالات (الصفوف) وعدد السمات (الأعمدة) التي تحتوي عليها البيانات مع خاصية الشكل.

```
...
# shape
print(dataset.shape)
```

يجب أن تشاهد 150 حالة و5 سمات:

```
(150, 5)
```

2.3 نظرة خاطفة على البيانات

إنها أيضًا فكرة جيدة دائمًا أن تراقب بياناتك.

```
...
# head
print(dataset.head(20))
```

يجب أن تشاهد أول 20 صفاً من البيانات:

	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa

3.3 ملخص إحصائي

يمكننا الآن إلقاء نظرة على ملخص لكل سمة.

يتضمن هذا العدد والمتوسط والقيم الدنيا والحد الأقصى بالإضافة إلى بعض النسب المئوية.

```
...
# descriptions
print(dataset.describe())
```

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

4.3 توزيعات الفئات

دعنا الآن نلقي نظرة على عدد الأمثلة (الصفوف) التي تنتمي إلى كل فئة. يمكننا اعتبار هذا عددًا مطلقًا.

```
...
# class distribution
print(dataset.groupby('class').size())
```

يمكننا أن نرى أن كل فئة لديها نفس العدد من المثيلات (50 أو 33٪ من مجموعة البيانات).

```
class
Iris-setosa         50
Iris-versicolor     50
Iris-virginica      50
```

5.3 مثال كامل

كمرجع، يمكننا ربط جميع العناصر السابقة معًا في نص واحد.

المثال الكامل مدرج أدناه:

```
# summarize the data
from pandas import read_csv
# Load dataset
url =
"https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
dataset = read_csv(url, names=names)
# shape
print(dataset.shape)
# head
print(dataset.head(20))
# descriptions
print(dataset.describe())
# class distribution
print(dataset.groupby('class').size())
```

4. رسم البيانات

لدينا الآن فكرة أساسية عن البيانات. نحن بحاجة إلى توسيع ذلك ببعض التمثيل المرئي.

سنلقي نظرة على نوعين من التمثيل المرئي:

1. المخططات أحادية المتغير Univariate plots لفهم كل سمة بشكل أفضل.
2. المخططات متعددة المتغيرات Multivariate plots لفهم العلاقات بين السمات بشكل أفضل.

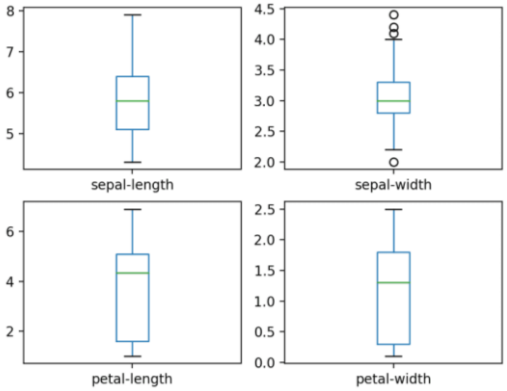
1.4 المخططات أحادية المتغير

نبدأ ببعض المخططات أحادية المتغير، أي قطع كل متغير فردي.

نظرًا لأن متغيرات الإدخال رقمية، يمكننا إنشاء مخططات صندوقية لكل منها.

```
...
# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False,
sharey=False)
pyplot.show()
```

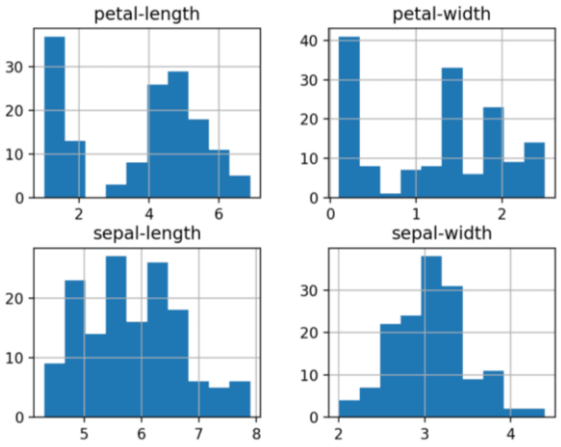
يعطينا هذا فكرة أوضح عن توزيع السمات المدخلة:



يمكننا أيضًا إنشاء هيستوكرام لكل متغير إدخال للحصول على فكرة عن التوزيع.

```
...
# histograms
dataset.hist()
pyplot.show()
```

يبدو أن اثنين من متغيرات الإدخال لهما توزيع غاوسي. من المفيد ملاحظة ذلك حيث يمكننا استخدام الخوارزميات التي يمكنها استغلال هذا الافتراض.



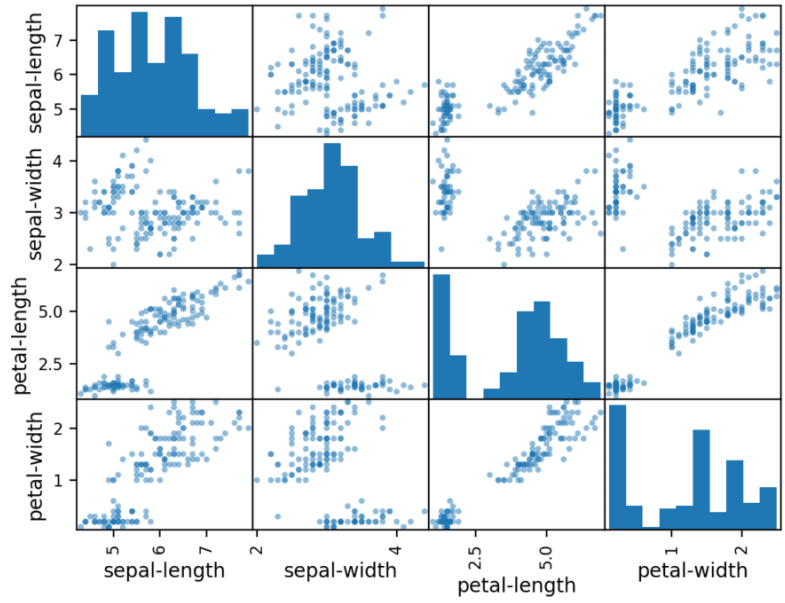
2.4 المخططات متعددة المتغيرات

الآن يمكننا النظر إلى التفاعلات بين المتغيرات.

أولاً، دعنا نلقي نظرة على المخططات المبعثرة scatterplots لجميع أزواج السمات. يمكن أن يكون هذا مفيدًا في تحديد العلاقات المنظمة بين متغيرات الإدخال.

```
...
# scatter plot matrix
scatter_matrix(dataset)
pyplot.show()
```

لاحظ التجميع القطري لبعض أزواج السمات. هذا يشير إلى وجود ارتباط كبير وعلاقة يمكن التنبؤ بها.



3.4 مثال كامل

كمرجع، يمكننا ربط جميع العناصر السابقة معًافي نص واحد.

المثال الكامل مدرج أدناه:

```
# visualize the data
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
# Load dataset
url =
"https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
dataset = read_csv(url, names=names)
# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False,
sharey=False)
pyplot.show()
# histograms
dataset.hist()
pyplot.show()
# scatter plot matrix
scatter_matrix(dataset)
pyplot.show()
```

5. تقييم بعض الخوارزميات

حان الوقت الآن لإنشاء بعض نماذج البيانات وتقدير دقتها على البيانات غير المرئية.

إليك ما سنقوم بتغطيتهفي هذه الخطوة:

1. افصل مجموعة بيانات التحقق.
2. قم بإعداد اختبار تسخير test harness لاستخدام التحقق المتقاطع بمقدار 10 أضعاف 10 ـ fold cross validation.
3. بناء نماذج مختلفة متعددة للتنبؤ بالأنواع من قياسات الزهور.
4. حدد أفضل موديل.

1.5 إنشاء مجموعة بيانات التحقق من الصحة

نحتاج أن نعرف أن النموذج الذي أنشأناه جيد.

لاحقًا، سوف نستخدم طرقًا إحصائية لتقدير دقة النماذج التي نقوم بإنشائها على بيانات غير مرئية. نريد أيضًا تقديرًا أكثر واقعية لدقة أفضل نموذج على البيانات غير المرئية من خلال تقييمها على بيانات فعلية غير مرئية.

أي أننا سنمنع بعض البيانات التي لن تتمكن الخوارزميات من رؤيتها وسنستخدم هذه البيانات للحصول على فكرة ثانية ومستقلة عن مدى دقة النموذج الأفضلفي الواقع.

سنقسم مجموعة البيانات المحملة إلى مجموعتين، 80٪ منها سنستخدمها للتدريب والتقييم والاختيار من بين نماذجنا، و20٪ سنحجمها كمجموعة بيانات للتحقق.

```
...
# Split-out validation dataset
array = dataset.values
X = array[:,0:4]
y = array[:,4]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1)
```

لديك الآن بيانات تدريبفي X_train وY_train لإعداد النماذج ومجموعات X_validation و Y_validation يمكننا استخدامها لاحقًا.

لاحظ أننا استخدمنا شريحة بايثون لتحديد الأعمدةفي مصفوفة NumPy. إذا كان هذا جديدًا بالنسبة لك، فقد ترغبفي التحقق من هذا المنشور:

- كيفية فهرسة مصفوفات NumPy وتقسيمها وإعادة تشكيلها للتعلم الآليفي Python.

2.5 اختبار تسخير Test Harness

سنستخدم 10-fold cross validation لتقدير دقة النموذج.

سيؤدي هذا إلى تقسيم مجموعة البيانات الخاصة بنا إلى 10 أجزاء، والتدريب على 9 واختبار على 1 والتكرار لجميع مجموعات تقسيمات التدريب ـ الاختبار.

يعني الطبقي أن كل طية أو تقسيم لمجموعة البيانات ستهدف إلى الحصول على نفس توزيع المثال حسب الفئة كما هو موجودفي مجموعة بيانات التدريب بأكملها.

لمزيد من المعلومات حول تقنية k-fold cross-validation، راجع البرنامج التعليمي:

- مقدمة لطيفة لـ k-fold cross validation.

قمنا بتعيين البذرة العشوائية عبر وسيطة random_state على رقم ثابت لضمان تقييم كل خوارزمية على نفس الانقسامات لمجموعة بيانات التدريب.

لا يهم البذور العشوائية المحددة، تعرف على المزيد حول مولدات الأرقام العشوائية الزائفة هنا:

- مقدمة لمولدات الأرقام العشوائية للتعلم الآليفي بايثون.

نحن نستخدم مقياس "الدقة accuracy " لتقييم النماذج.

هذه نسبة من عدد الحالات التي تم التنبؤ بها بشكل صحيح مقسومًا على العدد الإجمالي للمثيلاتفي مجموعة البيانات مضروبًافي 100 لإعطاء نسبة مئوية (على سبيل المثال دقة 95٪). سنستخدم متغير scoring عند تشغيل بناء وتقييم كل نموذج بعد ذلك.

3.5 بناء النماذج

لا نعرف أي الخوارزميات ستكون جيدةفي هذه المشكلة أو ما هي التكوينات التي يجب استخدامها.

حصلنا على فكرة من المخططات أن بعض الفئات يمكن فصلها جزئيًا خطيًافي بعض الأبعاد، لذلك نتوقع نتائج جيدة بشكل عام.

دعونا نختبر 6 خوارزميات مختلفة:

- Logistic Regression (LR)
- Linear Discriminant Analysis (LDA)
- K-Nearest Neighbors (KNN).
- Classification and Regression Trees (CART).
- Gaussian Naive Bayes (NB).
- Support Vector Machines (SVM).

هذا مزيج جيد من الخوارزميات الخطية البسيطة (LR وLDA) وغير الخطية (KNN وCART وNB وSVM).

دعونا نبني نماذجنا ونقيمها:

```
...
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression(solver='liblinear',
multi_class='ovr')))
```

```
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
# evaluate each model in turn
results = []
names = []
for name, model in models:
        kfold = StratifiedKFold(n_splits=10, random_state=1,
shuffle=True)
        cv_results = cross_val_score(model, X_train, Y_train,
cv=kfold, scoring='accuracy')
        results.append(cv_results)
        names.append(name)
        print('%s: %f (%f)' % (name, cv_results.mean(),
cv_results.std()))
```

4.5 تحديد أفضل نموذج

لدينا الآن 6 نماذج وتقديرات دقة لكل منها. نحتاج إلى مقارنة النماذج مع بعضها البعض واختيار الأكثر دقة.

بتشغيل المثال أعلاه، نحصل على النتائج الأولية التالية:

```
LR: 0.960897 (0.052113)
LDA: 0.973974 (0.040110)
KNN: 0.957191 (0.043263)
CART: 0.957191 (0.043263)
NB: 0.948858 (0.056322)
SVM: 0.983974 (0.032083)
```

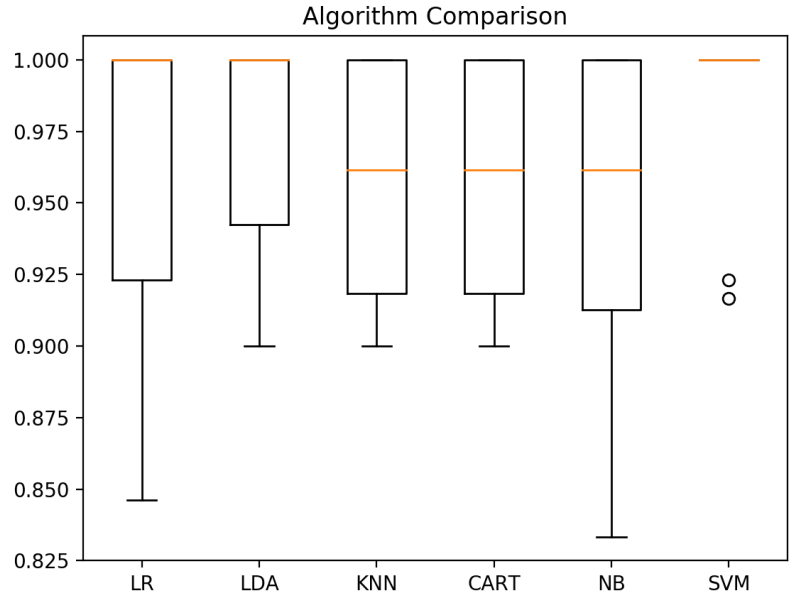
ملاحظة: قد تختلف نتائجك نظرًا للطبيعة العشوائية للخوارزمية أو إجراء التقييم، أو الاختلافات في الدقة العددية. ضع في اعتبارك تشغيل المثال عدة مرات وقارن النتيجة المتوسطة.

ما هي النتائج التي حصلت عليها؟

في هذه الحالة، يمكننا أن نرى أنه يبدو أن Support Vector Machines (SVM) لديها أكبر درجة دقة مقدرة بحوالي 0.98 أو 98٪.

يمكننا أيضًا إنشاء مخطط لنتائج تقييم النموذج ومقارنة الانتشار ومتوسط الدقة لكل نموذج. هناك مجموعة من مقاييس الدقة لكل خوارزمية لأنه تم تقييم كل خوارزمية 10 مرات (عن طريق 10-fold cross validation).

تتمثل إحدى الطرق المفيدة لمقارنة عينات النتائج لكل خوارزمية في إنشاء مخطط صندوقي لكل توزيع ومقارنة التوزيعات.



5.5 مثال كامل

كمرجع، يمكننا ربط جميع العناصر السابقة معًافي نص واحد.

المثال الكامل مدرج أدناه:

```
# compare algorithms
from pandas import read csv
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
# Load dataset
url =
"https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
dataset = read_csv(url, names=names)
# Split-out validation dataset
array = dataset.values
X = array[:,0:4]
y = array[:,4]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1, shuffle=True)
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression(solver='liblinear',
multi_class='ovr')))
```

```
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
# evaluate each model in turn
results = []
names = []
for name, model in models:
        kfold = StratifiedKFold(n_splits=10, random_state=1,
shuffle=True)
        cv_results = cross_val_score(model, X_train, Y_train,
cv=kfold, scoring='accuracy')
        results.append(cv_results)
        names.append(name)
        print('%s: %f (%f)' % (name, cv_results.mean(),
cv_results.std()))
# Compare Algorithms
pyplot.boxplot(results, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()
```

6. التنبؤات

يجب أن نختار خوارزمية لاستخدامها لعمل تنبؤات.

تشير النتائجفي القسم السابق إلى أن SVM ربما كان النموذج الأكثر دقة. سوف نستخدم هذا النموذج كنموذج نهائي لدينا.

نريد الآن الحصول على فكرة عن دقة النموذجفي مجموعة التحقق الخاصة بنا.

سيعطينا هذا فحصًا نهائيًا مستقلًا لدقة أفضل نموذج. من المهم الاحتفاظ بمجموعة التحقق فقطفي حالة حدوث زلة أثناء التدريب، مثل الضبط الزائد overfitting لمجموعة التدريب أو تسرب البيانات. كلتا هاتين المسألتين ستؤديان إلى نتيجة مفرطةفي التفاؤل.

6.1 عمل التنبؤات

يمكننا ملاءمة النموذج على مجموعة بيانات التدريب بأكملها وإجراء تنبؤات على مجموعة بيانات التحقق.

```
...
# Make predictions on validation dataset
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
```

قد ترغب أيضًافي عمل تنبؤات لصفوف فردية من البيانات. للحصول على أمثلة حول كيفية القيام بذلك، راجع البرنامج التعليمي:

- كيفية عمل تنبؤات باستخدام scikit-Learn.

قد ترغب أيضًافي حفظ النموذجفي ملف وتحميله لاحقًا لعمل تنبؤات بشأن البيانات الجديدة. للحصول على أمثلة حول كيفية القيام بذلك، راجع البرنامج التعليمي:

- حفظ وتحميل نماذج التعلم الآليفي Python باستخدام scikit-Learn.

2.6 تقييم التنبؤات

يمكننا تقييم التنبؤات من خلال مقارنتها بالنتائج المتوقعةفي مجموعة التحقق، ثم حساب دقة التصنيف، بالإضافة إلى مصفوفة الارتباك confusion matrix وتقرير التصنيف classification report.

```
....
# Evaluate predictions
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

يمكننا أن نرى أن الدقة تبلغ 0.966 أو حوالي 96٪في مجموعة البيانات المعلقة.

توفر مصفوفة الارتباك إشارة إلى الأخطاء التي تم ارتكابها.

أخيرًا، يوفر تقرير التصنيف تفصيلًا لكل فئة من خلال precision وrecall وf1-score وsupport الذي يُظهر نتائج ممتازة (تم منح مجموعة بيانات التحقق من الصحة صغيرة).

```
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

3.6 مثال كامل

كمرجع، يمكننا ربط جميع العناصر السابقة معًافي نص واحد.

المثال الكامل مدرج أدناه:

```
# make predictions
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
# Load dataset
url =
"https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
```

```
dataset = read_csv(url, names=names)
# Split-out validation dataset
array = dataset.values
X = array[:,0:4]
y = array[:,4]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1)
# Make predictions on validation dataset
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
# Evaluate predictions
print(accuracy score(Y validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

يمكنك تعلم الآلة في بايثون

العمل من خلال البرنامج التعليمي أعلاه. سيستغرق الأمر من 5 إلى 10 دقائق كحد أقصى!

أنت لا تحتاج إلى فهم كل شيء. (على الأقل ليس الآن) هدفك هو تشغيل البرنامج التعليمي من البداية إلى النهاية والحصول على نتيجة. لا تحتاج إلى فهم كل شيءفي المسار الأول. ضع قائمة بأسئلتك كما تذهب. استفد بشدة من المساعدة ("FunctionName") التي تساعدفي بناء الجملةفي Python للتعرف على جميع الدوال التي تستخدمها.

لا تحتاج إلى معرفة كيفية عمل الخوارزميات. من المهم معرفة القيود وكيفية تكوين خوارزميات التعلم الآلي. لكن التعرف على الخوارزميات يمكن أن يأتي لاحقًا. تحتاج إلى بناء معرفة الخوارزمية هذه ببطء على مدى فترة طويلة من الزمن. اليوم، ابدأ بالراحة مع المنصة.

لست بحاجة إلى أن تكون مبرمجًافي بايثون. يمكن أن يكون بناء جملة لغة Python بديهيًا إذا كنت جديدًا عليها. تمامًا مثل اللغات الأخرى، ركز على استدعاءات الوظائف (مثل الوظيفة ()) والمهام (مثل "b" = a). هذا سوف يوصلك إلى أقصى حد. أنت مطور، تعرف كيف تلتقط أساسيات اللغة بسرعة حقيقية. فقط ابدأ وتعمقفي التفاصيل لاحقًا.

لست بحاجة إلى أن تكون خبيرًافي التعلم الآلي. يمكنك التعرف على مزايا وقيود الخوارزميات المختلفة لاحقًا، وهناك الكثير من المنشورات التي يمكنك قراءتها لاحقًا لصقل خطوات مشروع التعلم الآلي وأهمية تقييم الدقة باستخدام التحقق المتقاطع.

ماذا عن الخطوات الأخرىفي مشروع التعلم الآلي. لم نغطي جميع الخطواتفي مشروع التعلم الآلي لأن هذا هو مشروعك الأول ونحتاج إلى التركيز على الخطوات الرئيسية. وهي تحميل البيانات والنظرفي البيانات وتقييم بعض الخوارزميات وعمل بعض التنبؤات.في البرامج التعليمية اللاحقة، يمكننا إلقاء نظرة على مهام إعداد البيانات وتحسين النتائج الأخرى.

الملخص

في هذا المنشور، اكتشفت كيفية إكمال مشروعك الأول للتعلم الآليفي Python خطوة بخطوة.

لقد اكتشفت أن إكمال مشروع صغير من البداية إلى النهاية بدءًا من تحميل البيانات وحتى إجراء التنبؤات هو أفضل طريقة للتعرف على نظام أساسي جديد.

خطوتك التالية

هل تعمل من خلال البرنامج التعليمي؟

1. العمل من خلال البرنامج التعليمي أعلاه.
2. ضع قائمة بأي أسئلة لديك.
3. ابحث عن الإجابات أو ابحث عنها.
4. تذكر أنه يمكنك استخدام المساعدة ("FunctionName")في Python للحصول على مساعدةفي أي دالة.

1) تصنيف زهرة Iris باستخدام التعلم الآلي Iris Flower Classification using Machine Learning

يعد تصنيف زهرة Iris أحد أكثر دراسات الحالة شيوعًا بين مجتمع علوم البيانات. قام كل مبتدئ في علم البيانات تقريبًا بحل دراسة الحالة هذه مرة واحدةفي حياتهم. هنا، يتم إعطاؤك القياسات المرتبطة بكل نوع من أنواع زهرة Iris وبناءً على هذه البيانات، يجب عليك تدريب نموذج التعلم الآلي لمهمة تصنيف زهور Iris. لذلك إذا كنت جديداً على التعلم الآلي ولم تحاول أبدًا حل دراسة الحالة هذه، فهذه المقالة مناسبة لك.في هذه المقالة، سوف أطلعك على تصنيف زهرة Iris مع التعلم الآلي باستخدام Python.

تصنيف زهرة Iris

زهرة القزحية لها ثلاثة أنواع. setosa و versicolor و virginica ، والتي تختلف حسب قياساتها. افترض الآن أن لديك قياسات زهور Iris وفقًا لنوعها، وهنا تتمثل مهمتكفي تدريب نموذج تعلم آلي يمكنه التعلم من قياسات أنواع Iris وتصنيفها.

آمل أن تكون قد فهمت الآن دراسة الحالة الخاصة بتصنيف زهرة Iris. على الرغم من أن مكتبة Scikit-Learn توفر مجموعة بيانات لتصنيف زهرة Iris، يمكنك أيضًا تنزيل مجموعة البيانات نفسها من هنا لمهمة تصنيف زهرة Iris باستخدام التعلم الآلي. الآنفي القسم أدناه، سأطلعك على كيفية تصنيف أنواع زهرة Iris بالتعلم الآلي باستخدام لغة برمجة Python.

تصنيف زهرة Iris باستخدام بايثون

خطوات تصنيف زهرة Iris:

1. تحميل البيانات.
2. تحليل ورسم مجموعة البيانات.
3. تدريب النموذج.
4. تقييم النموذج.
5. اختبار النموذج.

الخطوة 1 : تحميل البيانات:

```
# DataFlair Iris Flower Classification
# Import Packages
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
%matplotlib inline
```

أولاً، قمنا باستيراد بعض الحزم الضرورية للمشروع.

- سيتم استخدام Numpy لأي عمليات حسابية.
- سنستخدم Matplotlib و seaborn لرسم البيانات.
- تساعد Pandas في تحميل البيانات من مصادر مختلفة مثل التخزين المحلي وقاعدة البيانات وملف Excel وملف CSV وما إلى ذلك.

```
columns = ['Sepal length', 'Sepal width', 'Petal length', 'Petal
width', 'Class_labels']
# Load the data
df = pd.read_csv('iris.data', names=columns)
df.head()
```

بعد ذلك، نقوم بتحميل البيانات باستخدام pd.read_csv() وتعيين اسم العمود وفقًا لمعلومات بيانات Iris.

- يقرأ Pd.read_csv ملفات CSV. يشير CSV إلى قيمة مفصولة بفاصلة.
- يعرض df.head() الصفوف الخمسة الأولى فقط من جدول مجموعة البيانات.

```
In [7]: df.head()
```

Out[7]:

	Sepal length	Sepal width	Petal length	Petal width	Class_labels
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

- جميع القيم العددية بالسنتيمتر.

الخطوة 2: تحليل مجموعة البيانات وتصورها:

دعنا نرى بعض المعلومات حول مجموعة البيانات.

```
# Some basic statistical analysis about the data
df.describe()
```

```
In [8]: # Some basic statistical analysis about the data
        df.describe()
```

Out[8]:

	Sepal length	Sepal width	Petal length	Petal width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

من هذا الوصف، يمكننا رؤية جميع الأوصاف المتعلقة بالبيانات، مثل متوسط الطول والعرض، والحد الأدنى للقيمة، والحد الأقصى للقيمة، وقيمة التوزيع 25٪، و 50٪، و 75٪، إلخ.

دعونا نرسم مجموعة البيانات.

```
# Visualize the whole dataset
sns.pairplot(df, hue='Class_labels')
```

- لنرسم مجموعة البيانات بأكملها، استخدمنا طريقة مخطط الزوج من seaborn. يرسم معلومات مجموعة البيانات بأكملها.

- من هذا الرسم، يمكننا أن نقول أن iris-setosa منفصلة جيدًا عن الزهرتين الأخريين.
- و iris virginica هي أطول زهرة و iris setosa هي الأقصر.

- دعنا الآن نرسم متوسط كل ميزة لكل فئة.

```
# Separate features and target
data = df.values
X = data[:,0:4]
Y = data[:,4]
```

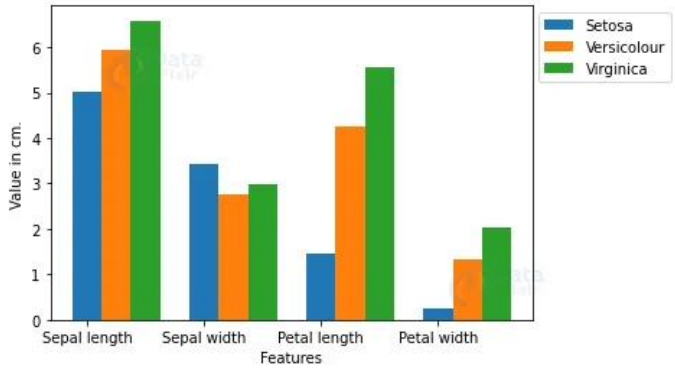
- هنا قمنا بفصل الميزات عن القيمة المستهدفة.

```
# Calculate average of each features for all classes
Y_Data = np.array([np.average(X[:, i][Y==j].astype('float32')) for i
in range (X.shape[1])
 for j in (np.unique(Y))])
Y_Data_reshaped = Y_Data.reshape(4, 3)
Y_Data_reshaped = np.swapaxes(Y_Data_reshaped, 0, 1)
X_axis = np.arange(len(columns)-1)
width = 0.25
```

- يحسب Np.average المتوسط من المصفوفة.
- استخدمنا هنا حلقتين for داخل قائمة. هذا هو المعروف باسم list comprehension.
- يساعد list comprehension على تقليل عدد أسطر الكود.
- Y_Data عبارة عن مصفوفة احادية البعد، لكن لدينا 4 ميزات لكل 3 فئات. لذلك قمنا بإعادة تشكيل Y_Data إلى مصفوفة على شكل (3 ، 4).
- ثم نقوم بتغيير محور المصفوفة المعاد تشكيلها.

```
# Plot the average
plt.bar(X_axis, Y_Data_reshaped[0], width, label = 'Setosa')
plt.bar(X_axis+width, Y_Data_reshaped[1], width, label =
'Versicolour')
plt.bar(X_axis+width*2, Y_Data_reshaped[2], width, label =
'Virginica')
plt.xticks(X_axis, columns[:4])
plt.xlabel("Features")
plt.ylabel("Value in cm.")
plt.legend(bbox_to_anchor=(1.3,1))
plt.show()
```

- استخدمنا matplotlib لإظهار المتوسطاتفي مخطط شريطي.



- استخدمنا matplotlib لإظهار المتوسطاتفي مخطط شريطي.

الخطوة 3: نموذج التدريب:

```
# Split the data to train and test dataset.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2)
```

- باستخدام train_test_split، قمنا بتقسيم البيانات بأكملها إلى مجموعات بيانات تدريب واختبار. سنستخدم لاحقًا مجموعة بيانات الاختبار للتحقق من دقة النموذج.

```
# Support vector machine algorithm
from sklearn.svm import SVC
svn = SVC()
svn.fit(X_train, y_train)
```

- هنا قمنا باستيراد مصنف آلة المتجهات الداعمة SVM من scikit-Learn.
- ثم أنشأنا كائنًا وأطلقنا عليه اسم svn.
- بعد ذلك، نقوم بتغذية مجموعة بيانات التدريب في الخوارزمية باستخدام طريقة svn.fit().

الخطوة 4: تقييم النموذج:

```
# Predict from the test dataset
predictions = svn.predict(X_test)

# Calculate the accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
```

- الآن نتوقع الفئات من مجموعة بيانات الاختبار باستخدام نموذجنا المدرب.
- ثم نتحقق من درجة الدقة للفئات المتوقعة.
- accuracy_score() تأخذ القيم الحقيقية والقيم المتوقعة وتعيد النسبة المئوية للدقة.

المخرجات:

```
0.9666666666666667
```

الدقة أعلى من 96٪.

دعنا الآن نرى تقرير التصنيف المفصل بناءً على مجموعة بيانات الاختبار.

```
# A detailed classification report
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))

                    precision    recall  f1-score   support

    Iris-setosa         1.00      1.00      1.00         9
Iris-versicolor         1.00      0.83      0.91        12
 Iris-virginica         0.82      1.00      0.90         9
```

```
accuracy                                    0.93         30
macro avg       0.94      0.94      0.94         30
weighted avg    0.95      0.93      0.93         30
```

- يقدم تقرير التصنيف تقريرًا مفصلاً عن التنبؤ.
- تحدد precision نسبة الإيجابيات الحقيقية إلى مجموع الإيجابيات الحقيقية الإيجابية والخاطئة.
- يحدد Recall نسبة الموجب الحقيقي إلى مجموع الموجب الحقيقي والسالب الخاطئ.
- F1-score هي متوسط precision وقيمة recall.
- Support هو عدد التكرارات الفعلية للفئة في مجموعة البيانات المحددة.

الخطوة 5: اختبار النموذج:

```
X_new = np.array([[3, 2, 1, 0.2], [  4.9, 2.2, 3.8, 1.1 ], [  5.3,
2.5, 4.6, 1.9 ]])
#Prediction of the species from the input vector
prediction = svn.predict(X_new)
print("Prediction of Species: {}".format(prediction))
```

- يمكننا حفظ النموذج باستخدام شكل pickle.
- ومرة أخرى يمكننا تحميل النموذج في أي برنامج آخر باستخدام pickle واستخدامه باستخدام model.prict للتنبؤ ببيانات Iris.

الملخص

في هذا المشروع، تعلمنا تدريب نموذج التعلم الآلي الخاص بنا تحت الإشراف باستخدام مع التعلم الآلي. من خلال هذا المشروع، تعرفنا على Iris Flower Classificationمشروع التعلم الآلي، وتحليل البيانات، والتمثيل المرئي للبيانات، وإنشاء النماذج، وما إلى ذلك.

2) توقع أسعار الكمبيوتر المحمول باستخدام التعلم الآلي Laptop Price Prediction using Machine Learning

التعلم الآلي هو فرع من فروع الذكاء الاصطناعي الذي يتعامل مع تنفيذ التطبيقات التي يمكن أن تقوم بالتنبؤفي المستقبل بناءً على البيانات السابقة. إذا كنت متحمسًا لعلوم البيانات أو ممارسًا، فستساعدك هذه المقالةفي بناء مشروع التعلم الآلي الشامل الخاص بك من البداية. هناك العديد من الخطوات المتضمنةفي بناء مشروع التعلم الآلي ولكن ليست كل الخطوات إلزامية لاستخدامها في مشروع واحد، وكل هذا يتوقف على البيانات.في هذه المقالة، سنبني مشروع توقع أسعار أجهزة الكمبيوتر المحمول ونتعرف على دورة حياة مشروع التعلم الآلي.

بيان المشكلة لتنبؤ بسعر الكمبيوتر المحمول

سنقوم بعمل مشروع لتنبؤ أسعار أجهزة الكمبيوتر المحمول. بيان المشكلة هو أنه إذا أراد أي مستخدم شراء جهاز كمبيوتر محمول، فيجب أن يكون تطبيقنا متوافقًا لتوفير سعر مبدئي للكمبيوتر المحمول وفقًا لتكوينات المستخدم. على الرغم من أنه يبدو وكأنه مشروع بسيط أو مجرد تطوير نموذج، إلا أن مجموعة البيانات التي لدينا صاخبة وتحتاج إلى الكثير من هندسة الميزات والمعالجة المسبقة التي ستثير اهتمامك بتطوير هذا المشروع.

مجموعة بيانات لتنبؤ بأسعار أجهزة الكمبيوتر المحمول

يمكنك تنزيل مجموعة البيانات من هنا. معظم الأعمدةفي مجموعة البيانات صاخبة noisy وتحتوي على الكثير من المعلومات. ولكن مع هندسة الميزات feature engineering التي تقوم بها، ستحصل على المزيد من النتائج الجيدة. المشكلة الوحيدة هي أننا نمتلك بيانات أقل ولكننا سنحصل على دقة جيدة بشأنها. الشيء الجيد الوحيد هو أنه من الأفضل أن يكون لديك بيانات كبيرة. سنقوم بتطوير موقع ويب يمكنه التنبؤ بسعر مبدئي لجهاز كمبيوتر محمول بناءً على تكوين المستخدم.

الفهم الأساسي لبيانات التنبؤ بسعر الكمبيوتر المحمول

الآن دعونا نبدأ العمل على مجموعة بياناتفي نوتبوك Jupyter الخاص بنا. الخطوة الأولى هي استيراد المكتبات وتحميل البيانات. بعد ذلك سنأخذ فهمًا أساسيًا للبيانات مثل شكلها وعينتها وهل هناك أي قيم فارغة موجودةفي مجموعة البيانات. يُعد فهم البيانات خطوة مهمة للتنبؤ أو أي مشروع للتعلم الآلي.

من الجيد عدم وجود قيم فارغة NULL values. ونحتاج إلى تغييرات طفيفةفي الوزن وعمود ذاكرة الوصول العشوائي لتحويلها إلى رقمية عن طريق إزالة الوحدة المكتوبة بعد القيمة. لذلك سنقوم بتنظيف البيانات هنا للحصول على أنواع الأعمدة الصحيحة.

```
data.drop(columns=['Unnamed: 0'],inplace=True)
```

```
## remove gb and kg from Ram and weight and convert the cols to
numeric
data['Ram'] = data['Ram'].str.replace("GB", "")
data['Weight'] = data['Weight'].str.replace("kg", "")
data['Ram'] = data['Ram'].astype('int32')
data['Weight'] = data['Weight'].astype('float32')
```

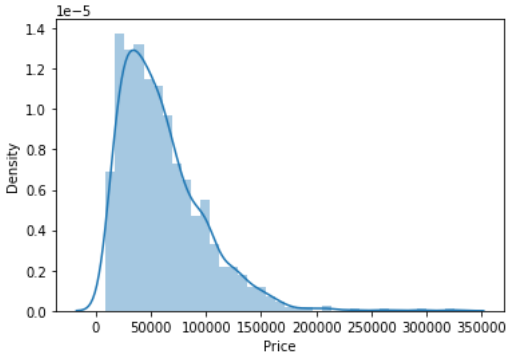
تحليل البيانات الاستكشافية لمجموعة بيانات توقع أسعار أجهزة الكمبيوتر المحمول

التحليل البيانات الاستكشافي Exploratory Data Analysis(EDA) هو عملية لاستكشاف وفهم علاقة البيانات والبيانات بعمق كامل بحيث تجعل خطوات هندسة الميزات ونمذجة التعلم الآلي سلسة ومبسطة للتنبؤ. تتضمن EDA التحليل أحادي المتغير أو ثنائي المتغير أو متعدد المتغيرات. تساعد EDA على إثبات صحة افتراضاتنا أو خطأها. بمعنى آخر، من المفيد إجراء اختبار الفرضيات. سنبدأ من العمود الأول ونستكشف كل عمود ونفهم التأثير الذي يحدثه على العمود الهدف.في الخطوة المطلوبة، سنقوم أيضًا بتنفيذ مهام المعالجة المسبقة وهندسة الميزات. هدفنافي أداء EDA المتعمق هو إعداد البيانات وتنظيفها لتحسين نمذجة التعلم الآلي لتحقيق أداء عالٍ ونماذج معممة. فلنبدأفي تحليل مجموعة البيانات وإعدادها للتنبؤ.

1) توزيع العمود الهدف

العمل مع توزيع العمود الهدف target column بيان مشكلة الانحدار مهم لفهمه.

```
sns.distplot(data['Price'])
plt.show()
```



توزيع المتغير المستهدف منحرف ومن الواضح أن السلع ذات الأسعار المنخفضة يتم بيعها وشرائها أكثر من السلع ذات العلامات التجارية.

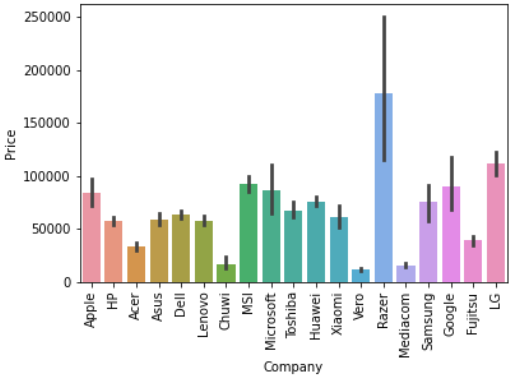
2) عمود الشركة Company column

نريد أن نفهم كيف يؤثر اسم العلامة التجارية على سعر الكمبيوتر المحمول أو ما هو متوسط سعر كل علامة تجارية لأجهزة الكمبيوتر المحمول؟ إذا قمت برسم مخطط تعداد (مخطط تواتر

frequency plot) لشركة ما، فإن الفئات الرئيسية الموجودة هي Lenovo وDell وHP وAsus وما إلى ذلك.

الآن إذا رسمنا علاقة الشركة بالسعر، فيمكنك ملاحظة كيف يختلف السعر باختلاف العلامات التجارية.

```
#what is avg price of each brand?

sns.barplot(x=data['Company'], y=data['Price'])

plt.xticks(rotation="vertical")

plt.show()
```

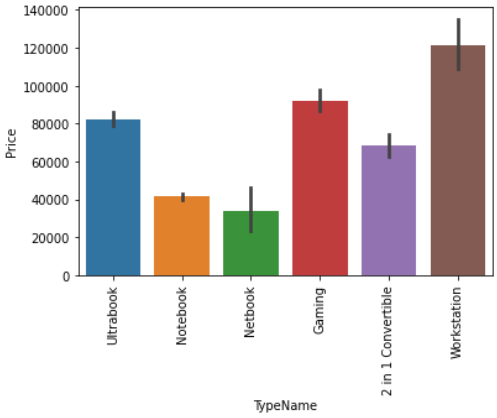


تعد أجهزة الكمبيوتر المحمولة Razer وApple وLG وMicrosoft وGoogle وMSI باهظة الثمن، والبعض الآخر في نطاق الميزانية.

3] نوع الكمبيوتر المحمول

ما نوع الكمبيوتر المحمول الذي تبحث عنه مثل كمبيوتر محمول للألعاب أو محطة عمل أو كمبيوتر محمول. نظرًا لأن الأشخاص الرئيسيين يفضلون الكمبيوتر المحمول لأنه يقع ضمن نطاق الميزانية ويمكن استنتاج نفس الشيء من بياناتنا.

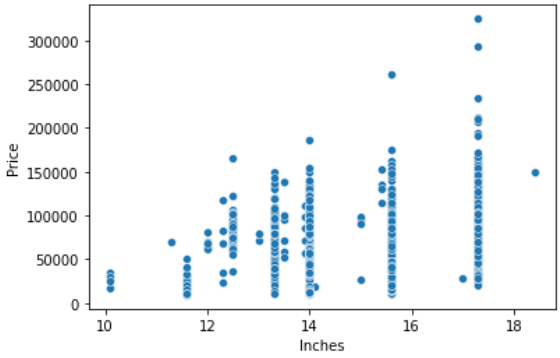
```
#data['TypeName'].value_counts().plot(kind='bar')
sns.barplot(x=data['TypeName'], y=data['Price'])
plt.xticks(rotation="vertical")
plt.show()
```



4) هل يختلف السعر باختلاف حجم الكمبيوتر المحمول بالبوصة؟

يتم استخدام مخطط المبعثر Scatter plot عندما يكون كلا العمودين رقميًا ويجيب على سؤالنا بطريقة أفضل. من المخطط أدناه يمكننا أن نستنتج أن هناك علاقة وليست علاقة قوية بين عمود السعر والحجم.

```
sns.scatterplot(x=data['Inches'],y=data['Price'])
```



هندسة الميزات والمعالجة المسبقة لنموذج التنبؤ بسعر الكمبيوتر المحمول

هندسة الميزات Feature engineering هي عملية لتحويل البيانات الأولية إلى معلومات مفيدة. هناك العديد من الطرق التي تندرج تحت هندسة الميزات مثل التحويل transformation والترميز الفئوي categorical encoding وما إلى ذلك. الآن الأعمدة التي لدينا صاخبة لذا نحتاج إلى تنفيذ بعض خطوات هندسة الميزات.

5) دقة الشاشة

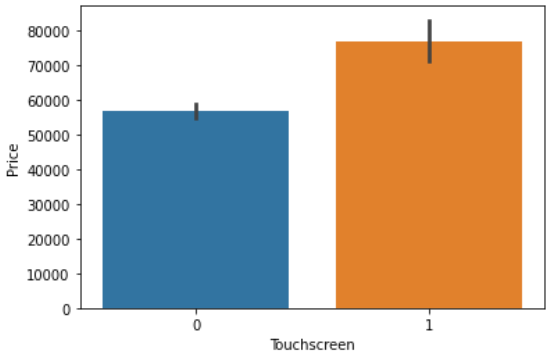
دقة الشاشة Screen Resolution تحتوي على الكثير من المعلومات. قبل أي تحليل أولاً، نحتاج إلى إجراء هندسة الميزات عليه. إذا لاحظت قيمًا فريدة للعمود، فيمكننا أن نرى أن كل القيم تعطي معلومات تتعلق بوجود لوحة IPS، وهي شاشة تعمل باللمس للكمبيوتر المحمول أم لا، ودقة شاشة المحور X والمحور Y. لذلك، سنقوم باستخراج العمود إلى 3 أعمدة جديدة في مجموعة البيانات.

استخراج معلومات شاشة اللمس

إنه متغير ثنائي لذا يمكننا ترميزه بالرقم 0 و 1. يعني أحدهما أن الكمبيوتر المحمول هو شاشة تعمل باللمس والصفر يشير إلى عدم وجود شاشة تعمل باللمس.

```
data['Touchscreen'] = data['ScreenResolution'].apply(lambda x:1 if
'Touchscreen' in x else 0)
#how many laptops in data are touchscreen
sns.countplot(data['Touchscreen'])
#Plot against price
sns.barplot(x=data['Touchscreen'],y=data['Price'])
```

إذا قمنا برسم عمود الشاشة التي تعمل باللمس مقابل السعر، فإن أجهزة الكمبيوتر المحمولة المزودة بشاشات تعمل باللمس تكون باهظة الثمن وهذا صحيحفي الحياة الواقعية.



استخراج معلومات وجود قناة IPS

إنه متغير ثنائي والرمز هو نفسه الذي استخدمناه أعلاه. أجهزة الكمبيوتر المحمولة المزودة بقناة IPS موجودة بشكل أقلفي بياناتنا ولكن من خلال مراقبة العلاقة مقابل سعر أجهزة الكمبيوتر المحمولة ذات قناة IPS مرتفعة.

```
#extract IPS column
data['Ips'] = data['ScreenResolution'].apply(lambda x:1 if 'IPS' in x
else 0)
sns.barplot(x=data['Ips'],y=data['Price'])
```

استخراج أبعاد دقة الشاشة المحور X والمحور Y

الآن كلا البُعد موجودانفي نهاية السلسلة النصية ويفصل بينهما بعلامة متقاطعة. لذلك سنقوم أولاً بتقسيم السلسلة التي تحتوي على مسافة والوصول إلى آخر سلسلة من القائمة. ثم قم بتقسيم السلسلة بعلامة متقاطعة والوصول إلى الصفر والأول للفهرس لأبعاد المحور X و Y.

```
def findXresolution(s):
  return s.split()[-1].split("x")[0]
def findYresolution(s):
  return s.split()[-1].split("x")[1]
#finding the x_res and y_res from screen resolution
data['X_res'] = data['ScreenResolution'].apply(lambda x:
findXresolution(x))
data['Y_res'] = data['ScreenResolution'].apply(lambda y:
findYresolution(y))
#convert to numeric
data['X_res'] = data['X_res'].astype('int')
data['Y_res'] = data['Y_res'].astype('int')
```

استبدال دقة البوصة، X و Y بـ PPI

إذا وجدت ارتباط الأعمدة بالسعر باستخدام طريقة corr، فيمكننا أن نرى أن البوصة ليس لها ارتباط قوي ولكن دقة المحور X و Y لها دقة عالية جدًا حتى نتمكن من الاستفادة منها وتحويل هذه الأعمدة الثلاثة إلى عمود واحد يُعرف بالبكسل لكل بوصة (PPI).في النهاية، هدفنا هو تحسين الأداء من خلال تقليل الميزات.

```
data['ppi'] = (((data['X_res']**2) +
(data['Y_res']**2))**0.5/data['Inches']).astype('float')
data.corr()['Price'].sort_values(ascending=False)
```

الآن عندما ترى ارتباط السعر، فإن مؤشر أسعار المنتجين له علاقة قوية.

```
data.corr()['Price'].sort_values(ascending=False)
```

```
Price          1.000000
Ram            0.743007
X_res          0.556529
Y_res          0.552809
ppi            0.473487
Ips            0.252208
Weight         0.210370
Touchscreen    0.191226
Inches         0.068197
Name: Price, dtype: float64
```

لذا يمكننا الآن إسقاط الأعمدة الإضافية غير المفيدة.في هذه المرحلة، بدأنافي الاحتفاظ بالأعمدة المهمةفي مجموعة البيانات الخاصة بنا.

```
data.drop(columns = ['ScreenResolution', 'Inches','X_res','Y_res'],
inplace=True)
```

6) عمود وحدة المعالجة المركزية

إذا لاحظت عمود وحدة المعالجة المركزية، فإنه يحتوي أيضًا على الكثير من المعلومات. إذا كنت تستخدم دالة فريدة أو دالة حساب القيمة مرة أخرىفي عمود وحدة المعالجة المركزية،

فلدينا 118 فئة مختلفة. المعلومات التي يقدمها حول المعالجات المسبقةفي أجهزة الكمبيوتر المحمولة والسرعة.

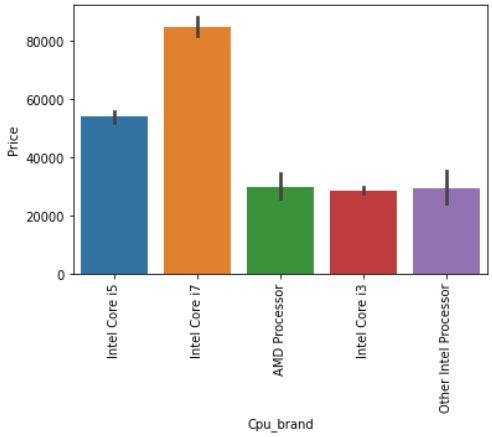
```
#first we will extract Name of CPU which is first 3 words from Cpu
column and then we will check which processor it is
def fetch_processor(x):
  cpu_name = " ".join(x.split()[0:3])
  if cpu_name == 'Intel Core i7' or cpu_name == 'Intel Core i5' or
cpu_name == 'Intel Core i3':
    return cpu_name
  elif cpu_name.split()[0] == 'Intel':
    return 'Other Intel Processor'
  else:
    return 'AMD Processor'
data['Cpu_brand'] = data['Cpu'].apply(lambda x: fetch_processor(x))
```

لاستخراج المعالج المسبق، نحتاج إلى استخراج الكلمات الثلاث الأولى من السلسلة. لدينا معالج Intel ومعالج AMD المسبق لذلك نحتفظ بـ 5 فئاتفي مجموعة البيانات لدينا مثل i3 وi5 وi7 ومعالجات Intel الأخرى ومعالجات AMD.

كيف يختلف السعر باختلاف المعالجات؟

يمكننا مرة أخرى استخدام خاصية المخطط الشريطي bar plot للإجابة على هذا السؤال. ومن الواضح أن سعر معالج i7 مرتفع، ثم معالج i5 ومعالج i3 ومعالج AMD يقعانفي نفس النطاق تقريبًا. ومن ثم سيعتمد السعر على المعالج المسبق.

```
sns.barplot(x=data['Cpu_brand'],y=data['Price'])
plt.xticks(rotation='vertical')
plt.show()
```

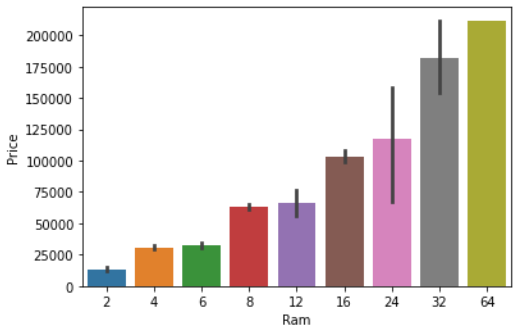


7) السعر مع ذاكرة الوصول العشوائي

مرة أخرى تحليل ثنائي المتغير للسعر مع ذاكرة الوصول العشوائي. إذا لاحظت المخطط، فإن السعر له علاقة إيجابية قوية جدًا بذاكرة الوصول العشوائية أو يمكنك قول علاقة خطية.

```
sns.barplot(data['Ram'], data['Price'])
```

```
plt.show()
```



8) عمود الذاكرة

عمود الذاكرة memory column هو مرة أخرى عمود صاخب يعطي فهمًا لمحركات الأقراص الثابتة. تأتي العديد من أجهزة الكمبيوتر المحمولة مزودة بـ HHD وSSD على حد سواء، كما يوجدفي بعضها فتحة خارجية لإدخالها بعد الشراء. يمكن أن يزعج هذا العمود تحليلك إذا لم يتم تصميمه بشكل صحيح. لذلك إذا كنت تستخدم عدد القيمفي عمود، فنحن لدينا 4 فئات مختلفة من الذاكرة مثل HHD وSSD وتخزين الفلاش والهجين.

```
#preprocessing
data['Memory'] = data['Memory'].astype(str).replace('.0', '',
regex=True)
data["Memory"] = data["Memory"].str.replace('GB', '')
data["Memory"] = data["Memory"].str.replace('TB', '000')
new = data["Memory"].str.split("+", n = 1, expand = True)
data["first"]= new[0]
data["first"]=data["first"].str.strip()
data["second"]= new[1]
data["Layer1HDD"] = data["first"].apply(lambda x: 1 if "HDD" in x else
0)
data["Layer1SSD"] = data["first"].apply(lambda x: 1 if "SSD" in x else
0)
data["Layer1Hybrid"] = data["first"].apply(lambda x: 1 if "Hybrid" in
x else 0)
data["Layer1Flash_Storage"] = data["first"].apply(lambda x: 1 if
"Flash Storage" in x else 0)
data['first'] = data['first'].str.replace(r'D', '')
data["second"].fillna("0", inplace = True)
data["Layer2HDD"] = data["second"].apply(lambda x: 1 if "HDD" in x
else 0)
data["Layer2SSD"] = data["second"].apply(lambda x: 1 if "SSD" in x
else 0)
data["Layer2Hybrid"] = data["second"].apply(lambda x: 1 if "Hybrid" in
x else 0)
data["Layer2Flash_Storage"] = data["second"].apply(lambda x: 1 if
"Flash Storage" in x else 0)
data['second'] = data['second'].str.replace(r'D', '')
#binary encoding
data["Layer2HDD"] = data["second"].apply(lambda x: 1 if "HDD" in x
else 0)
data["Layer2SSD"] = data["second"].apply(lambda x: 1 if "SSD" in x
else 0)
data["Layer2Hybrid"] = data["second"].apply(lambda x: 1 if "Hybrid" in
x else 0)
```

```
data["Layer2Flash_Storage"] = data["second"].apply(lambda x: 1 if
"Flash Storage" in x else 0)
#only keep integert(digits)
data['second'] = data['second'].str.replace(r'D', '')
#convert to numeric
data["first"] = data["first"].astype(int)
data["second"] = data["second"].astype(int)
#finalize the columns by keeping value
data["HDD"]=(data["first"]*data["Layer1HDD"]+data["second"]*data["Laye
r2HDD"])
data["SSD"]=(data["first"]*data["Layer1SSD"]+data["second"]*data["Laye
r2SSD"])
data["Hybrid"]=(data["first"]*data["Layer1Hybrid"]+data["second"]*data
["Layer2Hybrid"])
data["Flash_Storage"]=(data["first"]*data["Layer1Flash_Storage"]+data[
"second"]*data["Layer2Flash_Storage"])
#Drop the un required columns
data.drop(columns=['first', 'second', 'Layer1HDD', 'Layer1SSD',
'Layer1Hybrid',
       'Layer1Flash_Storage', 'Layer2HDD', 'Layer2SSD',
'Layer2Hybrid',
       'Layer2Flash_Storage'],inplace=True)
```

أولاً، قمنا بتنظيف عمود الذاكرة ثم صنعنا 4 أعمدة جديدة وهي عبارة عن عمود ثنائي حيث يحتوي كل عمود على 1 و 0 يشير إلى أن المبلغ أربعة موجود وغير موجود. يحتوي أي كمبيوتر محمول على نوع واحد من الذاكرة أو مزيج من نوعين. لذلكفي العمود الأول، يتكون من حجم الذاكرة الأول وإذا كانت الفتحة الثانية موجودةفي الكمبيوتر المحمول، فسيحتوي العمود الثاني عليها، وإلا فإننا نملأ القيم الخالية بصفر. بعد ذلكفي عمود معين، قمنا بضرب القيم بقيمتها الثنائية. هذا يعني أنهفي حالة وجود ذاكرة معينةفي أي كمبيوتر محمول، فإنها تحتوي على قيمة ثنائية كقيمة واحدة وسيتم ضرب القيمة الأولى بها، ونفس الشيء مع المجموعة الثانية. بالنسبة للكمبيوتر المحمول الذي يحتوي على فتحة ثانية، ستكون القيمة صفرًا مضروبةفي صفر تساوي صفرًا.

الآن عندما نرى ارتباط السعر، يكون للتخزين الهجين والتخزين الفلاش ارتباط أقل أو معدومًا مع السعر. سنقوم بإسقاط هذا العمود مع وحدة المعالجة المركزية والذاكرة التي لم تعد مطلوبة.

```
data.drop(columns=['Hybrid','Flash_Storage','Memory','Cpu'],inplace=Tr
ue)
```

9) متغير GPU

تحتوي وحدة المعالجة الرسومية (GPU) على العديد من الفئاتفي البيانات. لدينا بطاقة رسومية ذات علامة تجارية موجودة على جهاز كمبيوتر محمول. ليس لدينا عدد سعات مثل بطاقة الرسوم (6 جيجا بايت، 12 جيجا بايت) الموجودة. لذلك سنقوم ببساطة باستخراج اسم العلامة التجارية.

```
# Which brand GPU is in laptop
data['Gpu_brand'] = data['Gpu'].apply(lambda x:x.split()[0])
#there is only 1 row of ARM GPU so remove it
data = data[data['Gpu_brand'] != 'ARM']
data.drop(columns=['Gpu'],inplace=True)
```

إذا كنت تستخدم دالة عدد القيم، فهناك صف مع GPU لـ ARM لذلك قمنا بإزالة هذا الصف وبعد استخراج عمود GPU للعلامة التجارية لم تعد هناك حاجة.

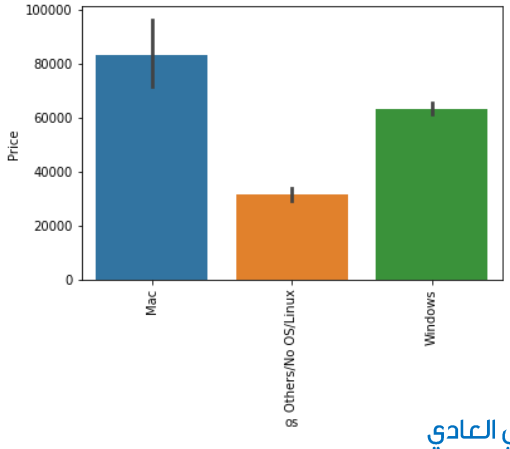
10) عمود نظام التشغيل

هناك العديد من فئات أنظمة التشغيل. سنحتفظ بجميع فئات windowsفي واحدة، وMacفي واحد، والباقيفي فئات أخرى. هذه طريقة هندسة ميزات بسيطة وأكثرها استخدامًا، يمكنك تجربة شيء آخر إذا وجدت ارتباطًا أكثر بالسعر.

```
#Get which OP sys
def cat_os(inp):
    if inp == 'Windows 10' or inp == 'Windows 7' or inp == 'Windows 10
S':
        return 'Windows'
    elif inp == 'macOS' or inp == 'Mac OS X':
        return 'Mac'
    else:
        return 'Others/No OS/Linux'
data['os'] = data['OpSys'].apply(cat_os)
data.drop(columns=['OpSys'],inplace=True)
```

عندما تقوم برسم السعر مقابل نظام التشغيل، فإن Mac كالمعتاد يكون أغلى سعرًا.

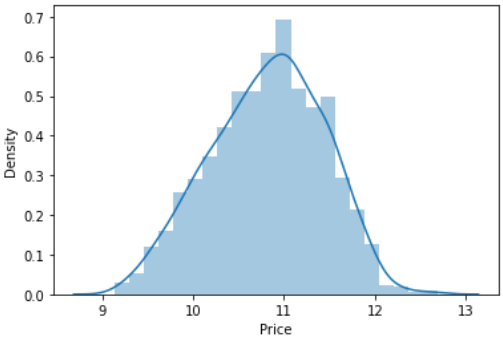
```
sns.barplot(x=data['os'],y=data['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



التحول اللوغاريتمي العادي

لقد رأينا توزيع المتغير الهدف الذي فوقه كان منحرفًا لليمين. بتحويله إلى التوزيع الطبيعي سيزداد أداء الخوارزمية. نأخذ سجل القيم التي تتحول إلى التوزيع الطبيعي الذي يمكنك ملاحظته أدناه. لذلك أثناء فصل المتغيرات التابعة والمستقلة، سنأخذ سجلًا للسعر، وفي عرض النتيجة نؤدي أسًا لها.

```
sns.distplot(np.log(data['Price']))
plt.show()
```



نمذجة التعلم الآلي لتنبؤ أسعار أجهزة الكمبيوتر المحمول

الآن قمنا بإعداد بياناتنا وفهم مجموعة البيانات بشكل أفضل. فلنبدأ بنمذجة التعلم الآلي ونعثر على أفضل خوارزمية مع أفضل المعلمات الفائقة لتحقيق أقصى قدر من الدقة.

استيراد المكتبات

```
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score,mean_absolute_error
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import
RandomForestRegressor,GradientBoostingRegressor,AdaBoostRegressor,Extr
aTreesRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
```

لقد قمنا باستيراد مكتبات لتقسيم البيانات، وخوارزميات يمكنك تجربتها.في وقت لا نعرف أيهما هو الأفضل حتى تتمكن من تجربة جميع الخوارزميات المستوردة.

التقسيم الى بيانات التدريب والاختبار

كما ناقشنا، أخذنا سجل المتغيرات التابعة. وتبدو بيانات التدريب شيئًا ما أسفل إطار البيانات.

```
X = data.drop(columns=['Price'])

y = np.log(data['Price'])

X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.15,random_state=2)
```

```
X.head()
```

	Company	TypeName	Ram	Weight	Touchscreen	Ips	ppi	Cpu_brand	HDD	SSD	Gpu_brand	os
0	Apple	Ultrabook	8	1.37	0	1	226.983005	Intel Core i5	0	128	Intel	Mac
1	Apple	Ultrabook	8	1.34	0	0	127.677940	Intel Core i5	0	0	Intel	Mac
2	HP	Notebook	8	1.86	0	0	141.211998	Intel Core i5	0	256	Intel	Others/No OS/Linux
3	Apple	Ultrabook	16	1.83	0	1	220.534624	Intel Core i7	0	512	AMD	Mac
4	Apple	Ultrabook	8	1.37	0	1	226.983005	Intel Core i5	0	256	Intel	Mac

تنفيذ خط أنابيب للتدريب والاختبار

الآن سنقوم بتنفيذ خط أنابيب لتبسيط عملية التدريب والاختبار. أولاً، نستخدم محول عمود ColumnTransformer لترميز المتغيرات الفئوية وهي الخطوة الأولى. بعد ذلك، نقوم بإنشاء كائن من خوارزمية لدينا ونمرر كلا الخطوتين إلى خط الأنابيب. باستخدام كائنات خط الأنابيب، نتوقع النتيجة على البيانات الجديدة ونعرض الدقة.

```
step1 = ColumnTransformer(transformers=[
('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')
step2 = RandomForestRegressor(n_estimators=100,
random_state=3,
max_samples=0.5,
max_features=0.75,
max_depth=15)
pipe = Pipeline([
('step1',step1),
('step2',step2)
])
pipe.fit(X_train,y_train)
y_pred = pipe.predict(X_test)
print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))
```

في الخطوة الأولى للترميز الفئوي، مررنا فهرس الأعمدة للترميز، وتمرر الوسائل التمريرية المفضلة Random Forestالأعمدة الرقمية الأخرى كما هي. أفضل دقة حصلت عليها مع على الإطلاق. لكن يمكنك استخدام هذا الرمز مرة أخرى عن طريق تغيير الخوارزمية . يمكنك القيام بضبط المعلمات الفائقة باستخدام Random Forestومعلماتها. أنا أعرض . يمكننا أيضًا القيام بميزة التحجيم ولكنها لا Random Search CV أو GridsearchCV تحدث أي تأثير على Random Forest.

3) تحليل المشاعر على تويتر باستخدام التعلم الالي Twitter Sentiment Analysis using Machine Learning

Twitter هو أحد منصات التواصل الاجتماعي حيث يتمتع الأشخاص بحرية مشاركة آرائهم حول أي موضوع. نرى أحيانًا مناقشة قوية على Twitter حول رأي شخص ما تؤدي أحيانًا إلى مجموعة من التغريدات السلبية. مع وضع ذلكفي الاعتبار، إذاكنت تريد معرفة كيفية إجراء تحليل المشاعر على Twitter، فهذه المقالة مناسبة لك.في هذه المقالة، سوف أطلعك على مهمة تحليل المشاعر على Twitter باستخدام Python.

تحليل المشاعر على تويتر

تحليل المشاعر Sentiment analysis مهمة معالجة اللغة الطبيعية. يجب على جميع منصات وسائل التواصل الاجتماعي مراقبة مشاعر المشاركينفي المناقشة. نرىفي الغالب آراء سلبية على تويتر عندما تكون المناقشة سياسية. لذلك، يجب أن تستمر كل منصةفي تحليل المشاعر للعثور على نوع الأشخاص الذين ينشرون الكراهية والسلبية على نظامهم الأساسي.

بالنسبة لمهمة تحليل المشاعر على Twitter، قمت بجمع مجموعة بيانات من Kaggle تحتوي على تغريدات حول مناقشة طويلة داخل مجموعة من المستخدمين. مهمتنا هنا هي تحديد عدد التغريدات السلبية والإيجابية حتى نتمكن من إعطاء نتيجة. لذلك،في القسم أدناه، سأقدم لك مهمة تحليل المشاعر على Twitter باستخدام Python.

تحليل المشاعر على Twitter باستخدام Python

لنبدأ مهمة تحليل المشاعر على Twitter من خلال استيراد مكتبات Python ومجموعة البيانات اللازمة:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import re
import nltk
import nltk

data =
pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-
data/master/twitter.csv")
print(data.head())
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class  \
0           0      3            0                   0        3      2
1           1      3            0                   3        0      1
2           2      3            0                   3        0      1
3           3      3            0                   2        1      1
4           4      6            0                   6        0      1

                                               tweet
0  !!! RT @mayasolovely: As a woman you shouldn't...
1  !!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2  !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3  !!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4  !!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
```

يحتوي عمود التغريدات tweet column في مجموعة البيانات أعلاه على التغريدات التي نحتاج إلى استخدامها لتحليل مشاعر المشاركين في المناقشة. ولكن للمضي قدمًا، يتعين علينا تنظيف الكثير من الأخطاء والرموز الخاصة الأخرى لأن هذه التغريدات تحتوي على الكثير من الأخطاء اللغوية. إذن إليك كيف يمكننا تنظيف عمود التغريدة:

```
nltk.download('stopwords')
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english'))

def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["tweet"] = data["tweet"].apply(clean)
```

الآن، الخطوة التالية هي حساب درجات المشاعر لهذه التغريدات وتعيين تسمية للتغريدات على أنها إيجابية positive أو سلبية negative أو محايدة neutral. إليك كيفية حساب درجات المشاعر في التغريدات:

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
sentiments = SentimentIntensityAnalyzer()
data["Positive"] = [sentiments.polarity_scores(i)["pos"] for i in
data["tweet"]]
data["Negative"] = [sentiments.polarity scores(i)["neg"] for i in
data["tweet"]]
data["Neutral"] = [sentiments.polarity_scores(i)["neu"] for i in
data["tweet"]]
```

الآن سأختار فقط الأعمدة من هذه البيانات التي نحتاجها لبقية مهمة تحليل المشاعر على Twitter:

```
data = data[["tweet", "Positive",
             "Negative", "Neutral"]]
```

```
print(data.head())
```

```
                                               tweet  Positive  Negative  \
0   rt mayasolov woman shouldnt complain clean ho...     0.147     0.157
1   rt  boy dat coldtyga dwn bad cuffin dat hoe  ...     0.000     0.280
2   rt urkindofbrand dawg rt  ever fuck bitch sta...     0.000     0.577
3            rt cganderson vivabas look like tranni     0.333     0.000
4   rt shenikarobert shit hear might true might f...     0.154     0.407

   Neutral
0    0.696
1    0.720
2    0.423
3    0.667
4    0.440
```

دعنا الآن نلقي نظرة على التصنيف الأكثر شيوعًا المخصص للتغريدات وفقًا لدرجات المشاعر:

```
x = sum(data["Positive"])
y = sum(data["Negative"])
```

```
z = sum(data["Neutral"])
```

```
def sentiment_score(a, b, c):
```

```
    if (a>b) and (a>c):
```

```
        print("Positive 😊 ")
```

```
    elif (b>a) and (b>c):
```

```
        print("Negative 😠 ")
```

```
    else:
```

```
        print("Neutral 😐 ")
```

```
sentiment_score(x, y, z)
```

Neutral 😐

لذا فإن معظم التغريدات محايدة، ما يعني أنها ليست إيجابية ولا سلبية. الآن دعنا نلقي نظرة على إجمالي درجات المشاعر:

```
print("Positive: ", x)
print("Negative: ", y)
print("Neutral: ", z)
```

Positive: 2880.086000000009
Negative: 7201.020999999922
Neutral: 14696.887999999733

مجموع التغريدات المحايدة أعلى بكثير من السلبية والإيجابية، لكن من بين جميع التغريدات السلبية أكبر من التغريدات الإيجابية، لذلك يمكننا القول إن معظم الآراء سلبية.

الملخص

هذه هي الطريقة التي يمكنك بها أداء مهمة تحليل المشاعر على Twitter باستخدام لغة برمجة Python. تحليل المشاعر مهمة معالجة اللغة الطبيعية. تحتاج جميع منصات وسائل التواصل الاجتماعي إلى التحقق من مشاعر الأشخاص المشاركينفي المناقشة. آمل أن تكون قد أحببت هذا المقال على تحليل المشاعر على Twitter باستخدام Python.

4) توقع أسعار السيارة مع التعلم الآلي Car Price Prediction with Machine Learning

يعتمد سعر السيارة على الكثير من العوامل مثل السمعة الحسنة للعلامة التجارية للسيارة وميزات السيارة والقدرة الحصانية والمسافة المقطوعة التي تعطيها وغيرها الكثير. يعد التنبؤ بسعر السيارة أحد مجالات البحث الرئيسيةفي التعلم الآلي. لذلك إذا كنت تريد معرفة كيفية تدريب نموذج التنبؤ بسعر السيارة، فهذه المقالة مناسبة لك.في هذه المقالة، سوف آخذكفي جولة حول كيفية تدريب نموذج التنبؤ بسعر السيارة باستخدام التعلم الآلي باستخدام Python.

توقع أسعار السيارة مع التعلم الآلي

أحد المجالات الرئيسية للبحثفي التعلم الآلي هو التنبؤ بسعر السيارات. يعتمد على التمويل ومجال التسويق. إنه موضوع بحث رئيسيفي التعلم الآلي لأن سعر السيارة يعتمد على العديد من العوامل. بعض العوامل التي تساهم كثيرًافي سعر السيارة هي:

1. الماركة Brand.
2. النموذج Model.
3. القوة الحصانية Horsepower.
4. عدد الأميال Mileage.
5. ميزات السلامة Safety Features.
6. GPS وغيرها الكثير.

إذا تجاهل المرء العلامة التجارية للسيارة، فإن الشركة المصنعة للسيارة تحدد سعر السيارة بشكل أساسي بناءً على الميزات التي يمكن أن تقدمها للعميل. لاحقًا، قد ترفع العلامة التجارية السعر اعتمادًا على حسن نيتها، ولكن أهم العوامل هي الميزات التي تمنحها السيارة لإضافة قيمة إلى حياتك. لذلك،في القسم أدناه، سوف أطلعك على مهمة تدريب نموذج التنبؤ بسعر السيارة باستخدام التعلم الآلي باستخدام لغة برمجة Python.

نموذج التنبؤ بسعر السيارة باستخدام لغة بايثون

تم تنزيل مجموعة البيانات التي أستخدمها هنا لتدريب نموذج توقع أسعار السيارة من Kaggle. يحتوي على بيانات حول جميع الميزات الرئيسية التي تساهمفي سعر السيارة. فلنبدأ هذه المهمة عن طريق استيراد مكتبات Python ومجموعة البيانات الضرورية:

مجموعة البيانات

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
```

```
data = pd.read_csv("CarPrice.csv")
data.head()
```

	car_ID	symboling	CarName	...	citympg	highwaympg	price
0	1	3	alfa-romero giulia	...	21	27	13495.0
1	2	3	alfa-romero stelvio	...	21	27	16500.0
2	3	1	alfa-romero Quadrifoglio	...	19	26	16500.0
3	4	2	audi 100 ls	...	24	30	13950.0
4	5	2	audi 100ls	...	18	22	17450.0

[5 rows x 26 columns]

يوجد 26 عمودًافي مجموعة البيانات هذه، لذلك من المهم جدًا التحقق مما إذا كانت مجموعة البيانات هذه تحتوي على قيم فارغة null values أم لا قبل المضي قدمًا:

```
data.isnull().sum()
```

```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

لذلك لا تحتوي مجموعة البيانات هذه على أي قيم فارغة، فلنلقِ الآن نظرة على بعض الأفكار المهمة الأخرى للحصول على فكرة عن نوع البيانات التي نتعامل معها:

```
data.info()
```

```
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

```
print(data.describe())
```

```
           car_ID   symboling   wheelbase  ...     citympg  highwaympg         price
count  205.000000  205.000000  205.000000  ...  205.000000  205.000000    205.000000
mean   103.000000    0.834146   98.756585  ...   25.219512   30.751220  13276.710571
std     59.322565    1.245307    6.021776  ...    6.542142    6.886443   7988.852332
min      1.000000   -2.000000   86.600000  ...   13.000000   16.000000   5118.000000
25%     52.000000    0.000000   94.500000  ...   19.000000   25.000000   7788.000000
50%    103.000000    1.000000   97.000000  ...   24.000000   30.000000  10295.000000
75%    154.000000    2.000000  102.400000  ...   30.000000   34.000000  16503.000000
max    205.000000    3.000000  120.900000  ...   49.000000   54.000000  45400.000000

[8 rows x 16 columns]
```

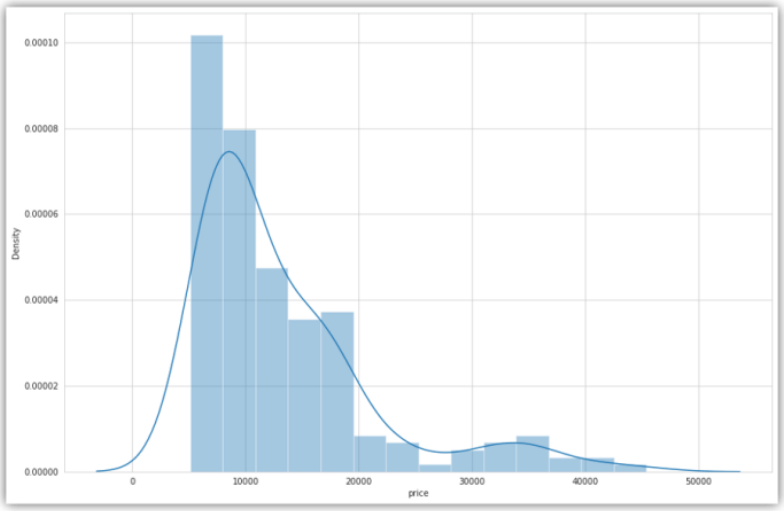
```
data.CarName.unique()
```

```
array(['alfa-romero giulia', 'alfa-romero stelvio',
       'alfa-romero Quadrifoglio', 'audi 100 ls', 'audi 100ls',
       'audi fox', 'audi 5000', 'audi 4000', 'audi 5000s (diesel)',
       'bmw 320i', 'bmw x1', 'bmw x3', 'bmw z4', 'bmw x4', 'bmw x5',
       'chevrolet impala', 'chevrolet monte carlo', 'chevrolet vega 2300',
```

```
'dodge rampage', 'dodge challenger se', 'dodge d200',
'dodge monaco (sw)', 'dodge colt hardtop', 'dodge colt (sw)',
'dodge coronet custom', 'dodge dart custom',
'dodge coronet custom (sw)', 'honda civic', 'honda civic cvcc',
'honda accord cvcc', 'honda accord lx', 'honda civic 1500 gl',
'honda accord', 'honda civic 1300', 'honda prelude',
'honda civic (auto)', 'isuzu MU-X', 'isuzu D-Max ',
'isuzu D-Max V-Cross', 'jaguar xj', 'jaguar xf', 'jaguar xk',
'maxda rx3', 'maxda glc deluxe', 'mazda rx2 coupe', 'mazda rx-4',
'mazda glc deluxe', 'mazda 626', 'mazda glc', 'mazda rx-7 gs',
'mazda glc 4', 'mazda glc custom l', 'mazda glc custom',
'buick electra 225 custom', 'buick century luxus (sw)',
'buick century', 'buick skyhawk', 'buick opel isuzu deluxe',
'buick skylark', 'buick century special',
'buick regal sport coupe (turbo)', 'mercury cougar',
'mitsubishi mirage', 'mitsubishi lancer', 'mitsubishi outlander',
'mitsubishi g4', 'mitsubishi mirage g4', 'mitsubishi montero',
'mitsubishi pajero', 'Nissan versa', 'nissan gt-r', 'nissan rogue',
'nissan latio', 'nissan titan', 'nissan leaf', 'nissan juke',
'nissan note', 'nissan clipper', 'nissan nv200', 'nissan dayz',
'nissan fuga', 'nissan otti', 'nissan teana', 'nissan kicks',
'peugeot 504', 'peugeot 304', 'peugeot 504 (sw)', 'peugeot 604sl',
'peugeot 505s turbo diesel', 'plymouth fury iii',
'plymouth cricket', 'plymouth satellite custom (sw)',
'plymouth fury gran sedan', 'plymouth valiant', 'plymouth duster',
'porsche macan', 'porcshce panamera', 'porsche cayenne',
'porsche boxter', 'renault 12tl', 'renault 5 gtl', 'saab 99e',
'saab 99le', 'saab 99gle', 'subaru', 'subaru dl', 'subaru brz',
'subaru baja', 'subaru r1', 'subaru r2', 'subaru trezia',
'subaru tribeca', 'toyota corona mark ii', 'toyota corona',
'toyota corolla 1200', 'toyota corona hardtop',
'toyota celica gt liftback', 'toyota corolla tercel',
'toyota corona liftback', 'toyota starlet', 'toyota tercel',
'toyota cressida', 'toyota celica gt', 'toyouta tercel',
'vokswagen rabbit', 'volkswagen 1131 deluxe sedan',
'volkswagen model 111', 'volkswagen type 3', 'volkswagen 411 (sw)',
'volkswagen super beetle', 'volkswagen dasher', 'vw dasher',
'vw rabbit', 'volkswagen rabbit', 'volkswagen rabbit custom',
'volvo 145e (sw)', 'volvo 144ea', 'volvo 244dl', 'volvo 245',
'volvo 264gl', 'volvo diesel', 'volvo 246'], dtype=object)
```

من المفترض أن يكون عمود السعرفي مجموعة البيانات هذه هو العمود الذي نحتاج إلى توقع قيمه. دعونا نرى توزيع قيم عمود السعر:

```
sns.set style("whitegrid")
plt.figure(figsize=(15, 10))
sns.distplot(data.price)
plt.show()
```



دعنا الآن نلقي نظرة على الارتباط بين جميع ميزات مجموعة البيانات هذه:

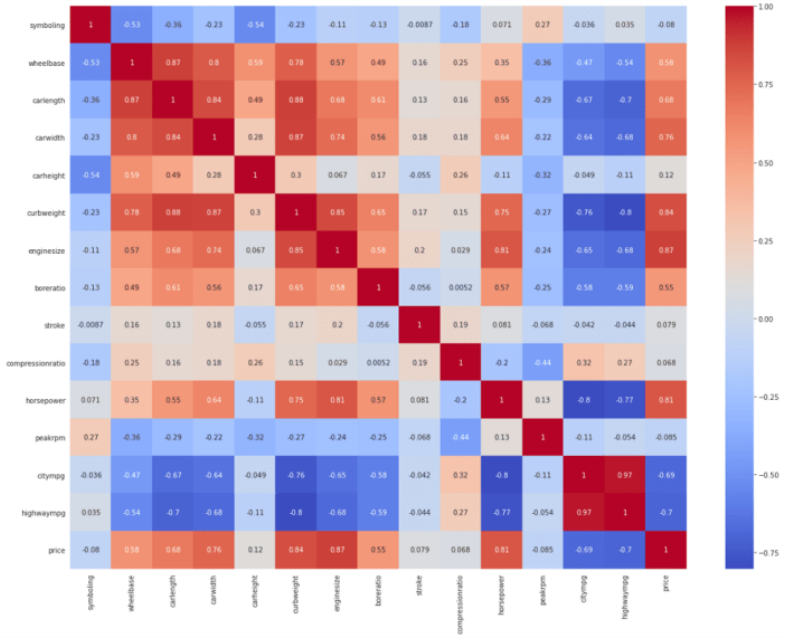
	car_ID	symboling	...	highwaympg	price
car_ID	1.000000	-0.151621	...	0.011255	-0.109093
symboling	-0.151621	1.000000	...	0.034606	-0.079978
wheelbase	0.129729	-0.531954	...	-0.544082	0.577816
carlength	0.170636	-0.357612	...	-0.704662	0.682920
carwidth	0.052387	-0.232919	...	-0.677218	0.759325
carheight	0.255960	-0.541038	...	-0.107358	0.119336
curbweight	0.071962	-0.227691	...	-0.797465	0.835305
enginesize	-0.033930	-0.105790	...	-0.677470	0.874145
boreratio	0.260064	-0.130051	...	-0.587012	0.553173
stroke	-0.160824	-0.008735	...	-0.043931	0.079443
compressionratio	0.150276	-0.178515	...	0.265201	0.067984
horsepower	-0.015006	0.070873	...	-0.770544	0.808139
peakrpm	-0.203789	0.273606	...	-0.054275	-0.085267
citympg	0.015940	-0.035823	...	0.971337	-0.685751
highwaympg	0.011255	0.034606	...	1.000000	-0.697599
price	-0.109093	-0.079978	...	-0.697599	1.000000

[16 rows x 16 columns]

```
plt.figure(figsize=(20, 15))
correlations = data.corr()
```

```
sns.heatmap(correlations, cmap="coolwarm", annot=True)
```

```
plt.show()
```



تدريب نموذج التنبؤ بسعر السيارة

سأستخدم خوارزمية انحدار شجرة القرار decision tree لتدريب نموذج التنبؤ بسعر السيارة. لذلك دعونا نقسم البيانات إلى مجموعات تدريب واختبار ونستخدم خوارزمية انحدار شجرة القرار لتدريب النموذج:

```
predict = "price"
data = data[["symboling", "wheelbase", "carlength",
             "carwidth", "carheight", "curbweight",
             "enginesize", "boreratio", "stroke",
             "compressionratio", "horsepower", "peakrpm",
             "citympg", "highwaympg", "price"]]
x = np.array(data.drop([predict], 1))
y = np.array(data[predict])

from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2)

from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
model.fit(xtrain, ytrain)
predictions = model.predict(xtest)

from sklearn.metrics import mean_absolute_error
model.score(xtest, predictions)
```

```
1.0
```

يعطي النموذج دقة 100٪في مجموعة الاختبار، وهو أمر ممتاز.

الملخص

هذه هي الطريقة التي يمكنك بها تدريب نموذج التعلم الآلي لمهمة التنبؤ بأسعار السيارات باستخدام لغة برمجة Python. إنه موضوع بحث رئيسيفي التعلم الآلي لأن سعر السيارة يعتمد على العديد من العوامل. أتمنى أن تكون قد أحببت هذا المقال حول مهمة تدريب نموذج للتنبؤ بأسعار السيارات باستخدام التعلم الآلي.

.

5) اكتشاف البريد العشوائي باستخدام التعلم الآلي Spam Detection using Machine Learning

يعد اكتشاف التنبيهات غير المرغوب فيهافي رسائل البريد الإلكتروني والرسائل أحد التطبيقات الرئيسية التي تحاول كل شركة تقنية كبيرة تحسينها لعملائها. يعد تطبيق المراسلة الرسمي من Apple وGmail من Google مثالين رائعين لمثل هذه التطبيقات حيث يعمل اكتشاف الرسائل غير المرغوب فيها بشكل جيد لحماية المستخدمين من تنبيهات الرسائل غير المرغوب فيها. لذلك، إذا كنت تتطلع إلى إنشاء نظام للكشف عن الرسائل غير المرغوب فيها، فهذه المقالة مناسبة لك.في هذه المقالة، سوف أطلعك على مهمة اكتشاف البريد العشوائي باستخدام التعلم الآلي باستخدام Python.

اكتشاف البريد العشوائي

عندما ترسل تفاصيل حول بريدك الإلكتروني أو رقم الاتصال الخاص بك على أي منصة، أصبح من السهل على تلك المنصات تسويق منتجاتها عن طريق الإعلان عنها عن طريق إرسال رسائل بريد إلكتروني أو عن طريق إرسال رسائل مباشرة إلى رقم الاتصال الخاص بك. ينتج عن هذا الكثير من التنبيهات والإشعارات بشأن البريد العشوائيفي صندوق الوارد الخاص بك. هذا هو المكان الذي تأتي فيه مهمة اكتشاف البريد العشوائي.

يعني Spam detection اكتشاف الرسائل غير المرغوب فيها أو رسائل البريد الإلكتروني من خلال فهم محتوى النص بحيث يمكنك فقط تلقي إشعارات حول الرسائل أو رسائل البريد الإلكتروني المهمة جدًا بالنسبة لك. إذا تم العثور على رسائل غير مرغوب فيها، فسيتم نقلها تلقائيًا إلى مجلد البريد العشوائي ولن يتم إخطارك بهذه التنبيهات. يساعد هذافي تحسين تجربة المستخدم، حيث يمكن للعديد من تنبيهات البريد العشوائي أن تزعج العديد من المستخدمين.

اكتشاف البريد العشوائي باستخدام بايثون

آمل أن تفهم الآن ما هو اكتشاف الرسائل غير المرغوب فيها، فلنرى الآن كيفية تدريب نموذج التعلم الآلي على اكتشاف تنبيهات الرسائل غير المرغوب فيها باستخدام Python. سأبدأ هذه المهمة عن طريق استيراد مكتبات Python الضرورية ومجموعة البيانات التي تحتاجها لهذه المهمة:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
data = pd.read_csv("https://raw.githubusercontent.com/amankharwal/SMS-
Spam-Detection/master/spam.csv", encoding= 'latin-1')
data.head()
```

	class	message	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

من مجموعة البيانات هذه، الفئة class والرسالة message هما الميزتان الوحيدتان اللتان نحتاجهما لتدريب نموذج التعلم الآلي على اكتشاف الرسائل غير المرغوب فيها، لذلك دعونا نحدد هذين العمودين كمجموعة بيانات جديدة:

```
data = data[["class", "message"]]
```

دعنا الآن نقسم مجموعة البيانات هذه إلى مجموعات تدريب واختبار وتدريب النموذج على اكتشاف الرسائل غير المرغوب فيها:

```
x = np.array(data["message"])
y = np.array(data["class"])
cv = CountVectorizer()
X = cv.fit_transform(x) # Fit the Data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

clf = MultinomialNB()
clf.fit(X_train,y_train)
```

دعنا الآن نختبر هذا النموذج من خلال أخذ إدخال المستخدم كرسالة لاكتشاف ما إذا كان بريداً عشوائيًا أم لا:

```
sample = input('Enter a message:')
data = cv.transform([sample]).toarray()
print(clf.predict(data))
```

```
Enter a message:You won $40 cash price
['spam']
```

الملخص

هذه هي الطريقة التي يمكنك بها تدريب نموذج التعلم الآلي على مهمة اكتشاف ما إذا كانت رسالة بريد إلكتروني أو رسالة بريد عشوائي أم لا. يكتشف كاشف الرسائل غير المرغوب فيها رسائل البريد العشوائي أو رسائل البريد الإلكتروني من خلال فهم محتوى النص بحيث يمكنك فقط تلقي إشعارات حول الرسائل أو رسائل البريد الإلكتروني المهمة جداً بالنسبة لك. آمل أن تكون قد أحببت هذه المقالة حول مهمة الكشف عن تنبيهات البريد العشوائي باستخدام التعلم الآلي باستخدام Python.

6) اكتشاف سرطان الثدي من خلال التعلم الآلي Breast Cancer Detection with Machine Learning

على مدى العقود الماضية، تم استخدام تقنيات التعلم الآلي على نطاق واسعفي الأنظمة الصحية الذكية، لا سيمافي تشخيص سرطان الثدي والتنبؤ به.في هذه المقالة، سوف أطلعك على كيفية إنشاء نموذج للكشف عن سرطان الثدي باستخدام التعلم الآلي ولغة برمجة Python.

يعد سرطان الثدي Breast Cancer من أكثر السرطانات شيوعًا بين النساء على مستوى العالم، حيث يمثل غالبية حالات السرطان الجديدة والوفيات المرتبطة بالسرطان وفقًا للإحصاءات العالمية، مما يجعله مشكلة صحية عامة رئيسيةفي العالم. مجتمع اليوم.

يمكن أن يؤدي التشخيص المبكر لسرطان الثدي إلى تحسين التشخيص وفرص البقاء على قيد الحياة بشكل كبير، حيث يمكن أن يعزز العلاج السريري للمرضىفي الوقت المناسب. التصنيف الأكثر دقة للأورام الحميدة يمكن أن يمنع المرضى من الخضوع لعلاجات غير ضرورية.

الكشف عن سرطان الثدي باستخدام التعلم الآلي

في هذا القسم، سأقوم بتطبيق خوارزمية Naive Bayesفي التعلم الآلي باستخدام Python. لهذه المهمة، سأستخدم قاعدة بيانات لمعلومات أورام سرطان الثدي للكشف عن سرطان الثدي.

لنبدأ باستيراد وتحميل مكتبات Python الضرورية ومجموعة بيانات سرطان الثدي المقدمة من Scikit-Learn:

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
```

نحتاج الآن إلى إنشاء متغيرات جديدة لكل مجموعة مهمة من المعلومات التي نجدها مفيدة وتعيين السماتفي مجموعة البيانات لتلك المتغيرات:

```
data = load breast cancer()
label_names = data["target_names"]
labels = data["target"]
feature_names = data["feature_names"]
features = data["data"]
```

لدينا الآن قيم لكل مجموعة من المعلومات المفيدةفي مجموعة البيانات. لفهم مجموعة البيانات الخاصة بنا بشكل أفضل، دعنا نلقي نظرة على بياناتنا عن طريق طباعة تسميات الفئة لدينا،

والتسمية الخاصة بمثيل البيانات الأول، وأسماء الكيانات الخاصة بنا، وقيم الكيان لمثيل البيانات الأول:

```
print(label_names)
print("Class label :", labels[0])
print(feature_names)
print(features[0], "\n")
```

```
['malignant' 'benign']
Class label : 0
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
```

الآن بعد أن تم تحميل بياناتنا، يمكننا العمل مع بياناتنا لبناء نموذج التعلم الآلي الخاص بنا باستخدام خوارزمية Naive Bayes لمهمة اكتشاف سرطان الثدي.

تقسيم مجموعة البيانات

لتقييم أداء المصنف، يجب عليك دائمًا اختبار النموذج على بيانات غير مرئية. لذلك، قبل إنشاء نموذج للتعلم الآلي للكشف عن سرطان الثدي، سأقسم بياناتك إلى قسمين: مجموعة تدريب بنسبة 80٪ ومجموعة اختبار بنسبة 20٪:

```
train, test, train_labels, test_labels = train_test_split(features,
labels,test_size=0.2,random_state=42)
```

استخدام Naive Bayes للكشف عن سرطان الثدي

هناك العديد من نماذج التعلم الآلي، ولكل نموذج نقاط قوته وضعفه. بالنسبة لمهمة نموذج الكشف عن سرطان الثدي، سأركز على خوارزمية بسيطة تعمل بشكل جيد بشكل عامفي مهام التصنيف الثنائي، وهي مصنف Naive Bayes:

```
gnb = GaussianNB()
gnb.fit(train, train_labels)
```

بعد تدريب النموذج، يمكننا بعد ذلك استخدام النموذج المدرب لعمل تنبؤات على مجموعة الاختبار الخاصة بنا، والتي نستخدمها دالة التنبؤ ().

ترجع الدالة التنبؤ () مجموعة من التنبؤات لكل مثيل بياناتفي مجموعة الاختبار. يمكننا بعد ذلك طباعة تنبؤاتنا للتعرف على ما حدده النموذج:

```
preds = gnb.predict(test)
print(preds, "\n")
```

```
[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
 1 1 0]
```

باستخدام مجموعة تصنيفات الفئات الحقيقية، يمكننا تقييم دقة تنبؤات نموذجنا من خلال مقارنة المصفوفتين (test_labels مقابل preds).

سأستخدم دالة () accuracy_score التي يوفرها Scikit-Learn لتحديد معدل الدقة لمصنف التعلم الآلي الخاص بنا:

```
print(accuracy_score(test_labels, preds))
```

```
0.9736842105263158
```

كما ترون من المخرجات أعلاه، فإن نموذج الكشف عن سرطان الثدي لدينا يعطي معدل دقة يقارب 97٪. هذا يعني أن 97٪ من الوقت يكون المصنف قادرًا على إجراء التنبؤ الصحيح.

الملخص

هذه هي الطريقة التي يمكننا بها بناء نموذج للكشف عن سرطان الثدي باستخدام التعلم الآلي ولغة برمجة بايثون. أتمنى أن تكون قد أحببت هذه المقالة حول كيفية بناء نموذج للكشف عن سرطان الثدي باستخدام التعلم الآلي.

7) التنبؤ بأمراض القلب باستخدام التعلم الآلي Heart Disease Prediction using Machine Learning

في هذه المقالة، سوف آخذكفي جولة حول كيفية تدريب نموذج لمهمة التنبؤ بأمراض القلب باستخدام التعلم الآلي. سأستخدم خوارزمية الانحدار اللوجستي Logistic Regressionفي التعلم الآلي لتدريب نموذج للتنبؤ بأمراض القلب.

مقدمة في التنبؤ بأمراض القلب

يعد التنبؤ بأمراض القلب Heart Disease Prediction وتشخيصها التحدي الأكبرفي الصناعة الطبية ويعتمد على عوامل مثل الفحص البدني وأعراض وعلامات المريض.

العوامل التي تؤثر على أمراض القلب هي مستويات الكوليسترولفي الجسم، وعادات التدخين والسمنة، والتاريخ العائلي للأمراض، وضغط الدم، وبيئة العمل. تلعب خوارزميات التعلم الآلي دورًا أساسيًا ودقيقًافي التنبؤ بأمراض القلب.

تسمح التطوراتفي التكنولوجيا للغة الآلة بالاندماج مع أدوات البيانات الضخمة لإدارة البيانات غير المهيكلة والمتنامية بشكل كبير. يُنظر إلى أمراض القلب على أنها أكثر الأمراض فتكًافي العالم في حياة الإنسان. على وجه الخصوص،في هذا النوع من المرض، لا يستطيع القلب دفع الكمية المطلوبة من الدم إلى أعضاء الجسم المتبقية لأداء وظائف منتظمة.

يمكن التنبؤ بأمراض القلب بناءً على أعراض مختلفة مثل العمر والجنس ومعدل ضربات القلب وما إلى ذلك ويقلل من معدل الوفيات لمرضى القلب.

بسبب الاستخدام المتزايد للتكنولوجيا وجمع البيانات، يمكننا الآن التنبؤ بأمراض القلب باستخدام خوارزميات التعلم الآلي. دعنا الآن نذهب إلى أبعد من مهمة التنبؤ بأمراض القلب باستخدام التعلم الآلي باستخدام بايثون.

التنبؤ بأمراض القلب باستخدام التعلم الآلي

الآنفي هذا القسم، سوف آخذك خلال مهمة التنبؤ بأمراض القلب باستخدام التعلم الآلي باستخدام خوارزمية الانحدار اللوجستي. نظرًا لأنني سأستخدم لغة برمجة Python لهذه المهمة الخاصة بالتنبؤ بأمراض القلب، فلنبدأ باستيراد بعض المكتبات الضرورية:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
```

يمكن تنزيل مجموعة البيانات التي أستخدمها هنا بسهولة من هنا. الآن دعنا نستورد البيانات ونتحرك أبعد من ذلك:

```
df = pd.read_csv("heart.csv")
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

تحليل البيانات استكشافية

قبل تدريب الانحدار اللوجستي، نحتاج إلى مراقبة البيانات وتحليلها لمعرفة ما سنعمل معه. الهدف هنا هو معرفة المزيد حول البيانات وتصبح موضوع تصديرفي مجموعة البيانات التي تعمل معها.

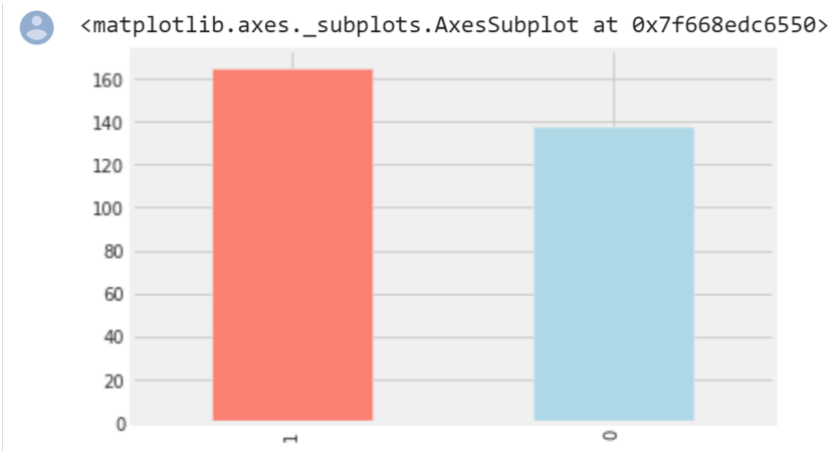
تساعدنا تحليل البيانات استكشافية EDAفي العثور على إجابات لبعض الأسئلة المهمة مثل: ما هو السؤال (الأسئلة) الذي تحاول حله؟ ما نوع البيانات التي لدينا وكيف نتعامل مع الأنواع المختلفة؟ ما هو المفقودفي البيانات وكيف تتعامل معه؟ أين القيم المتطرفة outliers ولماذا يجب أن تهتم؟ كيف يمكنك إضافة ميزات أو تغييرها أو إزالتها لتحقيق أقصى استفادة من بياناتك؟

لنبدأ الآن بتحليل البيانات الاستكشافية:

```
pd.set_option("display.float", "{:.2f}".format)
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00
mean	54.37	0.68	0.97	131.62	246.26	0.15	0.53	149.65	0.33	1.04	1.40	0.73	2.31	0.54
std	9.08	0.47	1.03	17.54	51.83	0.36	0.53	22.91	0.47	1.16	0.62	1.02	0.61	0.50
min	29.00	0.00	0.00	94.00	126.00	0.00	0.00	71.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	47.50	0.00	0.00	120.00	211.00	0.00	0.00	133.50	0.00	0.00	1.00	0.00	2.00	0.00
50%	55.00	1.00	1.00	130.00	240.00	0.00	1.00	153.00	0.00	0.80	1.00	0.00	2.00	1.00
75%	61.00	1.00	2.00	140.00	274.50	0.00	1.00	166.00	1.00	1.60	2.00	1.00	3.00	1.00
max	77.00	1.00	3.00	200.00	564.00	1.00	2.00	202.00	1.00	6.20	2.00	4.00	3.00	1.00

```
df.target.value_counts().plot(kind="bar", color=["salmon",
"lightblue"])
```



لدينا 165 شخصًا يعانون من أمراض القلب و138 مصابًا بأمراض القلب، لذا فإن مشكلتنا متوازنة.

```
# Checking for messing values
df.isna().sum()
```

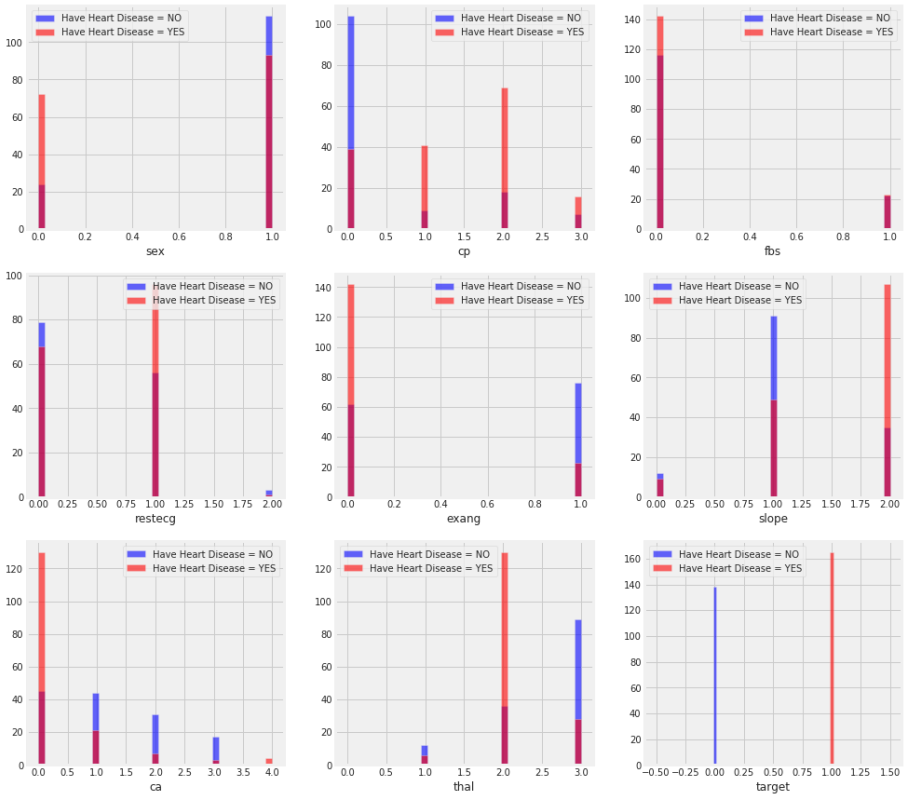
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

تبدو مجموعة البيانات هذه مثالية للاستخدام حيث لا توجد لدينا قيم فارغة.

```
categorical_val = []
continous_val = []
for column in df.columns:
    print('==============================')
    print(f"{column} : {df[column].unique()}")
    if len(df[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
```

```
==============================
age : [63 37 41 56 57 44 52 54 48 49 64 58 50 66 43 69 59 42 61 40 71 51 65 53
 46 45 39 47 62 34 35 29 55 60 67 68 74 76 70 38 77]
==============================
sex : [1 0]
==============================
cp : [3 2 1 0]
==============================
trestbps : [145 130 120 140 172 150 110 135 160 105 125 142 155 104 138 128 108 134
 122 115 118 100 124  94 112 102 152 101 132 148 178 129 180 136 126 106
 156 170 146 117 200 165 174 192 144 123 154 114 164]
==============================
chol : [233 250 204 236 354 192 294 263 199 168 239 275 266 211 283 219 340 226
 247 234 243 302 212 175 417 197 198 177 273 213 304 232 269 360 308 245
 208 264 321 325 235 257 216 256 231 141 252 201 222 260 182 303 265 309
 186 203 183 220 209 258 227 261 221 205 240 318 298 564 277 214 248 255
 207 223 288 160 394 315 246 244 270 195 196 254 126 313 262 215 193 271
 268 267 210 295 306 178 242 180 228 149 278 253 342 157 286 229 284 224
 206 167 230 335 276 353 225 330 290 172 305 188 282 185 326 274 164 307
 249 341 407 217 174 281 289 322 299 300 293 184 409 259 200 327 237 218
 319 166 311 169 187 176 241 131]
==============================
fbs : [1 0]
==============================
restecg : [0 1 2]
==============================
thalach : [150 187 172 178 163 148 153 173 162 174 160 139 171 144 158 114 151 161
 179 137 157 123 152 168 140 188 125 170 165 142 180 143 182 156 115 149
 146 175 186 185 159 130 190 132 147 154 202 166 164 184 122 169 138 111
 145 194 131 133 155 167 192 121  96 126 105 181 116 108 129 120 112 128
 109 113  99 177 141 136  97 127 103 124  88 195 106  95 117  71 118 134
  90]
==============================
exang : [0 1]
==============================
oldpeak : [2.3 3.5 1.4 0.8 0.6 0.4 1.3 0.  0.5 1.6 1.2 0.2 1.8 1.  2.6 1.5 3.  2.4
 0.1 1.9 4.2 1.1 2.  0.7 0.3 0.9 3.6 3.1 3.2 2.5 2.2 2.8 3.4 6.2 4.  5.6
 2.9 2.1 3.8 4.4]
==============================
slope : [0 2 1]
==============================
ca : [0 2 1 3 4]
==============================
thal : [1 2 3 0]
==============================
target : [1 0]
```

```
plt.figure(figsize=(15, 15))

for i, column in enumerate(categorical_val, 1):
    plt.subplot(3, 3, i)
    df[df["target"] == 0][column].hist(bins=35, color='blue',
label='Have Heart Disease = NO', alpha=0.6)
    df[df["target"] == 1][column].hist(bins=35, color='red',
label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```

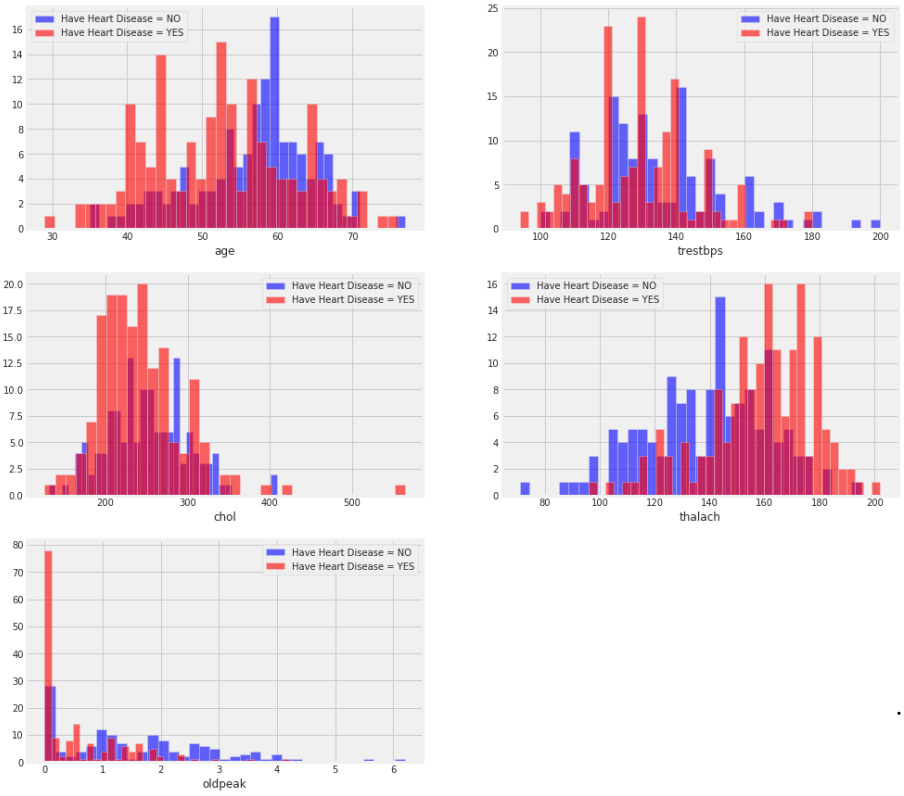


ملاحظات من المخطط أعلاه:

- {Chest pain} cp: الأشخاص الذين يعانون من cp 1، 2 ، 3 هم أكثر عرضة للإصابة بأمراض القلب من الأشخاص الذين يعانون من cp 0.
- {resting EKG results} restecg: الأشخاص الذين لديهم قيمة 1 (الإبلاغ عن إيقاع غير طبيعي للقلب ، والذي يمكن أن يتراوح من أعراض خفيفة إلى مشاكل شديدة) هم أكثر عرضة للإصابة بأمراض القلب.
- exang {الذبحة الصدرية الناتجة عن التمرين}: يعاني الأشخاص الذين تبلغ قيمتها 0 (لا ==> الذبحة الصدرية الناتجة عن ممارسة الرياضة) من أمراض القلب أكثر من الأشخاص الذين تبلغ قيمتها 1 (نعم ==> الذبحة الصدرية الناتجة عن التمرين)
- المنحدر {منحدر الجزء ST من تمرين الذروة}: الأشخاص الذين لديهم قيمة انحدار 2 (Downslopins: علامات تدل على وجود قلب غير صحي) هم أكثر عرضة للإصابة بأمراض القلب من الأشخاص الذين لديهم قيمة انحدار 2 منحدر هو 0 (Upslopins: أفضل معدل لضربات القلب مع التمرين) أو 1 (Flatsloping: الحد الأدنى من التغيير (قلب صحي نموذجي)).

- ca {عدد الأوعية الرئيسية (0ـ3) الملطخة بواسطة التنظير الفلوري}: كلما زادت حركة الدم كلما كان ذلك أفضل ، لذلك فإن الأشخاص الذين لديهم ca يساوي 0 هم أكثر عرضة للإصابة بأمراض القلب.
- thal {نتيجة إجهاد thalium}: الأشخاص الذين لديهم قيمة thal 2 (تم تصحيح العيب: كان ذات مرة عيبًا ولكنه لا بأس به الآن) هم أكثر عرضة للإصابة بأمراض القلب.

```
plt.figure(figsize=(15, 15))

for i, column in enumerate(continous_val, 1):
    plt.subplot(3, 2, i)
    df[df["target"] == 0][column].hist(bins=35, color='blue',
label='Have Heart Disease = NO', alpha=0.6)
    df[df["target"] == 1][column].hist(bins=35, color='red',
label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```

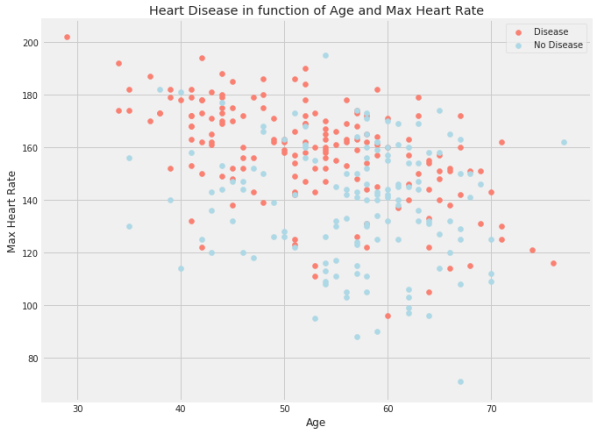


ملاحظات من المخطط أعلاه:

- trestbps: راحة ضغط الدم أي شيء يزيد عن 130ـ140 أمر مثير للقلق بشكل عام

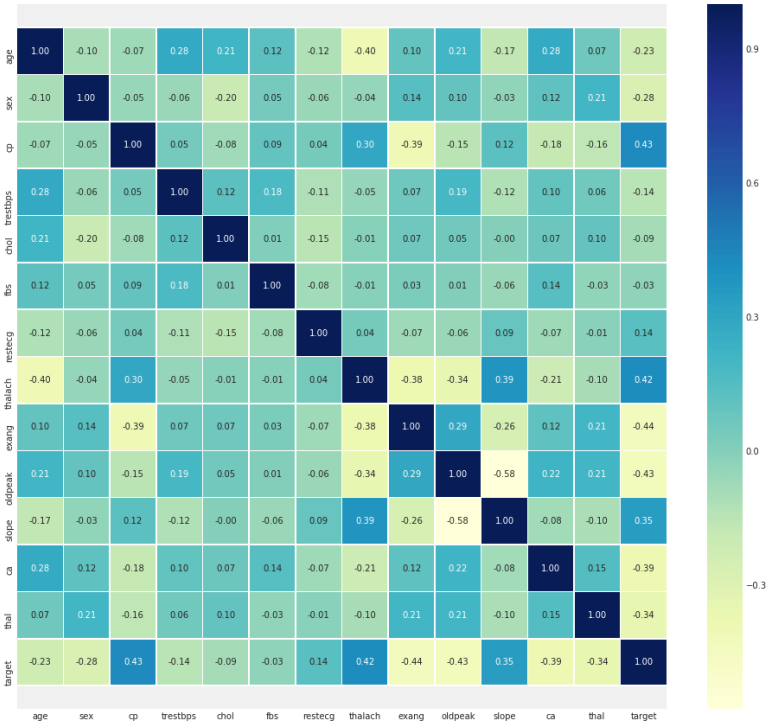
- الكولسترول chol: أكثر من 200 أمر مثير للقلق.
- thalach: الأشخاص الذين تزيد أعمارهم عن 140 كحد أقصى هم أكثر عرضة للإصابة بأمراض القلب.
- الذروة القديمة oldpeak للاكتئاب الناتج عن التمرين مقابل الراحة يؤدي إلى إجهاد القلب أثناء التمرين، فإن القلب غير الصحي سيضغط أكثر.

```
# Create another figure
plt.figure(figsize=(10, 8))

# Scatter with postivie examples
plt.scatter(df.age[df.target==1],
            df.thalach[df.target==1],
            c="salmon")

# Scatter with negative examples
plt.scatter(df.age[df.target==0],
            df.thalach[df.target==0],
            c="lightblue")

# Add some helpful info
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease", "No Disease"]);
```

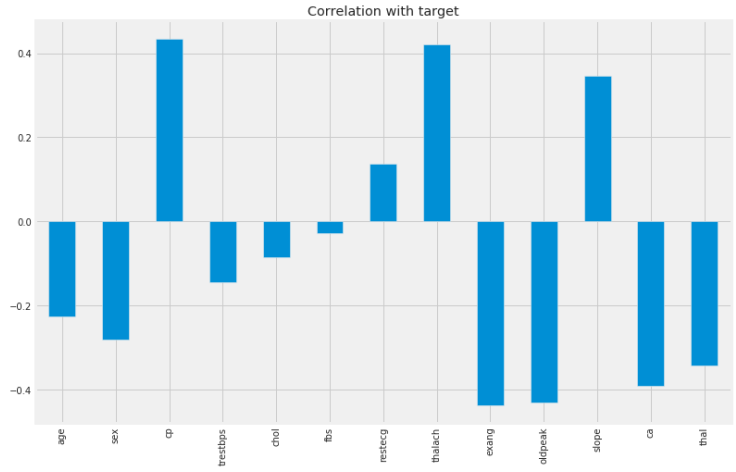


مصفوفة الارتباط

```
# Let's make our correlation matrix a little prettier
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```




```
df.drop('target', axis=1).corrwith(df.target).plot(kind='bar',
grid=True, figsize=(12, 8), title="Correlation with target")
```



ملاحظات من الارتباط:

- fbs و chol هي الأقل ارتباطًا بالمتغير المستهدف.
- جميع المتغيرات الأخرى لها ارتباط كبير مع المتغير المستهدف.

معالجة البيانات

بعد استكشاف مجموعة البيانات، يمكننا ملاحظة أننا بحاجة إلى تحويل بعض المتغيرات الفئوية categorical variables إلى متغيرات وهمية dummy variables وقياس جميع القيم قبل تدريب نماذج التعلم الآلي.

لذلك، بالنسبة لهذه المهمة، سأستخدم طريقة get_dummies لإنشاء أعمدة وهمية للمتغيرات الفئوية:

```
categorical_val.remove('target')
dataset = pd.get_dummies(df, columns = categorical_val)

from sklearn.preprocessing import StandardScaler

s_sc = StandardScaler()
col_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[col_to_scale] = s_sc.fit_transform(dataset[col_to_scale])
```

تطبيق الانحدار اللوجستي

الآن، سأقوم بتدريب نموذج التعلم الآلي لمهمة التنبؤ بأمراض القلب. سأستخدم خوارزمية الانحدار اللوجستي كما ذكرت في بداية المقال.

ولكن قبل تدريب النموذج، سأحدد أولاً دالة مساعدة لطباعة تقرير التصنيف لأداء نموذج التعلم الآلي:

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred,
output_dict=True))
        print("Train
Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) *
100:.2f}%")
        print("_____________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_____________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train,
pred)}\n")

    elif train==False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred,
output_dict=True))
        print("Test
Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) *
100:.2f}%")
        print("_____________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_____________________________________________")
```

```
        print(f"Confusion Matrix: \n {confusion_matrix(y_test,
pred)}\n")
```

دعنا الآن نقسم البيانات إلى مجموعات تدريب واختبار. سأقسم البيانات إلى 70٪ تدريب و30٪ اختبار:

```
from sklearn.model_selection import train_test_split

X = dataset.drop('target', axis=1)
y = dataset.target

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
```

الآن دعنا ندرب نموذج التعلم الآلي ونطبع تقرير التصنيف لنموذج الانحدار اللوجستي الخاص بنا:

```
from sklearn.linear_model import LogisticRegression

lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train, y_train)

print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 86.79%

_______________________________________________
CLASSIFICATION REPORT:
              0      1  accuracy  macro avg  weighted avg
precision  0.88   0.86      0.87       0.87          0.87
recall     0.82   0.90      0.87       0.86          0.87
f1-score   0.85   0.88      0.87       0.87          0.87
support   97.00 115.00      0.87     212.00        212.00
_______________________________________________
Confusion Matrix:
 [[ 80  17]
 [ 11 104]]

Test Result:
================================================
Accuracy Score: 86.81%

_______________________________________________
CLASSIFICATION REPORT:
              0     1  accuracy  macro avg  weighted avg
precision  0.87  0.87      0.87       0.87          0.87
recall     0.83  0.90      0.87       0.86          0.87
f1-score   0.85  0.88      0.87       0.87          0.87
support   41.00 50.00      0.87      91.00         91.00
_______________________________________________
Confusion Matrix:
 [[34  7]
 [ 5 45]]
```

```
test_score = accuracy_score(y_test, lr_clf.predict(X_test)) * 100
train_score = accuracy_score(y_train, lr_clf.predict(X_train)) * 100

results_df = pd.DataFrame(data=[["Logistic Regression", train_score,
test score]],
                          columns=['Model', 'Training Accuracy %',
'Testing Accuracy %'])
results_df
```

Model	Training Accuracy %	Testing Accuracy %
Logistic Regression	86.79	86.81

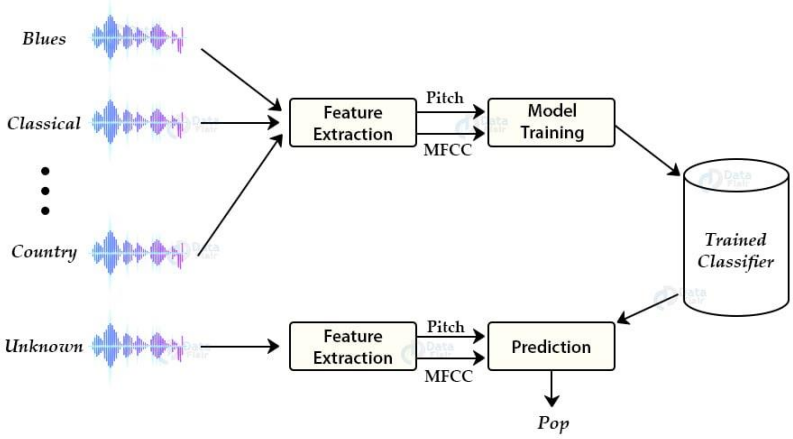
كما ترون، فإن النموذج يؤدي أداءً جيداً لمجموعة الاختبار لأنه يعطي نفس الدقة تقريبًافي مجموعة الاختبار كما هو الحالفي مجموعة التدريب.

لذلك آمل أن تكون قد أحببت هذه المقالة حول كيفية تدريب نموذج التعلم الآلي لمهمة التنبؤ بأمراض القلب باستخدام التعلم الآلي.

8) تصنيف نوع الموسيقى باستخدام التعلم الآلي Music Genre Classification using Machine Learning

سنقومفي هذا البرنامج التعليمي بتطوير مشروع التعلم الآلي لتصنيف الأنواع الموسيقية المختلفة تلقائيًا من الملفات الصوتية. سنقوم بتصنيف هذه الملفات الصوتية باستخدام ميزات منخفضة المستوى للتردد والمجال الزمني.

بالنسبة لهذا المشروع، نحتاج إلى مجموعة بيانات من المسارات الصوتية لها نفس الحجم ونطاق تردد مماثل. مجموعة بيانات تصنيف النوع GTZAN هي مجموعة البيانات الموصى بها لمشروع تصنيف نوع الموسيقى وقد تم جمعها لهذه المهمة فقط.



تصنيف نوع الموسيقى

حول مجموعة البيانات:

تم جمع مجموعة بيانات مجموعة النوع GTZANفي 2000ـ2001. يتكون من 1000 ملف صوتي مدة كل منها 30 ثانية. هناك 10 فئات (10 أنواع موسيقية) يحتوي كل منها على 100 مقطع صوتي. كل مسار بتنسيق wav.. يحتوي على ملفات صوتية من الأنواع العشرة التالية:

- Blues
- Classical
- Country
- Disco
- Hiphop
- Jazz
- Metal
- Pop
- Reggae
- Rock

نهج تصنيف نوع الموسيقى:

هناك طرق مختلفة لإجراء التصنيف على مجموعة البيانات هذه. بعض هذه الأساليب هي:

- Multiclass support vector machines
- K-means clustering
- K-nearest neighbors
- Convolutional neural networks

سنستخدم خوارزمية K ـ اقرب الجيران K-Nearest Neighbours لأنها أظهرتفي العديد من الأبحاث أفضل النتائج لهذه المشكلة.

K-Nearest Neighbours هي خوارزمية تعلم آلي شائعة للانحدار والتصنيف. يقوم بعمل تنبؤات حول نقاط البيانات بناءً على مقاييس التشابه الخاصة بها، أي المسافة بينها.

استخراج الميزات:

تتمثل الخطوة الأولى لمشروع تصنيف نوع الموسيقىفي استخراج الميزات والمكونات من الملفات الصوتية. يتضمن تحديد المحتوى اللغوي ونبذ الضجيج.

MFCC:

هذه هي أحدث الميزات المستخدمةفي دراسات التعرف على الكلام والكلام التلقائي. هناك مجموعة من الخطوات لإنشاء هذه الميزات:

- نظرًا لأن الإشارات الصوتية تتغير باستمرار ، فإننا نقسم أولاً هذه الإشارات إلى إطارات أصغر. يبلغ طول كل إطار حوالي 20ـ40 مللي ثانية.
- ثم نحاول تحديد الترددات المختلفة الموجودة في كل إطار.
- الآن ، افصل الترددات اللغوية عن الضوضاء.
- للتخلص من الضوضاء، فإنه يأخذ تحويل جيب التمام المنفصل (DCT) لهذه الترددات. باستخدام تقنية DCT ، نحتفظ فقط بتسلسل محدد من الترددات التي تحتوي على احتمالية عالية للمعلومات.

خطوات بناء تصنيف نوع الموسيقى:

قم بتنزيل مجموعة بيانات GTZAN من الرابط التالي:

مجموعة بيانات GTZAN

قم بإنشاء ملف python جديد "music_genre.py" والصق الكود الموضحفي الخطوات أدناه:

1. استيراد المكتبات الضرورية:

```
from python_speech_features import mfcc
import scipy.io.wavfile as wav
import numpy as np
```

```
from tempfile import TemporaryFile
import os
import pickle
import random
import operator

import math
import numpy as np
```

2. حدد دالة للحصول على المسافة بين متجهات الميزات والعثور على الجيران:

```
def getNeighbors(trainingSet, instance, k):
    distances = []
    for x in range (len(trainingSet)):
        dist = distance(trainingSet[x], instance, k )+
distance(instance, trainingSet[x], k)
        distances.append((trainingSet[x][2], dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors
```

3. تحديد أقرب الجيران:

```
def nearestClass(neighbors):
    classVote = {}

    for x in range(len(neighbors)):
        response = neighbors[x]
        if response in classVote:
            classVote[response]+=1
        else:
            classVote[response]=1

    sorter = sorted(classVote.items(), key = operator.itemgetter(1),
reverse=True)
    return sorter[0][0]
```

4. تحديد دالة لتقييم النموذج:

```
def getAccuracy(testSet, predictions):
    correct = 0
    for x in range (len(testSet)):
        if testSet[x][-1]==predictions[x]:
            correct+=1
    return 1.0*correct/len(testSet)
```

5. استخراج الميزات من مجموعة البيانات وتفريغ هذه الميزات في ملف ثنائي dat. "my.dat":

```
directory = "__path_to_dataset__"
f= open("my.dat" ,'wb')
i=0

for folder in os.listdir(directory):
    i+=1
    if i==11 :
        break
    for file in os.listdir(directory+folder):
        (rate,sig) = wav.read(directory+folder+"/"+file)
        mfcc_feat = mfcc(sig,rate ,winlen=0.020, appendEnergy = False)
        covariance = np.cov(np.matrix.transpose(mfcc_feat))
        mean_matrix = mfcc_feat.mean(0)
```

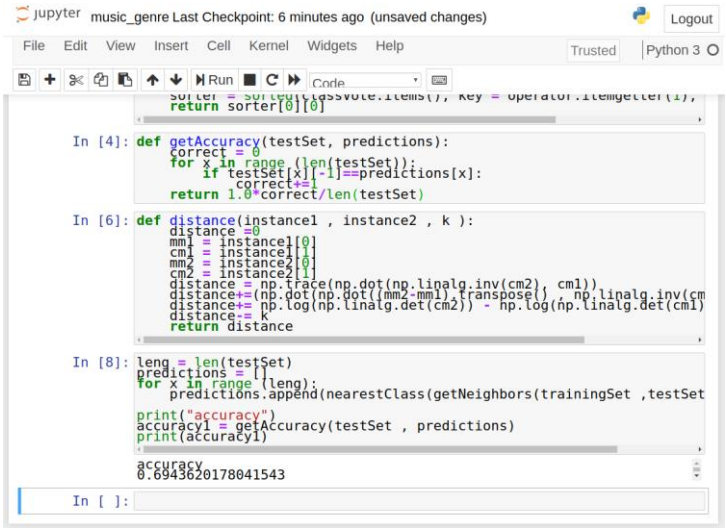
```
        feature = (mean_matrix , covariance , i)
        pickle.dump(feature , f)

f.close()
```

6. تقسيم التدريب والاختبار على مجموعة البيانات:

```
dataset = []
def loadDataset(filename , split , trSet , teSet):
    with open("my.dat" , 'rb') as f:
        while True:
            try:
                dataset.append(pickle.load(f))
            except EOFError:
                f.close()
                break

    for x in range(len(dataset)):
        if random.random() <split :
            trSet.append(dataset[x])
        else:
            teSet.append(dataset[x])

trainingSet = []
testSet = []
loadDataset("my.dat" , 0.66, trainingSet, testSet)
```

7. التنبؤ باستخدام KNN واحصل على دقة بيانات الاختبار:

```
leng = len(testSet)
predictions = []
for x in range (leng):
    predictions.append(nearestClass(getNeighbors(trainingSet
,testSet[x] , 5)))

accuracy1 = getAccuracy(testSet , predictions)
print(accuracy1)
```



8.اختبار المصنف بملف صوتي جديد

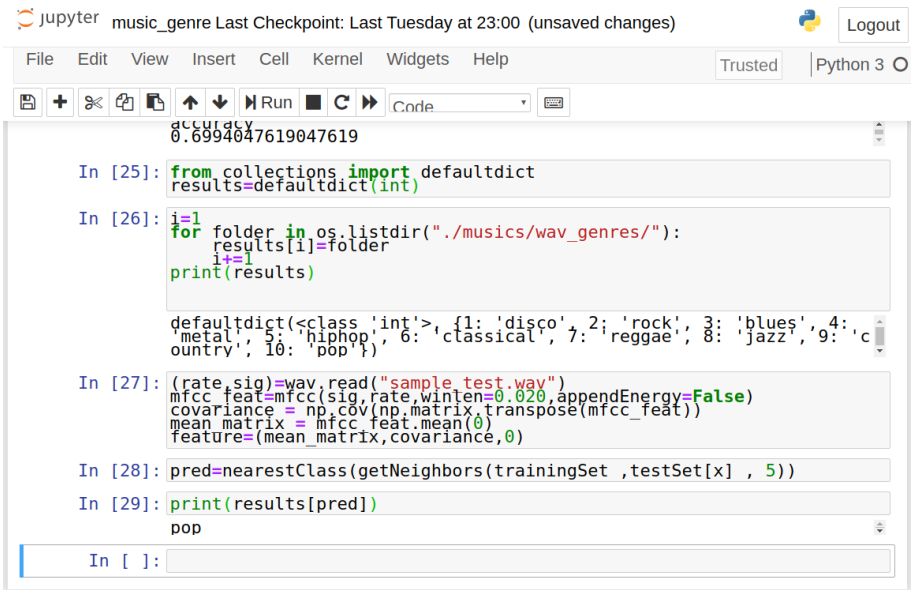
احفظ ملف الصوت الجديدفي الدليل الحالي. قم بإنشاء ملف جديد test.py وألصق السكريبت أدناه:

```
from python_speech_features import mfcc
import scipy.io.wavfile as wav
import numpy as np
from tempfile import TemporaryFile
import os
import pickle
import random
import operator

import math
import numpy as np
from collections import defaultdict

dataset = []
def loadDataset(filename):
    with open("my.dat" , 'rb') as f:
        while True:
            try:
                dataset.append(pickle.load(f))
            except EOFError:
                f.close()
                break

loadDataset("my.dat")

def distance(instance1 , instance2 , k ):
    distance =0
    mm1 = instance1[0]
    cm1 = instance1[1]
    mm2 = instance2[0]
    cm2 = instance2[1]
    distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))
    distance+=(np.dot(np.dot((mm2-mm1).transpose() ,
np.linalg.inv(cm2)) , mm2-mm1 ))
    distance+= np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1))
    distance-= k
    return distance

def getNeighbors(trainingSet , instance , k):
    distances =[]
    for x in range (len(trainingSet)):
        dist = distance(trainingSet[x], instance, k )+
distance(instance, trainingSet[x], k)
        distances.append((trainingSet[x][2], dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors

def nearestClass(neighbors):
    classVote ={}
    for x in range(len(neighbors)):
        response = neighbors[x]
        if response in classVote:
```

```
            classVote[response]+=1
        else:
            classVote[response]=1
    sorter = sorted(classVote.items(), key = operator.itemgetter(1),
reverse=True)
    return sorter[0][0]


results=defaultdict(int)

i=1
for folder in os.listdir("./musics/wav_genres/"):
    results[i]=folder
    i+=1

(rate,sig)=wav.read("__path_to_new_audio_file_")
mfcc_feat=mfcc(sig,rate,winlen=0.020,appendEnergy=False)
covariance = np.cov(np.matrix.transpose(mfcc_feat))
mean_matrix = mfcc_feat.mean(0)
feature=(mean_matrix,covariance,0)

pred=nearestClass(getNeighbors(dataset ,feature , 5))

print(results[pred])
```

الآن، قم بتشغيل هذا السكريبت للحصول على التنبؤ:

```
python3 test.py
```



الملخص

في مشروع تصنيف نوع الموسيقى هذا، قمنا بتطوير مصنف لملفات الصوت للتنبؤ بنوعها. نحن نعمل من خلال هذا المشروع على مجموعة بيانات تصنيف نوع الموسيقى GTZAN. يشرح هذا البرنامج التعليمي كيفية استخراج الميزات المهمة من الملفات الصوتية.في مشروع التعلم العميق هذا، قمنا بتنفيذ أقرب جوار K باستخدام عدد K على أنه 5..

9) تصنيف أسعار الهواتف الذكية باستخدام التعلم الآلي Mobile Price Classification using Machine Learning

سعر المنتج هو أهم سمة لتسويق هذا المنتج. أحد تلك المنتجات التي يكون السعر فيها مهمًا كثيرًا هو الهاتف الذكي لأنه يأتي مع الكثير من الميزات بحيث تفكر الشركة كثيرًافي كيفية تسعير هذا الهاتف المحمول الذي يمكن أن يبرر الميزات ويغطي أيضًا تكاليف التسويق والتصنيع للهاتف المحمول.في هذه المقالة، سوف أطلعك على مهمة تصنيف أسعار الأجهزة المحمولة باستخدام التعلم الآلي باستخدام Python.

تصنيف أسعار الأجهزة المحمولة مع التعلم الآلي

الهواتف المحمولة هي الأجهزة الإلكترونية الأكثر مبيعًا حيث يواصل الأشخاص تحديث هواتفهم المحمولة كلما وجدوا ميزات جديدةفي جهاز جديد. يتم بيع الآلاف من الهواتف المحمولة يوميًا، وفي مثل هذه الحالة يكون من الصعب جدًا على الشخص الذي يخطط لإنشاء أعمال الهاتف المحمول الخاصة به أن يقرر السعر الذي يجب أن يكون عليه الهاتف المحمول.

في القسم أدناه، سأقدم لك مشروع التعلم الآلي على نموذج تصنيف أسعار الأجهزة المحمولة حيث سأقوم بتدريب نموذج لتصنيف النطاق السعري للهواتف المحمولة باستخدام Python. تعتمد هذه المهمة على حل دراسة الحالة المذكورة أدناه.

" يريد السيد أمان أن يبدأ شركة الهواتف المحمولة الخاصة به ويريد أن يخوض معركة شاقة مع العلامات التجارية الكبرى للهواتف الذكية مثل Samsung وApple. لكنه لا يعرف كيفية تقدير سعر الهاتف المحمول الذي يمكن أن يغطي كلاً من تكاليف التسويق والتصنيع. لذافي هذه المهمة، لا يتعين عليك توقع الأسعار الفعلية للهواتف المحمولة ولكن عليك توقع النطاق السعري للهواتف المحمولة. "

تصنيف أسعار الأجهزة المحمولة باستخدام لغة بايثون

لذلك، نظرًا لأن مهمتنا هي تصنيف النطاق السعري للهواتف المحمولة وليس التنبؤ بالأسعار الفعلية، لذلك سأقوم هنا بتدريب نموذج تصنيف لتصنيف النطاق السعري للهواتف المحمولة على النحو التالي:

- 0 (تكلفة منخفضة).
- 1 (تكلفة متوسطة).
- 2 (تكلفة عالية).
- 4 (تكلفة عالية جدا).

سأبدأ مهمة تصنيف أسعار الأجهزة المحمولة باستخدام التعلم الآلي عن طريق استيراد مكتبات Python ومجموعة البيانات اللازمة:

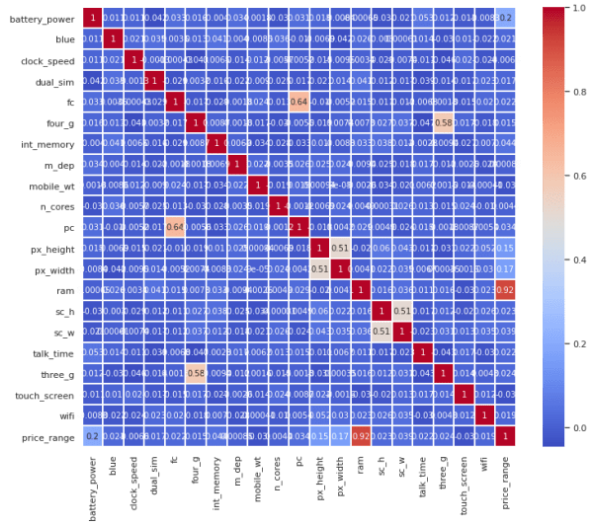
مجموعة البيانات

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
sns.set()

data = pd.read_csv("mobile_prices.csv")
print(data.head())
```

```
   battery_power  blue  clock_speed  ...  touch_screen  wifi  price_range
0            842     0          2.2  ...             0     1            1
1           1021     1          0.5  ...             1     0            2
2            563     1          0.5  ...             1     0            2
3            615     1          2.5  ...             0     0            2
4           1821     1          1.2  ...             1     0            1

[5 rows x 21 columns]
```

لذا، تحتوي مجموعة البيانات على 21 عمودًا ولحسن الحظ لا تحتوي مجموعة البيانات هذه على قيم مفقودة، لذا يمكننا البدء فقط من خلال تدريب النموذج، ولكن قبل ذلك، دعنا نلقي نظرة على الارتباط بين الميزاتفي مجموعة البيانات:

```
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr(), annot=True, cmap="coolwarm",
linecolor='white', linewidths=1)
```



تحضير البيانات

لا تحتوي مجموعة البيانات هذه على ميزات فئوية، لذلك يمكننا فقط استخدام هذه البيانات دون أي تحويل لأن جميع ميزات مجموعة البيانات رقمية. ولكن لتدريب نموذج، من المهم جدًا توحيد البيانات أو تطبيعها وتقسيمها إلى مجموعات تدريب واختبار.

لذلك دعونا نوحد مجموعة البيانات ونقسم البيانات إلى 80٪ تدريب و20٪ اختبار:

```
x = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
x = StandardScaler().fit_transform(x)
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.20, random_state=0)
```

نموذج تصنيف سعر الهاتف المحمول

دعنا الآن ندرب نموذج تصنيف أسعار الجوّال باستخدام Python. نظرًا لأن هذه مشكلة تصنيف، فسوف أستخدم خوارزمية الانحدار اللوجستي Logistic Regression المقدمة من Scikit-Learn:

```
from sklearn.linear_model import LogisticRegression
lreg = LogisticRegression()
lreg.fit(x_train, y_train)
y_pred = lreg.predict(x_test)
```

الآن دعنا نلقي نظرة على دقة النموذج:

```
accuracy = accuracy_score(y_test, y_pred) * 100
print("Accuracy of the Logistic Regression Model: ",accuracy)
```

```
Accuracy of the Logistic Regression Model:  95.5
```

لذا فإن النموذج يعطي دقة تبلغ حوالي 95.5٪ وهو أمر رائع. دعنا الآن نلقي نظرة على التنبؤات التي قدمها النموذج:

```
[3 0 2 2 3 0 0 3 3 1 1 3 0 2 3 0 3 2 2 1 0 0 3 1 2 2 3 1 3 1 1 0 2 0 2 3 0
 0 3 3 3 1 3 3 1 3 0 1 3 1 1 3 0 3 0 2 2 2 0 3 3 1 3 2 1 2 3 2 2 2 3 2 1 0
 1 3 2 2 1 2 3 3 3 0 0 0 2 1 2 3 1 2 2 1 0 3 3 3 0 3 1 1 3 1 3 2 2 3 2 3 3
 0 0 1 3 3 0 0 1 0 0 3 2 2 1 2 1 1 0 2 1 3 3 3 3 3 3 2 0 1 1 2 1 3 0 3 0 0
 2 0 1 1 1 1 3 0 0 3 1 3 2 1 3 1 2 3 3 2 1 0 3 1 2 3 3 0 2 2 3 1 2 1 0 1 2
 2 2 0 3 3 1 1 0 2 3 0 1 2 2 0 3 3 3 1 2 3 3 3 0 0 0 2 3 3 0 0 1 3 2 3 3 3
 0 0 2 3 3 1 0 2 0 0 0 3 2 1 2 2 1 1 0 2 3 3 0 0 1 3 3 1 3 0 3 1 1 0 2 3 3
 2 0 0 1 2 3 2 2 3 2 1 0 3 3 2 1 3 2 2 2 1 0 2 2 1 0 0 2 2 2 3 0 1 3 0 2 2
 3 0 2 0 1 1 3 0 0 2 3 1 2 0 2 0 3 0 3 3 2 3 1 2 2 1 1 1 0 1 0 3 1 0 3 0 0
 1 3 0 3 1 1 0 1 3 0 2 1 1 2 1 1 0 2 0 0 3 1 2 3 3 2 2 0 3 2 2 1 3 2 3 3 3 0
 2 0 3 0 1 1 2 3 1 3 1 2 0 1 2 3 0 0 1 3 0 3 0 2 2 1 1 0 2 0]
```

يوضح الناتج أعلاه النطاق السعري المصنف بواسطة النموذج. دعونا نلقي نظرة على عدد الهواتف المحمولة المصنفة لكل نطاق سعري:

```
(unique, counts) = np.unique(y_pred, return_counts=True)
price_range = np.asarray((unique, counts)).T
print(price_range)
```

```
[[  0  95]
 [  1  90]
 [  2  97]
 [  3 118]]
```

أتمنى أن تكون قد أحببت هذه المقالة حول تصنيف أسعار الأجهزة المحمولة باستخدام التعلم الآلي.

.

10) كشف السخرية باستخدام التعلم الآلي Sarcasm Detection using Machine Learning

كانت السخرية Sarcasm جزءًا من لغتنا لسنوات عديدة. هذا يعني أن تكون عكس ما تعنيه، عادةً بنبرة صوت مميزة بطريقة ممتعة. إذا كنت تعتقد أن أي شخص يمكنه فهم السخرية، فأنت مخطئ، لأن فهم السخرية يعتمد على مهاراتك اللغوية ومعرفتك بعقول الآخرين. لكن ماذا عن الكمبيوتر؟ هل من الممكن تدريب نموذج للتعلم الآلي يمكنه اكتشاف ما إذا كانت الجملة ساخرة أم لا؟ نعم إنه كذلك! لذلك إذا كنت تريد معرفة كيفية اكتشاف السخرية Sarcasm detection باستخدام التعلم الآلي، فهذه المقالة مناسبة لك.في هذه المقالة، سوف أطلعك على مهمة الكشف عن السخرية باستخدام التعلم الآلي باستخدام python.

اكتشاف السخرية مع التعلم الآلي

السخرية تعني أن تكون مضحكا بأن تكون عكس ما تعنيه. لقد كانت جزءًا من كل لغة بشرية لسنوات. اليوم، يتم استخدامه أيضًافي عناوين الأخبار والعديد من منصات التواصل الاجتماعي الأخرى لجذب المزيد من الاهتمام. اكتشاف السخرية هو معالجة لغة طبيعية ومهمة تصنيف ثنائي. يمكننا تدريب نموذج التعلم الآلي لاكتشاف ما إذا كانت الجملة ساخرة أم لا باستخدام مجموعة بيانات من الجمل الساخرة وغير الساخرة التي وجدتهافي Kaggle.

أتمنى أن تكون قد فهمت الآن ما هي السخرية.في القسم أدناه، سأوجهك خلال مهمة اكتشاف السخرية من خلال التعلم الآلي باستخدام لغة برمجة Python. يمكن تنزيل مجموعة البيانات التي أستخدمها لهذه المهمة من هنا.

اكتشاف السخرية باستخدام بايثون

لنبدأ الآن بمهمة اكتشاف السخرية باستخدام التعلم الآلي باستخدام Python. سأبدأ هذه المهمة عن طريق استيراد مكتبات Python ومجموعة البيانات الضرورية:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB

data = pd.read_json("Sarcasm.json", lines=True)
print(data.head())
```

```
                                article_link  ... is_sarcastic
0  https://www.huffingtonpost.com/entry/versace-b...  ...            0
1  https://www.huffingtonpost.com/entry/roseanne-...  ...            0
2  https://local.theonion.com/mom-starting-to-fea...  ...            1
3  https://politics.theonion.com/boehner-just-wan...  ...            1
4  https://www.huffingtonpost.com/entry/jk-rowlin...  ...            0

[5 rows x 3 columns]
```

يحتوي عمود "is_sarcastic"في مجموعة البيانات هذه على التسميات التي يتعين علينا توقعها لمهمة اكتشاف السخرية. يحتوي على قيم ثنائية مثل 1 و0، حيث 1 تعني ساخرة sarcastic و 0 ليست ساخرة not sarcastic. من أجل التبسيط، سأحول قيم هذا العمود إلى "sarcastic" و "not sarcastic" بدلاً من 1 و0:

```
data["is_sarcastic"] = data["is_sarcastic"].map({0: "Not Sarcasm", 1:
"Sarcasm"})
print(data.head())
```

```
                                article_link  ... is_sarcastic
0  https://www.huffingtonpost.com/entry/versace-b...  ...  Not Sarcasm
1  https://www.huffingtonpost.com/entry/roseanne-...  ...  Not Sarcasm
2  https://local.theonion.com/mom-starting-to-fea...  ...     Sarcasm
3  https://politics.theonion.com/boehner-just-wan...  ...     Sarcasm
4  https://www.huffingtonpost.com/entry/jk-rowlin...  ...  Not Sarcasm

[5 rows x 3 columns]
```

دعنا الآن نجهز البيانات لتدريب نموذج التعلم الآلي. تحتوي مجموعة البيانات هذه على ثلاثة أعمدة، نحتاج منها فقط إلى عمود "headline" كميزة والعمود "is_sarcastic" كتسمية. لذلك دعونا نحدد هذه الأعمدة ونقسم البيانات إلى مجموعة اختبار 20٪ ومجموعة تدريب 80٪:

```
data = data[["headline", "is_sarcastic"]]
x = np.array(data["headline"])
y = np.array(data["is_sarcastic"])

cv = CountVectorizer()
X = cv.fit_transform(x) # Fit the Data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=42)
```

سأستخدم الآن خوارزمية Bernoulli Naive Bayes لتدريب نموذج لمهمة اكتشاف السخرية:

```
model = BernoulliNB()
model.fit(X_train, y_train)
print(model.score(X_test, y_test))
```

```
0.8448146761512542
```

دعنا الآن نستخدم نصًا ساخرًا كمدخلات لاختبار ما إذا كان نموذج التعلم الآلي لدينا يكتشف السخرية أم لا:

```
user = input("Enter a Text: ")
data = cv.transform([user]).toarray()
output = model.predict(data)
print(output)
```

```
Enter a Text: Cows lose their jobs as milk prices drop
['Sarcasm']
```

الملخص

هذه هي الطريقة التي يمكنك بها استخدام التعلم الآلي لاكتشاف السخرية باستخدام لغة برمجة Python. كانت السخرية جزءًا من لغتنا لسنوات عديدة. هذا يعني أن تكون عكس ما تعنيه، عادةً بنبرة صوت مميزة بطريقة ممتعة. أتمنى أن تكون قد أحببت هذه المقالة حول مهمة الكشف عن السخرية باستخدام التعلم الآلي باستخدام Python.

11) توقع داء السكري باستخدام التعلم الآلي Predict Diabetes using Machine Learning

وفقًا لتقرير مراكز السيطرة على الأمراض والوقاية منها، فإن واحدًا من كل سبعة بالغين في الولايات المتحدة مصاب بمرض السكري. ولكن في السنوات القليلة القادمة يمكن أن يرتفع هذا المعدل. مع وضع هذا في الاعتبار اليوم، في هذه المقالة، سأوضح لك كيف يمكنك استخدام التعلم الآلي للتنبؤ بمرض السكري باستخدام Python.

دعنا ندخل على الفور إلى البيانات، يمكنك تنزيل البيانات التي استخدمتها في هذه المقالة إلى توقع داء السكري أدناه.

مجموع البيانات

الآن دعنا نستورد البيانات ونبدأ:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
diabetes = pd.read_csv('diabetes.csv')
print(diabetes.columns)
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure',
'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction',
'Age', 'Outcome'], dtype='object')
```

```
diabetes.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

تتكون مجموعة بيانات مرض السكري من 768 نقطة بيانات، مع 9 ميزات لكل منها:

```
print("dimension of diabetes data: {}".format(diabetes.shape))
```

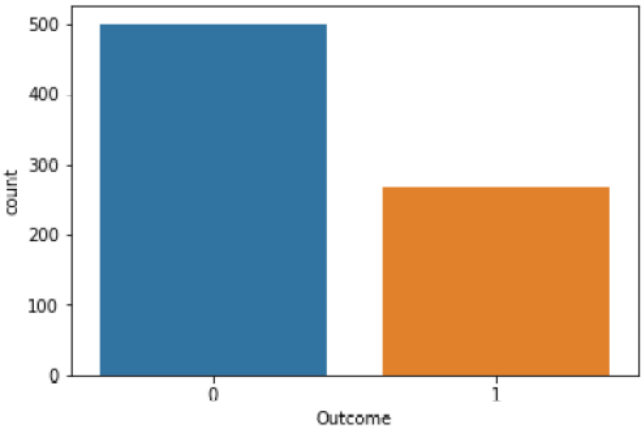
```
dimension of diabetes data: (768, 9)
```

"Outcome" هي الميزة التي سنتوقعها، 0 تعني عدم وجود مرض السكري، 1 يعني مرض السكري. من بين 768 نقطة بيانات، تم تصنيف 500 كـ 0 و268 كـ 1:

```
print(diabetes.groupby('Outcome').size())
```

```
Outcome
0    500
1    268
dtype: int64
```

```
import seaborn as sns
sns.countplot(diabetes['Outcome'],label="Count")
```




```
diabetes.info()
```

```
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

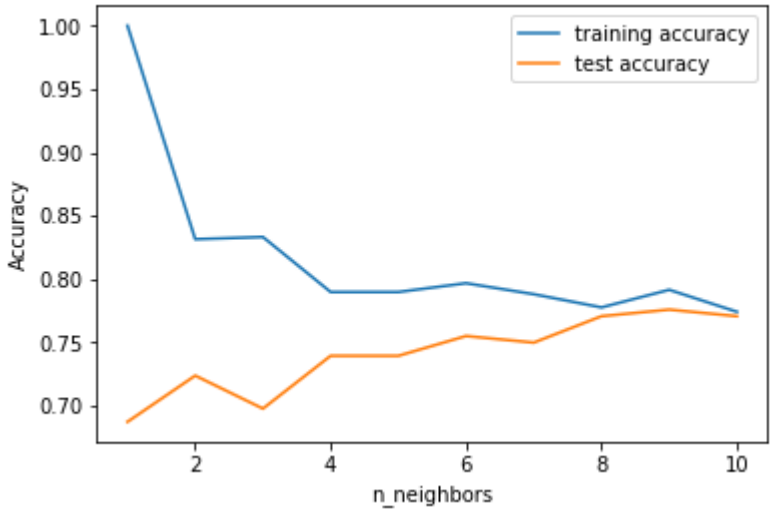
K- أقرب الجيران لتوقع مرض السكري

يمكن القول إن خوارزمية k-Nearest Neighbours هي أبسط خوارزمية تعلم الآلة. يتكون بناء النموذج فقط من تخزين مجموعة بيانات التدريب. لعمل توقع لنقطة جديدةفي مجموعة البيانات، تعثر الخوارزمية على أقرب نقاط البياناتفي مجموعة بيانات التدريب - "أقرب جيرانها".

أولاً، دعنا نتحرى ما إذا كان بإمكاننا تأكيد العلاقة بين تعقيد النموذج ودقته:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(diabetes.loc[:,
diabetes.columns != 'Outcome'], diabetes['Outcome'],
stratify=diabetes['Outcome'], random_state=66)
from sklearn.neighbors import KNeighborsClassifier
training_accuracy = []
test_accuracy = []
```

```
# try n_neighbors from 1 to 10
neighbors_settings = range(1, 11)
for n_neighbors in neighbors_settings:
    # build the model
    knn = KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(knn.score(X_train, y_train))
    # record test set accuracy
    test_accuracy.append(knn.score(X_test, y_test))
plt.plot(neighbors_settings, training_accuracy, label="training
accuracy")
plt.plot(neighbors_settings, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()
```



دعنا نتحقق من درجة دقة خوارزمية k_اقرب الجيران للتنبؤ بمرض السكري.

```
Accuracy of K-NN classifier on training set: 0.79
Accuracy of K-NN classifier on test set: 0.78
```

مصنف شجرة القرار

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train,
y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test,
y_test)))
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.714
```

الدقةفي مجموعة التدريب باستخدام مصنف شجرة القرار Decision Tree Classifier هي 100٪، بينما دقة مجموعة الاختبار أسوأ بكثير. هذا مؤشر على أن الشجرة تعاني من الضبط الزائد

overfitting ولا تعمم جيداً على البيانات الجديدة. لذلك، نحتاج إلى تطبيق التقليم المسبق pre-pruning على الشجرة.

الآن، سأفعل ذلك مرة أخرى عن طريق تعيين max_depth = 3، مما يقلل من عمق الشجرة. يؤدي هذا إلى دقة أقل في مجموعة التدريب، ولكن يؤدي إلى تحسين مجموعة الاختبار.

```
tree = DecisionTreeClassifier(max_depth=3, random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train,
y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X test,
y_test)))
```

```
Accuracy on training set: 0.773
Accuracy on test set: 0.740
```

أهمية الميزة في أشجار القرار

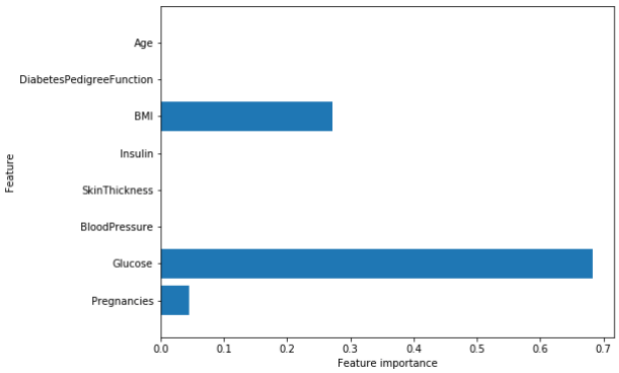
توضح أهمية الميزة مدى أهمية كل ميزة بالنسبة للقرار الذي يتخذه مصنف شجرة القرار. وهو رقم بين 0 و 1 لكل ميزة، حيث يعني 0 "غير مستخدم على الإطلاق" ويعني 1 "يتنبأ تمامًا بالهدف". تتلخص أهمية الميزة دائمًا في 1:

```
print("Feature importances:\n{}".format(tree.feature_importances_))
```

```
Feature importances: [ 0.04554275 0.6830362 0. 0. 0. 0.27142106 0. 0. ]
```

الآن دعنا نرسم أهمية ميزة شجرة القرار للتنبؤ بمرض السكري.

```
def plot_feature_importances_diabetes(model):
    plt.figure(figsize=(8,6))
    n_features = 8
    plt.barh(range(n_features), model.feature_importances_,
align='center')
    plt.yticks(np.arange(n_features), diabetes_features)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)
plot_feature_importances_diabetes(tree)
```



لذلك يتم استخدام ميزة الجلوكوز Glucose أكثر من غيرها للتنبؤ بمرض السكري.

الشبكات العصبية لتوقع مرض السكري

دعنا ندرب نموذج الشبكة العصبية للتنبؤ بمرض السكري:

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(random state=42)
mlp.fit(X_train, y_train)
print("Accuracy on training set: {:.2f}".format(mlp.score(X_train,
y_train)))
print("Accuracy on test set: {:.2f}".format(mlp.score(X_test,
y_test)))
```

```
Accuracy on training set: 0.71
Accuracy on test set: 0.67
```

دقة البيرسيبترون متعدد الطبقات (MLP) ليست جيدة مثل النماذج الأخرى على الإطلاق، فمن المحتمل أن يكون هذا بسبب تحجيم البيانات. تتوقع خوارزميات التعلم العميق أيضًا أن تختلف جميع ميزات الإدخال بطريقة مماثلة، ومن الأفضل أن يكون لها متوسط 0، وتباين 1. الآن سأقوم بإعادة قياس بياناتنا بحيث تفي بهذه المتطلبات للتنبؤ بمرض السكري بدقة جيدة.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X train scaled = scaler.fit transform(X train)
X_test_scaled = scaler.fit_transform(X_test)
mlp = MLPClassifier(random_state=0)
mlp.fit(X_train_scaled, y_train)
print("Accuracy on training set: {:.3f}".format(
    mlp.score(X_train_scaled, y_train)))
print("Accuracy on test set: {:.3f}".format(mlp.score(X_test_scaled,
y_test)))
```

```
Accuracy on training set: 0.823
Accuracy on test set: 0.802
```

دعنا الآن نزيد عدد التكرارات ومعلمة ألفا ونضيف معاملات أقوى إلى أوزان النموذج:

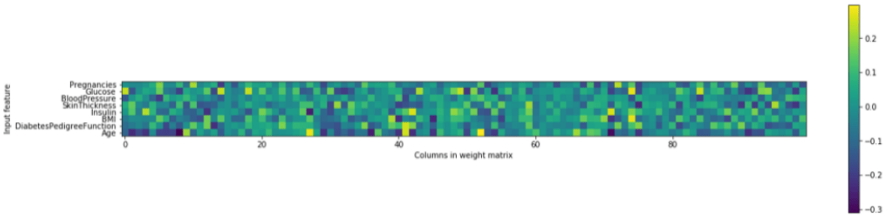
```
mlp = MLPClassifier(max_iter=1000, alpha=1, random_state=0)
mlp.fit(X_train_scaled, y_train)
print("Accuracy on training set: {:.3f}".format(
    mlp.score(X_train_scaled, y_train)))
print("Accuracy on test set: {:.3f}".format(mlp.score(X_test_scaled,
y_test)))
```

```
Accuracy on training set: 0.795
Accuracy on test set: 0.792
```

كانت النتيجة جيدة، لكننا غير قادرين على زيادة دقة الاختبار بشكل أكبر. لذلك، فإن أفضل نموذج لدينا حتى الآن هو نموذج الشبكة العصبية الافتراضي بعد القياس. الآن سوف أرسم خريطة حرارية لأوزان الطبقة الأولىفي الشبكة العصبية التي تم تعلمها من أجل التنبؤ بمرض السكري باستخدام مجموعة البيانات.

```
plt.figure(figsize=(20, 5))
plt.imshow(mlp.coefs_[0], interpolation='none', cmap='viridis')
```

```
plt.yticks(range(8), diabetes_features)
plt.xlabel("Columns in weight matrix")
plt.ylabel("Input feature")
plt.colorbar()
```



12) توقع أسعار المنازل باستخدام التعلم الآلي House Price Prediction using Machine learning

يمكن أن يساعد توقع أسعار المنازلفي تحديد سعر بيع منزلفي منطقة معينة ويمكن أن يساعد الأشخاصفي العثور على الوقت المناسب لشراء منزل.في هذه المقالة، سأقدم لك مشروع التعلم الآلي حول التنبؤ بأسعار المنزل باستخدام Python.

توقع سعر المنزل

في هذه المهمة الخاصة بتوقع أسعار المنازل باستخدام التعلم الآلي، تتمثل مهمتنافي استخدام البيانات من تعداد كاليفورنيا لإنشاء نموذج للتعلم الآلي للتنبؤ بأسعار المنازلفي الولاية. تتضمن البيانات ميزات مثل السكان ومتوسط الدخل وأسعار المنازل المتوسطة لكل مجموعة مجمعةفي كاليفورنيا.

مجموعات الكتل هي أصغر وحدة جغرافية يبلغ عدد سكانها عادة 600 إلى 3000 شخص. يمكننا أن نسميها مناطق باختصار.في النهاية، يجب أن يتعلم نموذج التعلم الآلي الخاص بنا من هذه البيانات وأن يكون قادرًا على التنبؤ بمتوسط سعر المنزلفي أي حي، بالنظر إلى جميع المقاييس الأخرى.

توقع سعر المنزل مع بايثون

آمل أن تكون قد فهمت بيان المشكلة أعلاه حول التنبؤ بأسعار المنازل. الآن، سوف آخذك خلال مشروع التعلم الآلي عن توقع أسعار المنازل باستخدام بايثون. لنبدأ باستيراد مكتبات Python ومجموعة البيانات الضرورية:

مجموعة البيانات

```
import pandas as pd
housing = pd.read csv("housing.csv")
housing.head()
```

```
housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

يمثل كل صف حي وهناك 10 سماتفي مجموعة البيانات. دعنا الآن نستخدم طريقة info() المفيدة للحصول على وصف سريع للبيانات، وخاصة العدد الإجمالي للصفوف ونوع كل سمة وعدد القيم غير الصفرية:

```
housing.info()
```

```
#   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

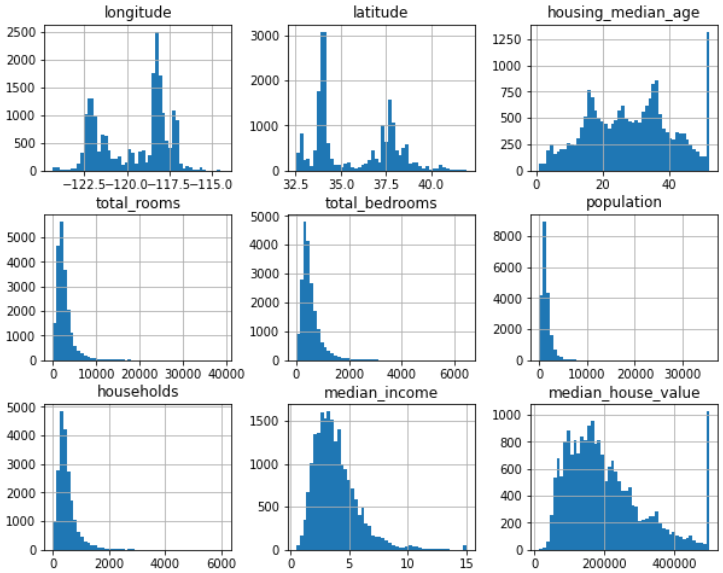
هناك 20،640 مثيلاًفي مجموعة البيانات. لاحظ أن سمة total_bedrooms تحتوي فقط على 20،433 قيمة غير صفرية، مما يعني أن 207 مقاطعة لا تحتوي على قيم. سيتعين علينا التعامل مع ذلك لاحقًا.

جميع السمات رقمية باستثناء حقل ocean_proximity. نوعه هو كائن، لذلك يمكن أن يحتوي على أي نوع من كائنات Python. يمكنك معرفة الفئات الموجودةفي هذا العمود وعدد الدوائر التي تنتمي إلى كل فئة باستخدام طريقة value_counts():

```
housing.ocean_proximity.value_counts()
```

هناك طريقة أخرى سريعة للتعرف على نوع البيانات التي تتعامل معها وهي رسم الهستوكرام لكل سمة عددية:

```
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(10, 8))
plt.show()
```



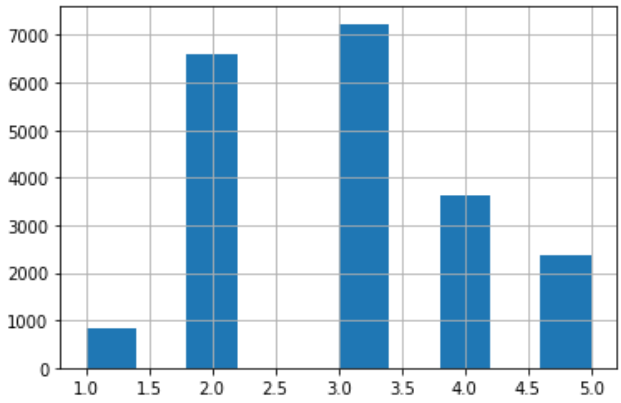
تتمثل الخطوة التاليةفي مهمة توقع أسعار المنازلفي تقسيم البيانات إلى مجموعات تدريب واختبار. يعد إنشاء مجموعة اختبار أمرًا سهلاً من الناحية النظرية: حدد بعض الحالات بشكل عشوائي، عادةً 20٪ من مجموعة البيانات (أو أقل إذا كانت مجموعة البيانات كبيرة جدًا)، وقم بوضعها جانبًا:

```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(housing, test_size=0.2,
random_state=42)
```

دعونا نلقي نظرة فاحصة على الهستوكرام لمتوسط الدخل، حيث تتجمع معظم قيم الدخل المتوسط حول 1.5 إلى 6، لكن بعض متوسط الدخل يتجاوز بكثير 6.

من المهم أن يكون لديك عدد كافٍ من المثيلاتفي مجموعة البيانات الخاصة بك لكل طبقة، وإلا فإن تقدير أهمية الطبقة قد يكون متحيزًا. هذا يعني أنه لا يجب أن يكون لديك عدد كبير جدًا من الطبقات وأن كل طبقة يجب أن تكون كبيرة بما يكفي:

```
import numpy as np
housing['income_cat'] = pd.cut(housing['median_income'], bins=[0.,
1.5, 3.0, 4.5, 6., np.inf], labels=[1, 2, 3, 4, 5])
housing['income_cat'].hist()
plt.show()
```



أخذ العينات الطبقية على مجموعة البيانات

الآن الخطوة التالية هي إجراء بعض العينات الطبقية على مجموعة البيانات. ولكن لماذا نحتاج إلى القيام بذلك يمكنك معرفة كل شيء عنه من هنا. أنت الآن جاهز لأداء أخذ العينات الطبقية stratified sampling على أساس فئة الدخل. لهذا يمكنك استخدام فئة StratifiedShuffleSplit من Scikit-Learn:

```
from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2,
random_state=42)
```

```
for train_index, test_index in split.split(housing,
housing["income_cat"]):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]
print(strat_test_set['income_cat'].value_counts() /
len(strat_test_set))
```

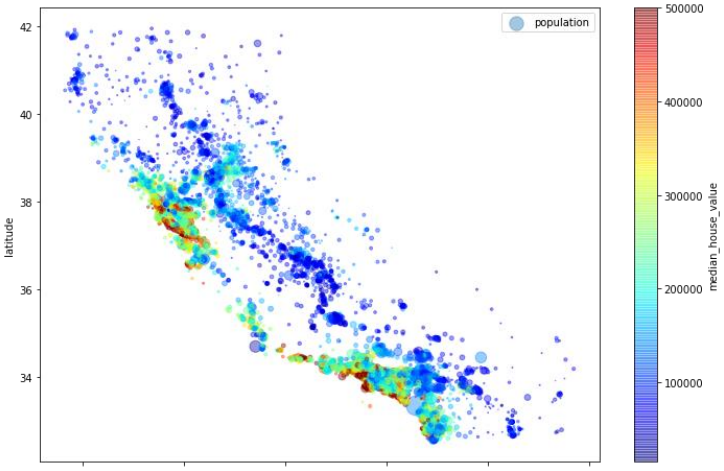
```
3    0.350533
2    0.318798
4    0.176357
5    0.114583
1    0.039729
Name: income_cat, dtype: float64
```

أنت الآن بحاجة إلى إزالة سمة Income_cat التي أضفناها لاستعادة البيانات إلى شكلها:

```
for set_ in (strat_train_set, strat_test_set):
    set_.drop('income_cat', axis=1, inplace=True)
housing = strat_train_set.copy()
```

الآن قبل إنشاء نموذج التعلم الآلي للتنبؤ بأسعار المنزل باستخدام Python، دعنا نرسم البيانات من حيث خطوط الطول والعرض:

```
housing.plot(kind='scatter', x='longitude', y='latitude', alpha=0.4,
s=housing['population']/100, label='population,'
figsize=(12, 8), c='median_house_value', cmap=plt.get_cmap('jet'),
colorbar=True(
plt.legend()
plt.show()
```



يوضح الرسم البياني أسعار المنازلفي كاليفورنيا حيث يكون اللون الأحمر باهظًا والأزرق رخيصًا وتشير الدوائر الأكبر إلى المناطق ذات الكثافة السكانية العالية.

إيجاد الارتباطات

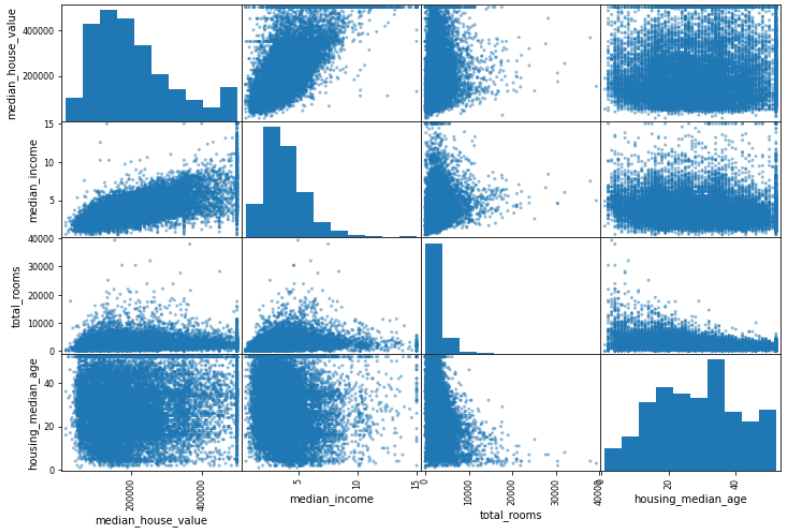
نظرًا لأن مجموعة البيانات ليست كبيرة جدًا، يمكنك بسهولة حساب معامل الارتباط القياسي بين كل زوج من السمات باستخدام طريقة corr():

```
corr_matrix = housing.corr()
print(corr_matrix.median_house_value.sort_values(ascending=False))
```

```
median_house_value    1.000000
median_income         0.687160
total_rooms           0.135097
housing_median_age    0.114110
households            0.064506
total_bedrooms        0.047689
population           -0.026920
longitude            -0.047432
latitude             -0.142724
Name: median_house_value, dtype: float64
```

تتراوح نطاقات الارتباط بين 1- و 1. عندما تكون قريبة من 1 فهذا يعني أن هناك ارتباط إيجابي وعندما تكون قريبة من 1- فهذا يعني أن هناك ارتباط سلبي. عندما يكون قريبًا من 0، فهذا يعني أنه لا يوجد ارتباط خطي.

هناك طريقة أخرى للتحقق من الارتباط بين السمات وهي استخدام دالة pandas scatter_matrix()، التي ترسم كل سمة رقمية مقابل كل سمة رقمية أخرى:



والآن دعونا نلقي نظرة على مصفوفة الارتباط correlation matrix مرة أخرى عن طريق إضافة ثلاثة أعمدة جديدة إلى مجموعة البيانات؛ عدد الغرف لكل أسرة وعدد غرف النوم لكل غرفة وعدد السكان لكل أسرة:

```
housing["rooms_per_household"] =
housing["total_rooms"]/housing["households"]
housing["bedrooms_per_room"] =
housing["total_bedrooms"]/housing["total_rooms"]
```

```
housing["population_per_household"] =
housing["population"]/housing["households"]

corr_matrix = housing.corr()
print(corr_matrix["median_house_value"].sort_values(ascending=False))
```

```
median_house_value          1.000000
median_income               0.687160
rooms_per_household         0.146285
total_rooms                 0.135097
housing_median_age          0.114110
households                  0.064506
total_bedrooms              0.047689
population_per_household   -0.021985
population                 -0.026920
longitude                  -0.047432
latitude                   -0.142724
bedrooms_per_room          -0.259984
Name: median_house_value, dtype: float64
```

تحضير البيانات

الآن، هذه هي أهم خطوة قبل تدريب نموذج التعلم الآلي لمهمة التنبؤ بأسعار المنزل. الآن دعونا نجري جميع عمليات تحويل البيانات الضرورية:

كما ترى، هناك العديد من خطوات تحويل البيانات التي يجب تنفيذها بالترتيب الصحيح. لحسن الحظ، يوفر Scikit-Learn فئة Pipeline لمساعدتكفي مثل هذه التسلسلات من التحويلات. فيما يلي خط أنابيب صغير للسمات الرقمية:

الانحدار الخطي لتوقع أسعار المنازل باستخدام بايثون

سأستخدم الآن خوارزمية الانحدار الخطي لمهمة التنبؤ بسعر المنزل باستخدام Python:

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)

data = housing.iloc[5:]
labels = housing_labels.iloc[5:]
data_preparation = full_pipeline.transform(data)
print("Predictions: ", lin_reg.predict(data_preparation))
```

```
Predictions:  [210644.60459286 317768.80697211 210956.43331178  59218.98886849
 189747.55849879]
```

13) كشف الأخبار الوهمية باستخدام التعلم الآلي Fake News Detection using Machine Learning

الأخبار المزيفة Fake news هي واحدة من أكبر المشكلات التي تواجه وسائل التواصل الاجتماعي عبر الإنترنت وحتى بعض المواقع الإخبارية.في معظم الأوقات، نرى الكثير من الأخبار الكاذبة حول السياسة. لذا فإن استخدام التعلم الآلي لاكتشاف الأخبار الزائفة يُعد مهمة صعبة للغاية. إذا كنت تريد معرفة كيفية اكتشاف الأخبار المزيفة باستخدام التعلم الآلي، فهذه المقالة مناسبة لك.في هذه المقالة، سوف أطلعك على مهمة كشف الأخبار الوهمية باستخدام التعلم الآلي باستخدام python.

كشف الأخبار الكاذبة

الأخبار المزيفة هي واحدة من أكبر المشاكل لأنها تؤدي إلى الكثير من المعلومات الخاطئةفي منطقة معينة.في معظم الأحيان، قد يؤدي نشر أخبار كاذبة حول المعتقدات السياسية والدينية لمجتمع ما إلى أعمال شغب وأعمال عنف كما رأيتفي البلد الذي تعيش فيه. لذلك، للكشف عن الأخبار المزيفة، يمكننا العثور على علاقات بين عناوين الأخبار المزيفة حتى نتمكن من تدريب نموذج للتعلم الآلي يمكنه إخبارنا ما إذا كانت معلومة معينة مزيفة أم حقيقية بمجرد ملاحظة العنوان الرئيسيفي الأخبار. لذلكفي القسم أدناه، سأقدم لكم مشروع التعلم الآلي عن اكتشاف الأخبار المزيفة باستخدام لغة برمجة Python.

كشف الأخبار الوهمية باستخدام بايثون

تحتوي مجموعة البيانات التي أستخدمها هنا لمهمة الكشف عن الأخبار المزيفة على بيانات حول عنوان الأخبار ومحتوى الأخبار وعمود يُعرف باسم التصنيف يوضح ما إذا كانت الأخبار مزيفة أم حقيقية. لذلك يمكننا استخدام مجموعة البيانات هذه للعثور على العلاقات بين عناوين الأخبار الزائفة والحقيقية لفهم نوع العناوين الرئيسيةفي معظم الأخبار المزيفة. لذلك دعونا نستورد مكتبات Python الضرورية ومجموعة البيانات التي نحتاجها لهذه المهمة:

مجموعة البيانات

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB

data = pd.read_csv("news.csv")
print(data.head())
```

```
   Unnamed: 0  ... label
0        8476  ...  FAKE
1       10294  ...  FAKE
2        3608  ...  REAL
3       10142  ...  FAKE
4         875  ...  REAL
```

مجموعة البيانات هذه كبيرة جدًا ولحسن الحظ لا تزال لا تحتوي على قيم مفقودة، لذا دون إضاعة أي وقت، دعنا نستخدم عمود العنوان title column كميزة نحتاجها لتدريب نموذج التعلم الآلي وعمود التسمية label column على أنهما القيم التي نريد توقعها:

```
x = np.array(data["title"])
y = np.array(data["label"])

cv = CountVectorizer()
x = cv.fit_transform(x)
```

دعنا الآن نفصل مجموعة البيانات إلى مجموعات تدريب واختبار، وبعد ذلك سأستخدم خوارزمية Multinomial Naive Bayes لتدريب نموذج اكتشاف الأخبار المزيفة:

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2,
random_state=42)
model = MultinomialNB()
model.fit(xtrain, ytrain)
print(model.score(xtest, ytest))
```

```
0.8074191002367798
```

الآن دعونا نختبر هذا النموذج. لاختبار نموذجنا المدرّب، سأقوم أولاً بتدوين عنوان أي خبر موجودفي أخبار google لمعرفة ما إذا كان نموذجنا يتوقع أن الأخبار حقيقية أم لا:

```
news_headline = "CA Exams 2021: Supreme Court asks ICAI to extend opt-
out option for July exams, final order tomorrow"
data = cv.transform([news_headline]).toarray()
print(model.predict(data))
```

```
['REAL']
```

سأقوم الآن بكتابة عنوان أخبار مزيف عشوائي لمعرفة ما إذا كان النموذج يتنبأ بأن الأخبار مزيفة أم لا:

```
news_headline = "Cow dung can cure Corona Virus"
data = cv.transform([news_headline]).toarray()
print(model.predict(data))
```

```
['FAKE']
```

الملخص

هذه هي الطريقة التي يمكننا بها تدريب نموذج التعلم الآلي لمهمة الكشف عن الأخبار المزيفة باستخدام لغة برمجة Python. الأخبار المزيفة هي واحدة من أكبر المشاكل لأنها تؤدي إلى الكثير من المعلومات الخاطئةفي منطقة معينة. أتمنى أن تكون قد أحببت هذه المقالة حول مهمة الكشف عن الأخبار المزيفة باستخدام التعلم الآلي باستخدام Python.

14) الكشف عن سرطان البروستاتا باستخدام التعلم الآلي Prostate Cancer Detection using Machine Learning

استيراد مكتبات الضرورية

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import accuracy_score
```

```
Cancer = pd.read_csv('https://cainvas-
static.s3.amazonaws.com/media/user_data/cainvas-
admin/Prostate_Cancer_Data-_CSV.csv')
```

```
Cancer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 100 non-null    int64
 1   diagnosis_result   100 non-null    object
 2   radius             100 non-null    int64
 3   texture            100 non-null    int64
 4   perimeter          100 non-null    int64
 5   area               100 non-null    int64
 6   smoothness         100 non-null    float64
 7   compactness        100 non-null    float64
 8   symmetry           100 non-null    float64
 9   fractal_dimension  100 non-null    float64
dtypes: float64(4), int64(5), object(1)
memory usage: 7.9+ KB
```

```
Cancer.head(10)
```

	id	diagnosis_result	radius	texture	perimeter	area	smoothness	compactness	symmetry	fractal_dimension
0	1	M	23	12	151	954	0.143	0.278	0.242	0.079
1	2	B	9	13	133	1326	0.143	0.079	0.181	0.057
2	3	M	21	27	130	1203	0.125	0.160	0.207	0.060
3	4	M	14	16	78	386	0.070	0.284	0.260	0.097
4	5	M	9	19	135	1297	0.141	0.133	0.181	0.059
5	6	B	25	25	83	477	0.128	0.170	0.209	0.076
6	7	M	16	26	120	1040	0.095	0.109	0.179	0.057
7	8	M	15	18	90	578	0.119	0.165	0.220	0.075
8	9	M	19	24	88	520	0.127	0.193	0.235	0.074
9	10	M	25	11	84	476	0.119	0.240	0.203	0.082

```
Cancer.tail()
```

	id	diagnosis_result	radius	texture	perimeter	area	smoothness	compactness	symmetry	fractal_dimension
95	96	M	23	16	132	1264	0.091	0.131	0.210	0.056
96	97	B	22	14	78	451	0.105	0.071	0.190	0.066
97	98	B	19	27	62	295	0.102	0.053	0.135	0.069
98	99	B	21	24	74	413	0.090	0.075	0.162	0.066
99	100	M	16	27	94	643	0.098	0.114	0.188	0.064

```
Cancer.describe()
```

	id	radius	texture	perimeter	area	smoothness	compactness	symmetry	fractal_dimension
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
mean	50.500000	16.850000	18.230000	96.780000	702.880000	0.102730	0.126700	0.193170	0.064690
std	29.011492	4.879094	5.192954	23.676089	319.710895	0.014642	0.061144	0.030785	0.008151
min	1.000000	9.000000	11.000000	52.000000	202.000000	0.070000	0.038000	0.135000	0.053000
25%	25.750000	12.000000	14.000000	82.500000	476.750000	0.093500	0.080500	0.172000	0.059000
50%	50.500000	17.000000	17.500000	94.000000	644.000000	0.102000	0.118500	0.190000	0.063000
75%	75.250000	21.000000	22.250000	114.250000	917.000000	0.112000	0.157000	0.209000	0.069000
max	100.000000	25.000000	27.000000	172.000000	1878.000000	0.143000	0.345000	0.304000	0.097000

```
Cancer.columns
```

```
Index(['id', 'diagnosis_result', 'radius', 'texture', 'perimeter', 'area',
       'smoothness', 'compactness', 'symmetry', 'fractal_dimension'],
      dtype='object')
```

- نحن لا نهتم بمعرف الأعمدة. لذا، نحن نسقط ذلك!

```
# We don't care id of the columns. So, we drop that!
Cancer.drop(['id'],axis=1,inplace=True)
```

```
Cancer.head()
```

	diagnosis_result	radius	texture	perimeter	area	smoothness	compactness	symmetry	fractal_dimension
0	M	23	12	151	954	0.143	0.278	0.242	0.079
1	B	9	13	133	1326	0.143	0.079	0.181	0.057
2	M	21	27	130	1203	0.125	0.160	0.207	0.060
3	M	14	16	78	386	0.070	0.284	0.260	0.097
4	M	9	19	135	1297	0.141	0.133	0.181	0.059

تحليل البيانات استكشافية

- Diagnosis_result هو العمود الأكثر أهمية بالنسبة لنا. لأننا سنصنف البيانات تعتمد على هذا العمود.
- لدينا الأعداد الصحيحة من أجل التصنيف. لذلك ، يجب علينا تحويلها من كائن إلى عدد صحيح.

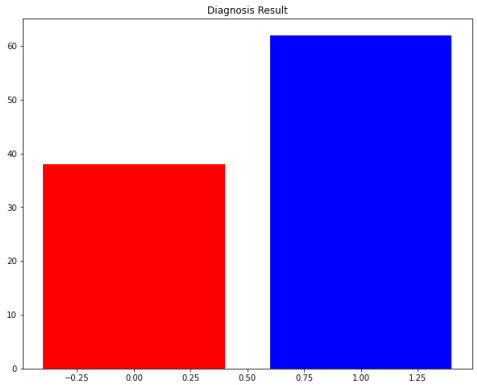
```
Cancer.diagnosis_result = [1 if each == 'M' else 0 for each in
Cancer.diagnosis_result]
```

اختبار النموذج

```
# Let's check it.
Cancer.diagnosis_result.value_counts()
```

```
1    62
0    38
Name: diagnosis_result, dtype: int64
```

```
plt.figure(figsize=(10,8))
plt.bar(list(Cancer['diagnosis_result'].value_counts().index),
Cancer['diagnosis result'].value counts(), color = ['b','r'])
plt.title('Diagnosis Result')
plt.show()
print(Cancer['diagnosis_result'].value_counts())
```




```
1    62
0    38
Name: diagnosis_result, dtype: int64
```

مجموعة بيانات التدريب والاختبار

يجب علينا تعيين قيم x و y لتقسيم بيانات مجموعة التدريب_الاختبار.

```
y = Cancer.diagnosis_result.values
x_data = Cancer.drop(['diagnosis_result'],axis=1)
```

انظر إلى قيمنا:

```
y
```

```
array([1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1])
```

```
x_data.head()
```

	radius	texture	perimeter	area	smoothness	compactness	symmetry	fractal_dimension
0	23	12	151	954	0.143	0.278	0.242	0.079
1	9	13	133	1326	0.143	0.079	0.181	0.057
2	21	27	130	1203	0.125	0.160	0.207	0.060
3	14	16	78	386	0.070	0.284	0.260	0.097
4	9	19	135	1297	0.141	0.133	0.181	0.059

تسوية البيانات

يعني التسوية Normalization جعل جميع قيم البيانات يتراوح مقياسها بين 0 و 1.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
x = scaler.fit_transform(x_data)
```

```
x
```

```
array([[0.875     , 0.0625    , 0.825     , 0.44868735, 1.        ,
        0.78175896, 0.63313609, 0.59090909],
       [0.        , 0.125     , 0.675     , 0.67064439, 1.        ,
        0.13355049, 0.27218935, 0.09090909],
       [0.75      , 1.        , 0.65      , 0.59725537, 0.75342466,
        0.39739414, 0.4260355 , 0.15909091],
       [0.3125    , 0.3125    , 0.21666667, 0.1097852 , 0.        ,
        0.80130293, 0.73964497, 1.        ],
       [0.        , 0.5       , 0.69166667, 0.65334129, 0.97260274,
        0.30944625, 0.27218935, 0.13636364],
       [1.        , 0.875     , 0.25833333, 0.16408115, 0.79452055,
        0.42996743, 0.43786982, 0.52272727],
       [0.4375    , 0.9375    , 0.56666667, 0.5       , 0.34246575,
        0.23127036, 0.26035503, 0.09090909],
       [0.375     , 0.4375    , 0.31666667, 0.22434368, 0.67123288,
        0.41368078, 0.50295858, 0.5       ],
       [0.625     , 0.8125    , 0.3       , 0.18973747, 0.78082192,
        0.50488599, 0.59171598, 0.47727273],
       [1.        , 0.        , 0.26666667, 0.16348449, 0.67123288,
        0.65798046, 0.40236686, 0.65909091],
       [0.9375    , 0.625     , 0.425     , 0.35560859, 0.16438356,
        0.09446254, 0.10650888, 0.09090909],
       [0.5       , 0.25      , 0.43333333, 0.34546539, 0.36986301,
        0.29641694, 0.28994083, 0.18181818],
```

تدريب واختبار مجموعة البيانات المنقسمة

نحن على استعداد لتقسيم البيانات كتدريب واختبار.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size=0.2,random_state=42)
#%40 data will assign as 'Test Datas'
method_names=[] # In Conclusion part, I'll try to show you which
method gave the best result.
method_scores=[]
```

لنلقِ نظرة على القيم الجديدة.

```
x_train
```

```
array([[0.5625    , 0.125     , 0.175     , 0.12350835, 0.34246575,
        0.05537459, 0.33727811, 0.13636364],
       [0.125     , 0.        , 0.23333333, 0.1575179 , 0.24657534,
        0.18241042, 0.34319527, 0.25      ],
       [0.0625    , 0.8125    , 0.375     , 0.26431981, 0.47945205,
        0.48534202, 0.53254438, 0.36363636],
       [0.125     , 0.        , 0.63333333, 0.53818616, 0.28767123,
        0.58957655, 0.56804734, 0.22727273],
       [0.5       , 0.625     , 0.24166667, 0.17959427, 0.38356164,
        0.04560261, 0.14201183, 0.09090909],
       [0.8125    , 0.5       , 0.375     , 0.27267303, 0.60273973,
        0.39739414, 0.56213018, 0.40909091],
       [0.9375    , 0.3125    , 0.28333333, 0.21539379, 0.16438356,
        0.07166124, 0.25443787, 0.06818182],
       [0.8125    , 0.1875    , 0.21666667, 0.14856802, 0.47945205,
        0.10749186, 0.32544379, 0.29545455],
       [1.        , 0.        , 0.26666667, 0.16348449, 0.67123288,
        0.65798046, 0.40236686, 0.65909091],
       [0.75      , 0.0625    , 0.51666667, 0.43377088, 0.50684932,
        0.4723127 , 0.34319527, 0.27272727],
       [0.5       , 0.25      , 0.43333333, 0.34546539, 0.36986301,
        0.29641694, 0.28994083, 0.18181818],
```

النموذج باستخدام ANN

```
model = Sequential()
model.add(Dense(32,activation = 'relu',input_dim = x_train.shape[1]))
model.add(Dense(64,activation = 'relu'))
model.add(Dense(1,activation = 'sigmoid'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 32)                288
_________________________________________________________________
dense_4 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 65
=================================================================
Total params: 2,465
Trainable params: 2,465
Non-trainable params: 0
```

```
model.compile(optimizer = 'adam', loss='binary_crossentropy',
metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs = 120,
validation_data=(x_test,y_test))
```

```
Epoch 114/120
3/3 [==============================] - 0s 10ms/step - loss: 0.2234 - accuracy: 0.9250 - val_loss: 0.5366 - val_accuracy: 0.8500
Epoch 115/120
3/3 [==============================] - 0s 11ms/step - loss: 0.2227 - accuracy: 0.9250 - val_loss: 0.5393 - val_accuracy: 0.8500
Epoch 116/120
3/3 [==============================] - 0s 45ms/step - loss: 0.2210 - accuracy: 0.9250 - val_loss: 0.5431 - val_accuracy: 0.8500
Epoch 117/120
3/3 [==============================] - 0s 67ms/step - loss: 0.2187 - accuracy: 0.9375 - val_loss: 0.5509 - val_accuracy: 0.8000
Epoch 118/120
3/3 [==============================] - 0s 41ms/step - loss: 0.2233 - accuracy: 0.9125 - val_loss: 0.5656 - val_accuracy: 0.8000
Epoch 119/120
3/3 [==============================] - 0s 35ms/step - loss: 0.2214 - accuracy: 0.9000 - val_loss: 0.5635 - val_accuracy: 0.8000
Epoch 120/120
3/3 [==============================] - 0s 42ms/step - loss: 0.2222 - accuracy: 0.9125 - val_loss: 0.5548 - val_accuracy: 0.8000
```

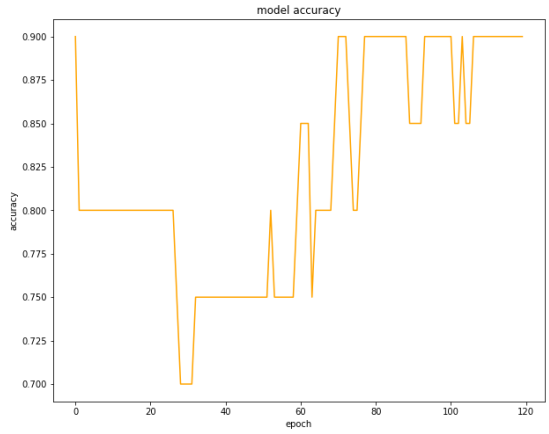
```
method_names.append("ANN")
method_scores.append(0.851)
```

```
trainX = np.reshape(x_train, (x_train.shape[0], x_train.shape[1],1))
testX = np.reshape(x_test, (x_test.shape[0],x_test.shape[1],1))
# Print and check shapes
print("Shape of trainX is {}".format(trainX.shape))
print("Shape of testX is {}".format(testX.shape))
```

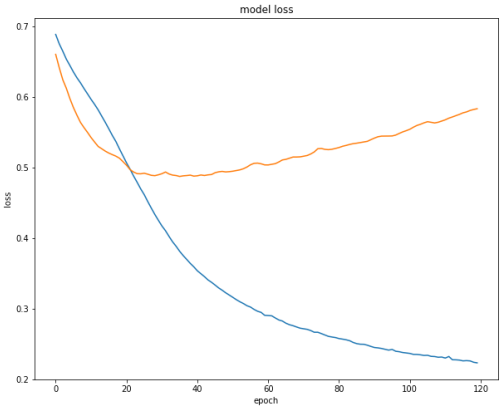
```
Shape of trainX is (80, 8, 1)
Shape of testX is (20, 8, 1)
```

الرسم البياني للدقة والخطأ

```
plt.figure(figsize=(10,8))
#plt.plot(history.history[''])
plt.plot(history.history['val accuracy'],color='orange')
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```




```
plt.figure(figsize=(10,8))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```




```
y_pred = model.predict_classes(x_test)
y_pred
```

```
array([[1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [0],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [0],
       [1],
       [1],
       [1]], dtype=int32)
```

```
y_test.shape
```

```
(20,)
```

```
y_pred = np.squeeze(y_pred)
y_pred.shape
```

```
(20,)
```

```
print('Test Accuracy : ',accuracy_score(y_test, y_pred))
```

```
Test Accuracy :  0.9
```

حفظ النموذج

```
model.save('prostatecancer.h5')
```

15) التنبؤ بأمراض القلب والأوعية الدموية باستخدام التعلم الآلي
Predicting Cardiovascular Disease Using Machine Learning

وفقًا لويكيبيديا، يعتبر القلب والأوعية الدموية السبب الرئيسي للوفاة على مستوى العالم. إنه مزيج من القلب والأوعية الدموية المختلفة مثل أمراض القلب والنوبات القلبية والسكتة الدماغية وفشل القلب وعدم انتظام ضربات القلب ومشاكل صمام القلب وما إلى ذلك. يعتبر ارتفاع ضغط الدم وارتفاع الكوليسترول والسكري والخمول البدني من الأسباب الرئيسية لزيادة المخاطر للإصابة بهذا المرض. من خلال تقليل عوامل الخطر السلوكية مثل التدخين والنظام الغذائي غير الصحي وتعاطي الكحول وقلة النشاط البدني، يمكن منع هذا المرض.

إذا كان بإمكان الناس أن يكونوا على دراية مسبقة بهذا المرض قبل أن يتحول إلى مستوى أكثر خطورة، فيمكننا تقليل عدد الوفيات والمرضى المعرضين للخطر بدرجة كبيرة. بمساعدة التطوير في التعلم الآلي والقوة الحاسوبية العالية، أدت إلى تقدم أسيفي الذكاء الاصطناعيفي مجال الطب، حيث يمكن للناس استخدام هذه التقنيات والتوصل إلى نموذج والقيام بالتنبؤات لتحديد احتمالية إصابة الأشخاص بهذا المرضفي المراحل المبكرة.

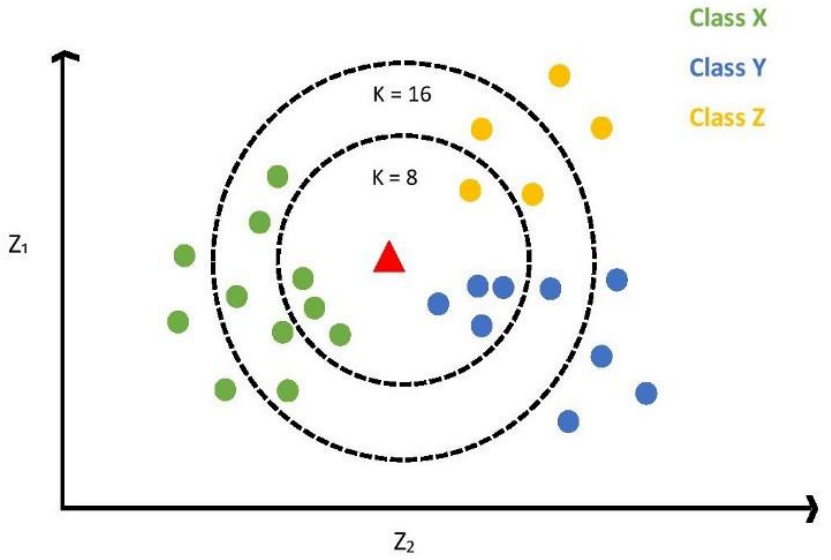
في هذه المقالة، تم اقتراح نموذج التعلم الآلي وتنفيذه لتحديد احتمالية إصابة الشخص بهذا المرض أم لا من خلال التركيز على عوامل مثل المعلومات الواقعية ونتائج الفحوصات الطبية ومعلومات المريض التي تم جمعها من مجموعة بيانات عبر الإنترنت. تم استخدام خوارزمية K Nearest Neighbours وهي خوارزمية تصنيف معروفة وجيدة الأداء لتنفيذ هذا النموذج.

اختيار الخوارزمية

K Nearest Neighbours هي خوارزمية بسيطة ولكنها تعمل بشكل لا يصدقفي الممارسة حيث تخزن جميع الحالات المتاحة وتصنف البيانات أو الحالة الجديدة بناءً على مقياس التشابه. يقترح أنه إذا كانت النقطة الجديدة المضافة إلى العينة مشابهة لنقاط الجوار، فإن هذه النقطة ستنتمي إلى فئة معينة من نقاط الجوار. بشكل عام، تستخدم خوارزمية KNNفي تطبيقات البحث حيث يبحث الأشخاص عن عناصر مماثلة. يشير Kفي خوارزمية KNN إلى عدد أقرب جيران للنقطة الجديدة التي يجب توقعها.

تُعرف خوارزمية KNN أيضًا باسم المتعلم الكسول نظرًا لوجود مرحلة تعلم أقلفي النموذج نظرًا لقدرته على التعلم السريع جدًا. بدلاً من ذلك، يحفظ مجموعة بيانات التدريب ويحدث كل العملفي وقت طلب التنبؤ.

كيف تعمل خوارزمية KNN؟



عندما نضيف نقطة جديدة إلى مجموعة بيانات باستخدام خوارزمية KNN، يمكننا التنبؤ بالفئة التي تنتمي إليها النقطة الجديدة. من أجل بدء التنبؤ، فإن أول شيء يتعين علينا القيام به هو تحديد قيمة K. وفقًا للشكل 1، تنتمي النقاط ذات اللون الأخضر إلى الفئة X، والنقاط ذات اللون الأزرق تنتمي إلى الفئة Y بينما تنتمي مكاييل اللون الأصفر إلى فئة Z. عندما K = 8، نحتاج إلى تحديد 8 نقاط مجاورة لها أقل مسافة للنقطة الجديدة التي يمثلها المثلث. كما هو موضح في الشكل 1 عندما تكون K = 8 نقطة جديدة قريبة من نقطة صفراء واحدة وثلاث نقاط خضراء وأربع نقاط زرقاء. نظرًا لأن لدينا غالبية النقاط الزرقاء، في هذه الحالة، يمكننا القول إنه بالنسبة إلى K = 8، فإن النقطة الجديدة تنتمي إلى الفئة Y.

للمضي قدمًا إذا كانت K = 16، يتعين علينا البحث عن 16 نقطة مختلفة أقرب إلى النقاط الجديدة. بعد حساب المسافة، وجد أنه عندما تكون K = 16، تكون النقطة الجديدة أقرب إلى ثلاث نقاط صفراء، وخمس نقاط زرقاء، وثماني نقاط خضراء. لذلك، يمكننا القول إنه عندما تكون K = 16، فإن النقطة الجديدة تنتمي إلى الفئة X.

من أجل العثور على أفضل قيمة K، يمكننا استخدام تقنية التحقق المتبادل cross-validation لاختبار عدة قيم لـ K. وسأوضح لك كيفية استخدام تقنية التحقق المتبادل للعثور على أفضل قيمة K في هذه المقالة. لإيجاد أقل مسافة بين نقاط الجوار، يمكننا استخدام المسافة الإقليدية أو

مسافة مانهاتن.في المسافة الإقليدية، ستأخذ المسافة المستقيمة بين نقطتينفي مساحة إقليدية بينما تحسب مسافة مانهاتن المسافة بين المتجهات الحقيقية باستخدام مجموع الفرق المطلق.

جمع البيانات

من أجل التنبؤ بما إذاكان الشخص يعاني من أمراض القلب والأوعية الدموية، تم اختيار مجموعة بيانات من Kaggale.com. تتكون مجموعة البيانات هذه من ثلاثة أنواع من البيانات على التوالي، والمعلومات الواقعية، ونتائج الفحص الطبي (ميزة الفحص)، والمعلومات التي يقدمها المرضى (السمات الشخصية). أيضًا، يمكن تقسيم البيانات الموجودةفي مجموعة البيانات إلى بيانات فئوية وبيانات رقمية. تتألف مجموعة البيانات الأصلية من 70000 مثيل بيانات و14 ميزة على النحو التاليفي الجدول التالي.

	Feature	Description
Categorical Data	age_days	| Factual Information | age in days | int (days)
	age_year	| Factual Information | age in years | int (days)
	height	| Factual Information | height | int (cm) |
	weight	| Factual Information | weight | float (kg) |
	ap_hi	| Systolic blood pressure | Examination Feature | int |
	ap_lo	|Diastolic blood pressure | Examination Featur | int |
Numerical Data	gender	| Factual Information |2:male, 1:female|
	cholesterol	Cholesterol | Examination Feature | cholesterol | 1: normal, 2: above normal, 3: well above normal |
	gluc	Glucose | Examination Feature | gluc | 1: normal, 2: above normal, 3: well above normal |
	smoke	Smoking |Subjective Feature | smoke | binary |
	alco	Alcohol intake | Subjective Feature | alco | binary |
	active	physical activity | Subjective Feature | active | binary |
	cardio	Presence or absence of cardiovascular disease | **Target Variable** | cardio | binary |
	id	| Factual Information |

تنفيذ النموذج

استيراد مكتبات

كخطوة أولى، تم استيراد جميع المكتبات المطلوبة للحساب عالي الأداء، ورسم البيانات، وتحليل نموذج البيانات على النحو التالي.

```
import pandas as pd
pd.options.mode.chained_assignment = None
import matplotlib.pyplot as pt
from matplotlib import rcParams
import seaborn as sns
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import cross_val_score
import sklearn.metrics as met
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

استيراد مجموعة البيانات

تم استيراد مجموعة عبر الإنترنت لأمراض القلب والأوعية الدموية كملف CSV لاجراء التحليل على النحو التالي البيانات. وفقًا للأهمية المنخفضة لميزة "id"، تمت إزالتها من مجموعة البيانات واستيراد مجموعة البيانات المتبقية.

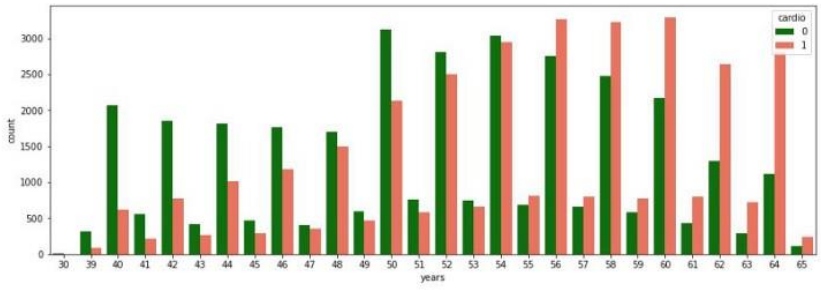
```
data=pd.read_csv('data.csv').drop(["id"],axis=1)
```

العرض المرئي للمعلومات

تم إجراء تمثيل رسومي لمقارنة العلاقة بين الفئات العمرية وأمراض القلب والأوعية الدموية.

```
rcParams['figure.figsize'] = 15, 5
data['years'] = (data['age_year']).round().astype('int')
sns.countplot(x='years', hue='cardio', data = data,
palette=["#008000", "#FF6347"]);
```

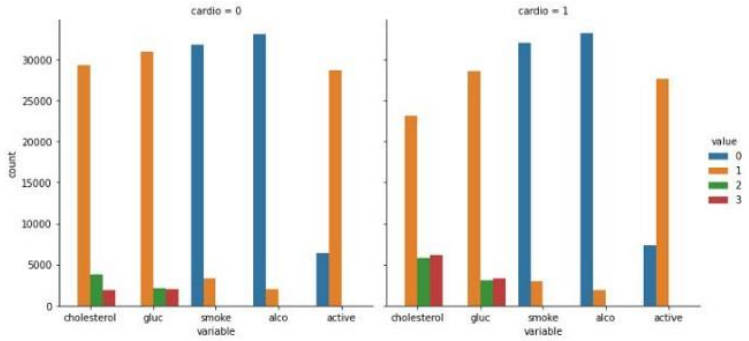
ستنتج هذه الشفرة البرمجية مخططًا شريطيًا يمثل المحور السيني العمر بالسنوات بينما يمثل المحور الصادي عدد الأشخاص. كما هو موضح في الشكل ادناه، يمثل اللون الأحمر الأشخاص الذين يعانون من أمراض القلب والأوعية الدموية بينما يمثل اللون الأخضر الأشخاص الذين لا يعانون من هذا المرض.



وفقًا للرسم البياني، يمكن ملاحظة أن الأشخاص الذين تتراوح أعمارهم بين 56 و60 عامًا أكثر تعرضًا للمرض.

علاوة على ذلك، تم إجراء تحليل مرئي لتوزيع البيانات الفئوية من مقتطف الكود أدناه.

```
df_double = pd.melt(data, id_vars=['cardio'],
value_vars=['cholesterol','gluc', 'smoke', 'alco', 'active'])
sns.catplot(x="variable", hue="value", col="cardio",data=df_double,
kind="count");
```



كما هو موضحفي الشكل اعلاه، يمثل المخطط الشريطي الأول توزيع البيانات الفئوية بين الأشخاص المصابين بأمراض القلب والأوعية الدموية بينما يمثل المخطط الشريطي الثاني توزيع البيانات الفئوية بين الأشخاص الذين لا يعانون من المرض. يُظهر التحليل ثنائي المتغير أعلاه أن الأشخاص الذين يعانون من أمراض القلب والأوعية الدموية لديهم مستوى كوليسترول وجلوكوز أعلى من غيرهم.

معالجة البيانات

اختيار الميزات

كما ذكرنا سابقًا نظرًا للأهمية المنخفضة لميزة "id"، فقد تمت إزالتها من مجموعة البيانات. ومع ذلك، تمت إضافة ميزة جديدة تسمى مؤشر كتلة الجسم "bmi" كميزة مشتقة من السمتين الموجودتين "الطول height" و "الوزن weight"، حيث أن قيمة مؤشر كتلة الجسم لها تأثير كبير على أمراض القلب والأوعية الدموية.

```
data["bmi"] = data["weight"] / (data["height"]/100)**2
```

بعد تحديد الميزة، تكون مجموعة الميزات النهائية كما يليفي الشكل ادناه.

	count	mean	std	min	25%	50%	75%	max
age_days	70000.0	19468.865814	2467.251667	10798.000000	17664.000000	19703.000000	21327.000000	23713.000000
age_year	70000.0	53.339358	6.759594	29.583562	48.394521	53.980822	58.430137	64.967123
gender	70000.0	1.349571	0.476838	1.000000	1.000000	1.000000	2.000000	2.000000
height	70000.0	164.359229	8.210126	55.000000	159.000000	165.000000	170.000000	250.000000
weight	70000.0	74.205690	14.395757	10.000000	65.000000	72.000000	82.000000	200.000000
ap_hi	70000.0	128.817286	154.011419	-150.000000	120.000000	120.000000	140.000000	16020.000000
ap_lo	70000.0	96.630414	188.472530	-70.000000	80.000000	80.000000	90.000000	11000.000000
cholesterol	70000.0	1.366871	0.680250	1.000000	1.000000	1.000000	2.000000	3.000000
gluc	70000.0	1.226457	0.572270	1.000000	1.000000	1.000000	1.000000	3.000000
smoke	70000.0	0.088129	0.283484	0.000000	0.000000	0.000000	0.000000	1.000000
alco	70000.0	0.053771	0.225568	0.000000	0.000000	0.000000	0.000000	1.000000
active	70000.0	0.803729	0.397179	0.000000	1.000000	1.000000	1.000000	1.000000
cardio	70000.0	0.499700	0.500003	0.000000	0.000000	0.000000	1.000000	1.000000
bmi	70000.0	27.556513	6.091511	3.471784	23.875115	26.374068	30.222222	298.666667
years	70000.0	53.338686	6.765294	30.000000	48.000000	54.000000	58.000000	65.000000

التحقق من القيم الخالية

باستخدام دالة ()isnull التي تقدمها pandas، يمكن فحص إطار البيانات بالكامل لتحديد القيم المفقودة أو قيم NAN. كما تظهر النتيجةفي الشكل اعلاه، فإنها تعطي خطأ من خلال التأكيد على عدم وجود قيم مفقودةفي مجموعة البيانات.

```
data.isnull().values.any()
```

```
Out[14]: False
```

تنظيف البيانات

من أجل التوصل إلى مجموعة بيانات دقيقة، تم الكشف عن السجلات الفاسدة أو غير المرغوب فيها وإزالتها عن طريق تنظيف البيانات data cleansing. كما تريفي الشكل السابق، يبلغ الحد الأقصى للارتفاع 250 سم والحد الأقصى للوزن 200 كجم والحد الأقصى لقيمة مؤشر كتلة الجسم bmi المشتق من الطول والوزن هو 298 والتي لها قيم غير ملائمة عند النظر إليها ومقارنتها ببعضها البعض. لذلك، تمت إزالة البيانات غير ذات الصلة وتعميم مجموعة البيانات عن طريق إزالة القيم المتطرفة outliers على النحو التالي.

```
data.drop(data[(data['height'] > data['height'].quantile(0.975)) |
(data['height'] < data['height'].quantile(0.025))].index,inplace=True)
data.drop(data[(data['weight'] > data['weight'].quantile(0.975)) |
(data['weight'] < data['weight'].quantile(0.025))].index,inplace=True)
```

علاوة على ذلك، لا يمكن أن يتجاوز ضغط الدم الانبساطي (ap_lo) ضغط الدم الانقباضي (ap_hi) لأن الضغط الانقباضي هو أقصى ضغط يمارسه القلب أثناء النبض والضغط الانبساطي هو مقدار الضغطفي الشرايين بين النبضات. كما أن الفرق العددي بين ضغط الدم الانقباضي وضغط الدم الانبساطي المعروف بضغط الدم ولا يمكن أن يكون ذا قيمة سالبة. من خلال النظر في هذه الحقائق المتطرفة من ap_hi و ap_lo تمت إزالتها للتخلص من بيانات ضغط الدم غير الدقيقة.

```
data.drop(data[(data['ap_hi'] > data['ap_hi'].quantile(0.975)) |
(data['ap_hi'] < data['ap_hi'].quantile(0.025))].index,inplace=True)
data.drop(data[(data['ap_lo'] > data['ap_lo'].quantile(0.975)) |
(data['ap_lo'] < data['ap_lo'].quantile(0.025))].index,inplace=True)
```

بعد عملية تنظيف البيانات، يمكننا أن نرى مجموعة بيانات محدثة مع عدد جديد منخفض يساوي 63866 مقدار مجموعة البيانات، وتغيرت قيم الحد الأدنى والحد الأقصى للطول والوزن وap_hi و ap_lo كمافي الشكل التالي.

	count	mean	std	min	25%	50%	75%	max
age_days	63866.0	19472.641171	2461.983315	10798.000000	17679.250000	19705.000000	21323.000000	23713.000000
age_year	63866.0	53.349702	6.745160	29.583562	48.436301	53.986301	58.419178	64.967123
gender	63866.0	1.347806	0.476278	1.000000	1.000000	1.000000	2.000000	2.000000
height	63866.0	164.497855	6.862322	150.000000	160.000000	165.000000	169.000000	180.000000
weight	63866.0	73.543564	11.720806	52.000000	65.000000	72.000000	81.000000	106.000000
ap_hi	63866.0	128.815442	160.987785	-150.000000	120.000000	120.000000	140.000000	16020.000000
ap_lo	63866.0	95.953308	186.287388	-70.000000	80.000000	80.000000	90.000000	11000.000000
cholesterol	63866.0	1.359049	0.674782	1.000000	1.000000	1.000000	1.000000	3.000000
gluc	63866.0	1.222654	0.568902	1.000000	1.000000	1.000000	1.000000	3.000000
smoke	63866.0	0.086353	0.280886	0.000000	0.000000	0.000000	0.000000	1.000000
alco	63866.0	0.052876	0.223788	0.000000	0.000000	0.000000	0.000000	1.000000
active	63866.0	0.803683	0.397214	0.000000	1.000000	1.000000	1.000000	1.000000
cardio	63866.0	0.498199	0.500001	0.000000	0.000000	0.000000	1.000000	1.000000
bmi	63866.0	27.232986	4.446555	16.049383	23.875433	26.298488	29.823253	46.666667
years	63866.0	53.349450	6.750712	30.000000	48.000000	54.000000	58.000000	65.000000

توحيد البيانات

من أجل التأكد من أن البيانات متسقة داخليًا بحيث يمكن مقارنة البيانات بسهولة مع بعضها البعض، تم إجراء عملية توحيد البيانات data standardization على البيانات الرقمية فقطفي مجموعة البيانات.

```
standardScaler = StandardScaler()
scale_columns = ['age_year', 'height', 'weight', 'ap_hi',
'ap_lo','bmi']
data[scale_columns] =
standardScaler.fit_transform(data[scale_columns])
```

تقسيم مجموعة البيانات

تم إجراء تقسيم البيانات وفق نهجين رئيسيين:

1. تقسيم مجموعة البيانات إلى ميزات وتسميات.
   - التسميات: الاختيار النهائي أو النتيجة التي يجب توقعها.
   - الميزات: تستخدم السمات للتنبؤ بالتسميات.

```
Y = data['cardio']
X = data.drop(['cardio'], axis = 1)
```

2. تقسيم مجموعة البيانات إلى مجموعات بيانات تدريب واختبار.
   - مجموعة بيانات التدريب: عينة من البيانات المستخدمة لملاءمة النموذج
   - مجموعة بيانات الاختبار: مجموعة عينات من البيانات المستخدمة لتقديم تقييم على النموذج النهائي.

```
x_training, x_testing, y_training, y_testing = train_test_split(X, Y,
test_size = 0.33, random_state = 0)
```

بناء مصنف K أقرب الجيران

في البداية، نظرًا لأننا لا نعرف أفضل قيمة K للنموذج هنا، يمكننا إعطاء قيمة "n_neighbours" كـ 1. ثم سيتم تثبيت النموذج بواسطة بيانات التدريب.

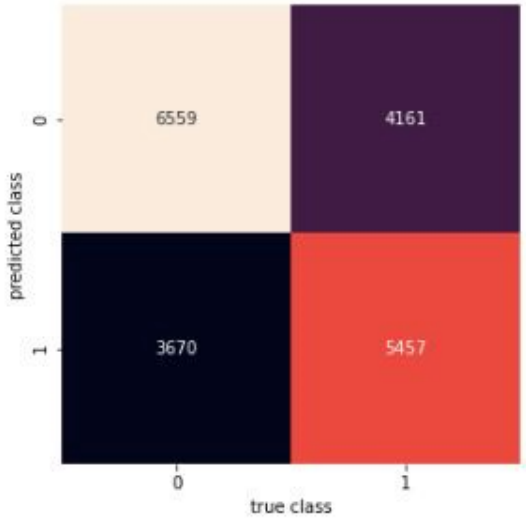
```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(x_training,y_training)
```

ثم يمكننا القيام بالتنبؤ على النحو التالي.

```
prediction = knn.predict(x_testing)
```

من أجل الحصول على مزيد من الفهم حول أداء نموذج التصنيف، تم استخدام مصفوفة الارتباك كما هو موضح في الشكل التالي.

```
conmat=confusion_matrix(y_testing,prediction)
sns.heatmap(conmat.T, square=True, annot=True, fmt='d', cbar=False)
pt.xlabel('true class')
pt.ylabel('predicted class')
```



من مصفوفة الارتباك، ستعطي تقرير التصنيف الممثل كمافي الشكل 8 بدقة 60.54. لكن تم إجراء هذا التنبؤ لـ K = 1. لذلك، نحتاج إلى اختيار أفضل قيمة K.

	precision	recall	f1-score	support
0	0.61	0.64	0.63	10229
1	0.60	0.57	0.58	9618
accuracy			0.61	19847
macro avg	0.60	0.60	0.60	19847
weighted avg	0.61	0.61	0.60	19847

```
Accuracy score given for test data: 60.543155136796486
```

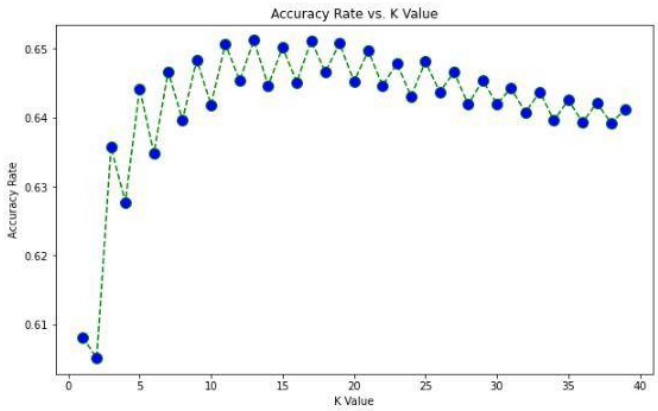
اختيار أفضل قيمة K.

تم استخدام طريقة الكوع elbow method لاختيار قيمة K جيدة من خلال التركيز على كل من الدقة ومعدل الخطأ.

1. اختر قيمة K بناءً على معدل الدقة.

هنا تم إنشاء قائمة لتخزين قيم معدل الدقة وسيتم تشغيل حلقة من 1 إلى 40 من أجل مراعاة قيمة K. داخل الحلقة، ستحسب قيمة الدقة لـ K ذات الصلة من 1 إلى 40 وتخزينهافي القائمة. من القائمة التي تم إنشاؤها، تم رسم رسم بياني لتحديد أفضل قيمة K والتي ستجعل الدقة أكثر استقرارًا.

```
accuracy_rate = []

for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    score=cross_val_score(knn,X,data['cardio'],cv=10)
    accuracy_rate.append(score.mean())

pt.figure(figsize=(10,6))
pt.plot(range(1,40),accuracy_rate,color='green', linestyle='dashed',
marker='o',
         markerfacecolor='blue', markersize=10)
pt.title('Accuracy Rate vs. K Value')
pt.xlabel('K')
pt.ylabel('Accuracy Rate')
```

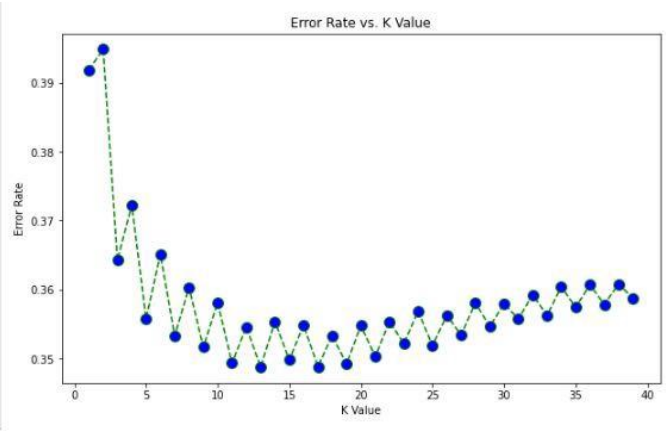


2. اختر قيمة K بناءً على معدل الخطأ.

مثل معدل الدقة، يتم حساب معدل الخطأ أيضًا على النحو التالي.

```
error_rate = []

for i in range(1,40):
```

```
    knn = KNeighborsClassifier(n_neighbors=i)
    score=cross_val_score(knn,X,data['cardio'],cv=10)
    error_rate.append(1-score.mean())

pt.figure(figsize=(10,6))
pt.plot(range(1,40),error_rate,color='green', linestyle='dashed',
marker='o',
        markerfacecolor='blue', markersize=10)
pt.title('Error Rate vs. K Value')
pt.xlabel('K Value')
pt.ylabel('Error Rate')
```



وفقًا للشكل 9، يمكن ملاحظة أنه بعد K> 5 سيزداد معدل الدقة ولن ينخفض أبدًا إلى ما دون النقطة حيث K> 5. وبالمثل، فإن معدل الخطأفي الشكل 10 بعد K> 5 ينخفض ولا يرتفع أبدًا فوق نقطة معينة حيث K> 5.

لذلك، يمكننا أن نفترض أن K = 5 هي قيمة جيدة جدًا بعد عملية اختيار قيمة K. بعد اختيار أفضل قيمة K، يمكننا مرة أخرى تشغيل الخوارزمية بقيمة K الجديدة التي تساوي 5.

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_training,y_training)
prediction = knn.predict(x_testing)
```

تقرير التصنيف لـ K = 5 كما يليفي الشكل 11 ووفقًا لذلك يمكننا أن نرى زيادة قيم الدقة والاستدعاء من K = 1 كما زادت قيمة الدقة أيضًا.

	precision	recall	f1-score	support
0	0.63	0.71	0.67	10229
1	0.64	0.56	0.60	9618
accuracy			0.64	19847
macro avg	0.64	0.63	0.63	19847
weighted avg	0.64	0.64	0.63	19847

```
Accuracy score given for test data: 63.71743840378898
```

الملخص

بمجرد اكتمال تنفيذ النموذج، كان ما كشفته قبل تنظيف البيانات غير ذات الصلة عن طريق إزالة القيم المتطرفة، كانت دقة النموذج 55٪. بعد تنقية البيانات الخاصة بالوزن والطول وap_hi وap_lo، تتحول دقة النموذج إلى 60٪. لكنها كانت لـ K = 1. بعد إعطاء أفضل قيمة K المحددة للنموذج كـ K = 5، زادت الدقة إلى 63٪. الفكرة الرئيسية وراء ذلك هي أن النموذج يعطي مستويات دقة مختلفة لقيم K مختلفة ويمكن تحديد أفضل قيمة K باستخدام تحليل معدل الخطأ أو معدل الدقة للنموذج. ومع ذلك، فإن حجم وميزات مجموعة البيانات لها تأثير كبير على النموذج للوصول إلى معدل دقة جيد.

ومع ذلك، لا تعمل خوارزمية KNN بشكل جيد مع مجموعات البيانات الكبيرة ولا تعمل بشكل جيد مع البيانات عالية الأبعاد لأنه من الصعب حساب المسافة بين كل نقطة بيانات وهو عيبفي الخوارزمية. لذلك، آملفي المستقبل تطبيق نفس مجموعة البيانات على خوارزميات التصنيف الأخرى والعثور على أفضل خوارزمية مناسبة لتحسين أداء النموذج ومشاركتها معكفي مقالاتي اللاحقة.

الكود متاح هنا.

16) توقع تناقص الموظفين باستخدام التعلم الآلي Employee Attrition Prediction using machine learning

في هذه المقالة، سأقدم لك مشروع التعلم الآلي حول التنبؤ بتناقص الموظفين باستخدام لغة برمجة Python. يعتبر الموظفون العمود الفقري للمؤسسة. يعتمد نجاح أو فشل المنظمة على الموظفين الذين يعملونفي المنظمة. يجب أن تتعامل المنظمات مع المشكلات عندما يغادر الموظفون المدربون والمهرة وذوي الخبرة المنظمة للحصول على فرص أفضل.

ما هو توقع تناقص الموظفين؟

يتم تقليص تناقص الموظفين Employee Attritionفي أي منظمة يستقيل فيها الموظفون. الموظفون أصول قيمة لأي منظمة. من الضروري معرفة ما إذا كان الموظفون غير راضين أو ما إذا كانت هناك أسباب أخرى لترك وظائفهم.

في الوقت الحاضر، للحصول على فرص أفضل، يتوق الموظفون إلى الانتقال من مؤسسة إلى أخرى. لكن إذا تركوا وظائفهم بشكل غير متوقع، فقد يؤدي ذلك إلى خسارة فادحة للمؤسسة. سوف يستهلك الموظف الجديد المال والوقت، وسيستغرق الموظفون المعينون حديثًا وقتًا لجعل المؤسسة المعنية مربحة.

يعد الاحتفاظ بالموظفين المهرة والعمل الدؤوب أحد أهم التحديات التي تواجهها العديد من المؤسسات. لذلك، من خلال تحسين رضا الموظفين وتوفير بيئة عمل مرغوبة، يمكننا بالتأكيد تقليل هذه المشكلة بشكل كبير.

مشروع التعلم الآلي حول توقع تناقص الموظفين باستخدام بايثون

في هذا القسم، سوف أخوضكفي مشروع تعلم الآلة حول توقع تناقص الموظفين باستخدام لغة برمجة Python. سأبدأ هذه المهمة عن طريق استيراد مكتبات Python الضرورية التي نحتاجها لهذه المهمة:

تحميل مجموعة البيانات

```
import numpy as np # linear algebra
import pandas as pd # data processing
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Import statements required for Plotly
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
```

```
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (accuracy_score, log_loss,
classification report)
from imblearn.over_sampling import SMOTE
import xgboost
```

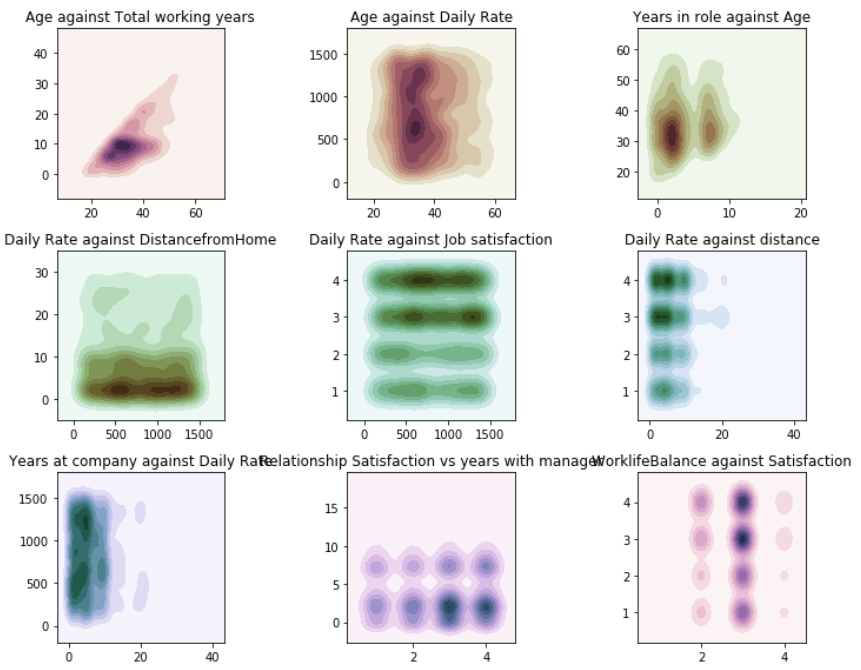
الآن دعنا نقرأ البيانات ونجري بعض التحليل الاستكشافي للبيانات لفهم مجموعة البيانات هذه بشكل صحيح:

```
attrition = pd.read_csv('Employee-Attrition.csv')
```

عادةً ما تكون إحدى الخطوات الأولىفي استكشاف البيانات هي الحصول على فكرة تقريبية عن كيفية توزيع الميزات فيما بينها. للقيام بذلك، سأستخدم دالة kdeplotفي مكتبة seabornفي Python:

```
f, axes = plt.subplots(3, 3, figsize=(10, 8),
                       sharex=False, sharey=False)

# Defining our colormap scheme
s = np.linspace(0, 3, 10)
cmap = sns.cubehelix_palette(start=0.0, light=1, as_cmap=True)

# Generate and plot
x = attrition['Age'].values
y = attrition['TotalWorkingYears'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, cut=5, ax=axes[0,0])
axes[0,0].set( title = 'Age against Total working years')

cmap = sns.cubehelix_palette(start=0.333333333333, light=1,
as_cmap=True)
# Generate and plot
x = attrition['Age'].values
y = attrition['DailyRate'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[0,1])
axes[0,1].set( title = 'Age against Daily Rate')

cmap = sns.cubehelix_palette(start=0.666666666667, light=1,
as_cmap=True)
# Generate and plot
x = attrition['YearsInCurrentRole'].values
y = attrition['Age'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[0,2])
axes[0,2].set( title = 'Years in role against Age')

cmap = sns.cubehelix_palette(start=1.0, light=1, as_cmap=True)
# Generate and plot
x = attrition['DailyRate'].values
y = attrition['DistanceFromHome'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[1,0])
axes[1,0].set( title = 'Daily Rate against DistancefromHome')

cmap = sns.cubehelix_palette(start=1.333333333333, light=1,
as_cmap=True)
# Generate and plot
x = attrition['DailyRate'].values
y = attrition['JobSatisfaction'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[1,1])
axes[1,1].set( title = 'Daily Rate against Job satisfaction')
```

```
cmap = sns.cubehelix_palette(start=1.666666666667, light=1,
as_cmap=True)
# Generate and plot
x = attrition['YearsAtCompany'].values
y = attrition['JobSatisfaction'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[1,2])
axes[1,2].set( title = 'Daily Rate against distance')

cmap = sns.cubehelix_palette(start=2.0, light=1, as_cmap=True)
# Generate and plot
x = attrition['YearsAtCompany'].values
y = attrition['DailyRate'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[2,0])
axes[2,0].set( title = 'Years at company against Daily Rate')

cmap = sns.cubehelix_palette(start=2.333333333333, light=1,
as_cmap=True)
# Generate and plot
x = attrition['RelationshipSatisfaction'].values
y = attrition['YearsWithCurrManager'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[2,1])
axes[2,1].set( title = 'Relationship Satisfaction vs years with
manager')

cmap = sns.cubehelix_palette(start=2.666666666667, light=1,
as_cmap=True)
# Generate and plot
x = attrition['WorkLifeBalance'].values
y = attrition['JobSatisfaction'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[2,2])
axes[2,2].set( title = 'WorklifeBalance against Satisfaction')

f.tight_layout()
```

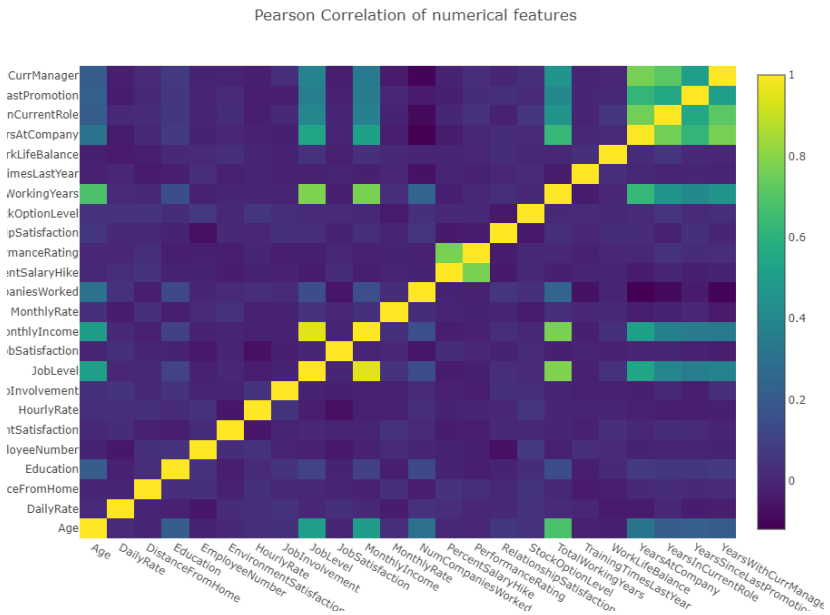


إيجاد الارتباط

الخطوة التاليةفي استكشاف البيانات هي العثور على مصفوفة الارتباط correlation matrix. من خلال رسم مصفوفة الارتباط، نحصل على نظرة جيدة حقًا على كيفية ارتباط الميزات ببعضها البعض.

في مخطط الارتباط هذا، سأستخدم مكتبة Plotlyفي Python لإنتاج مصفوفة ارتباط Pearson تفاعلية عبر دالة Heatmap على النحو التالي:

```
# Define a dictionary for the target mapping
target_map = {'Yes':1, 'No':0}
# Use the pandas apply method to numerically encode our attrition
target variable
attrition["Attrition_numerical"] = attrition["Attrition"].apply(lambda
x: target_map[x])

# creating a list of only numerical values
numerical = [u'Age', u'DailyRate', u'DistanceFromHome',
             u'Education', u'EmployeeNumber',
u'EnvironmentSatisfaction',
             u'HourlyRate', u'JobInvolvement', u'JobLevel',
u'JobSatisfaction',
             u'MonthlyIncome', u'MonthlyRate', u'NumCompaniesWorked',
             u'PercentSalaryHike', u'PerformanceRating',
u'RelationshipSatisfaction',
             u'StockOptionLevel', u'TotalWorkingYears',
             u'TrainingTimesLastYear', u'WorkLifeBalance',
u'YearsAtCompany',
             u'YearsInCurrentRole',
u'YearsSinceLastPromotion',u'YearsWithCurrManager']
data = [
    go.Heatmap(
        z= attrition[numerical].astype(float).corr().values, #
Generating the Pearson correlation
        x=attrition[numerical].columns.values,
        y=attrition[numerical].columns.values,
        colorscale='Viridis',
        reversescale = False,
#         text = True ,
        opacity = 1.0

    )
]


layout = go.Layout(
    title='Pearson Correlation of numerical features',
    xaxis = dict(ticks='', nticks=36),
    yaxis = dict(ticks='' ),
    width = 900, height = 700,

)


fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='labelled-heatmap')
```



ملاحظات من المخطط السابق:

من مخطط الارتباط، يمكننا أن نرى أن الكثير من أعمدتنا تبدو ضعيفة الارتباط ببعضها البعض. بشكل عام، عند بناء نموذج تنبؤي، سيكون من الأفضل تدريب نموذج بميزات لا ترتبط ببعضها البعض كثيرًا حتى لا نحتاج إلى التعامل مع الميزات الزائدة عن الحاجة.

في حالة وجود عدد كبير من الخصائص المترابطة، ربما يمكننا تطبيق تقنية مثل تحليل المكون الرئيسي (PCA) لتقليل المساحة المميزة.

هندسة الميزات

بعد استكشاف مجموعة البيانات الخاصة بنا، دعنا ننتقل الآن إلى مهمة هندسة الميزات features engineering والترميز الرقمي للقيم الفئويةفي مجموعة البيانات الخاصة بنا. تتضمن هندسة الميزات إنشاء ميزات وعلاقات جديدة من الميزات الحالية التي لدينا.

بالنسبة لهذه المهمة، سنفصل الأعمدة الرقمية عن الأعمدة الفئوية على النحو التالي:

```
attrition = attrition.drop(['Attrition_numerical'], axis=1)

# Empty list to store columns with categorical data
categorical = []
for col, value in attrition.iteritems():
    if value.dtype == 'object':
        categorical.append(col)

# Store the numerical columns in a list numerical
numerical = attrition.columns.difference(categorical)
```

بعد تحديد أي من ميزاتنا تحتوي على بيانات فئوية، يمكننا البدءفي ترميز البيانات رقميًا. للقيام بذلك، سأستخدم طريقة get_dummies الخاصة بـ Pandasفي Python والتي تنشئ متغيرات وهمية مرمزة من المتغيرات الفئوية:

```
attrition_cat = attrition[categorical]
attrition_cat = attrition_cat.drop(['Attrition'], axis=1) # Dropping
the target column
attrition_cat = pd.get_dummies(attrition_cat)
attrition_cat.head(3)
attrition_num = attrition[numerical]
attrition_final = pd.concat([attrition_num, attrition_cat], axis=1)
```

الخطوة الأخيرة التي نحتاج إلى تذكرها هي إنشاء المتغير المستهدف. الهدف،في هذه الحالة، يتم تحديده بواسطة عمود Attrition الذي يحتوي على متغيرات فئوية وبالتالي يتطلب ترميزًا رقميًا. نقوم بترميزها رقميًا عن طريق إنشاء قاموس باستخدام التعيين المحدد كـ 1: نعم و0: لا:

```
target_map = {'Yes':1, 'No':0}
 #Use the pandas apply method to numerically encode our attrition
target variable
target = attrition["Attrition"].apply(lambda x: target_map[x])
```

التعلم الآلي لتوقع تناقص الموظفين باستخدام بايثون

الآن، نحتاج إلى تدريب نموذج التعلم الآلي للتنبؤ بتوقع تناقص الموظفين باستخدام Python. بالنسبة لهذه المهمة، سأستخدم نموذج تصنيف الغابات العشوائية Random Forest المقدم من Scikit-Learn.

ولكن قبل تنفيذ التعلم الآلي للتنبؤ بتوقع تناقص الموظف، نحتاج إلى تقسيم البيانات إلى مجموعة تدريب ومجموعة اختبار:

```
from sklearn.cross_validation import train_test_split
from sklearn.cross_validation import StratifiedShuffleSplit

# Split data into train and test sets as well as for validation and
testing
train, test, target_train, target_val =
train_test_split(attrition_final,
                                                        target,
                                                        train_size=
0.80,

random_state=0);
#train, test, target train, target val =
StratifiedShuffleSplit(attrition_final, target, random_state=0)
```

دعنا الآن ندرب نموذج تصنيف الغابة العشوائية لمهمة التنبؤ بتناقص الموظفين باستخدام التعلم الآلي وPython:

```
oversampler=SMOTE(random_state=0)
smote_train, smote_target = oversampler.fit_sample(train,target_train)

seed = 0   # We set our random seed to zero for reproducibility
# Random Forest parameters
rf_params = {
```

```
    'n_jobs': -1,
    'n_estimators': 1000,
#      'warm_start': True,
    'max_features': 0.3,
    'max_depth': 4,
    'min_samples_leaf': 2,
    'max_features' : 'sqrt',
    'random_state' : seed,
    'verbose': 0
}

rf = RandomForestClassifier(**rf_params)
rf.fit(smote train, smote target)
rf_predictions = rf.predict(test)
print("Accuracy score: {}".format(accuracy_score(target_val,
rf_predictions)))
print("="*80)
print(classification_report(target_val, rf_predictions))
```

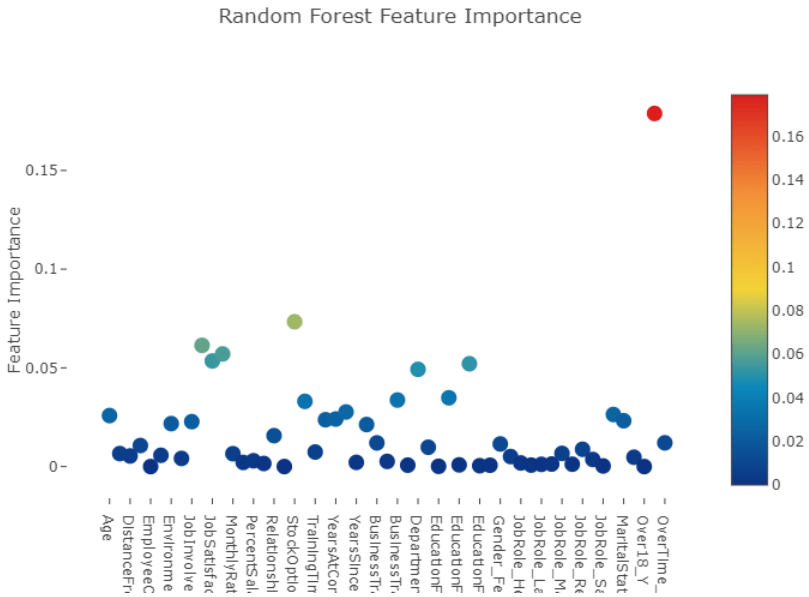
```
Accuracy score: 0.8537414965986394
================================================================================
             precision    recall  f1-score   support

          0       0.90      0.93      0.91       245
          1       0.57      0.49      0.53        49

avg / total       0.85      0.85      0.85       294
```

كما لوحظ، فإن Random Forest الخاصة بنا ترجع دقة حوالي 88 ٪ لتنبؤاتها، وللوهلة الأولى، قد يبدو هذا نموذجًا جيدًا إلى حد ما.

يحتوي مصنف Sklearn's Random Forest أيضًا على سمة مفيدة جدًا لتحليل أهمية الميزة والتي تخبرنا عن الميزات الموجودةفي مجموعة البيانات الخاصة بنا والتي تلقت الأهمية الأكبر بواسطة خوارزمية Random Forest. دعنا نتخيل الميزات التي تم أخذهافي الاعتبار من خلال نموذج التعلم الآلي الخاص بنا لتناقص الموظفين:

```
trace = go.Scatter(
    y = rf.feature_importances_,
    x = attrition_final.columns.values,
    mode='markers',
    marker=dict(
        sizemode = 'diameter',
        sizeref = 1,
        size = 13,
        #size= rf.feature_importances_,
        #color = np.random.randn(500), #set color equal to a variable
        color = rf.feature_importances_,
        colorscale='Portland',
        showscale=True
    ),
    text = attrition_final.columns.values
)
data = [trace]
```

```
layout= go.Layout(
    autosize= True,
    title= 'Random Forest Feature Importance',
    hovermode= 'closest',
     xaxis= dict(
         ticklen= 5,
         showgrid=False,
        zeroline=False,
        showline=False
     ),
    yaxis=dict(
        title= 'Feature Importance',
        showgrid=False,
        zeroline=False,
        ticklen= 5,
        gridwidth= 2
    ),
    showlegend= False
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig,filename='scatter2010')
```



آمل أن تكون قد أحببت هذه المقالة حول مشروع التعلم الآلي حول توقع تناقص الموظفين باستخدام لغة برمجة Python.

17) كشف الاحتيال باستخدام التعلم الآلي Fraud Detection using Machine Learning

يعد الاحتيال Fraud أحد المشكلات الرئيسية التي نطرحها بشكل رئيسيفي البنوك والتأمين على الحياة والتأمين الصحي وغيرها الكثير. تعتمد عمليات الاحتيال الرئيسية هذه على الشخص الذي يحاول بيع المنتج أو الخدمة المزيفة لك، إذا كنت قد نضجت بما يكفي لتقرير الخطأ، فلن تدخل أبدًافي أي معاملات احتيال. لكن أحد هذه الاحتيالات التي تتزايد كثيرًا هذه الأيام هو الاحتيالفي سداد المدفوعات.في هذه المقالة، سأنتقل بك إلى حل للكشف عن الاحتيال باستخدام التعلم الآلي.

يمكن تنزيل مجموعة البيانات التي سأستخدمها لهذه المهمة بسهولة من هنا. مجموعة البيانات التي أستخدمها هي بيانات المعاملات للمشتريات عبر الإنترنت التي تم جمعها من بائع تجزئة للتجارة الإلكترونية. تحتوي مجموعة البيانات على أكثر من 39000 معاملة، تحتوي كل معاملة على 5 ميزات تصف طبيعة المعاملات. لذا فلنبدأ باستيراد جميع المكتبات الضرورية التي نحتاجها لاكتشاف الاحتيال باستخدام التعلم الآلي:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
```

نموذج كشف الاحتيال في الدفع

لحسن الحظ، فإن مجموعة البيانات التي أستخدمها منظمة بشكل جيد للغاية مع عدم وجود قيم مفقودة فيها، ولا أجد أي نطاق لتنظيف البيانات فيها. لذلك دون إضاعة أي وقت، سأغوصفي بناء نموذج التعلم الآلي الخاص بنا. لذا، سأبدأ الآن باستيراد البيانات:

```
from google.colab import files
uploaded = files.upload()
df = pd.read_csv('payment_fraud.csv')
df.head()
```

	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays	label
0	29	1	4.745402	paypal	28.204861	0
1	725	1	4.742303	storecredit	0.000000	0
2	845	1	4.921318	creditcard	0.000000	0
3	503	1	4.886641	creditcard	0.000000	0
4	2000	1	5.040929	creditcard	0.000000	0

الآن، سأقسم البيانات إلى مجموعات تدريب واختبار:

```
# Split dataset up into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
```

```
    df.drop('label', axis=1), df['label'],
    test_size=0.33, random_state=17)
```

نظرًا لأن هذه مشكلة تصنيف ثنائي، فسوف أستخدم خوارزمية الانحدار اللوجستي Logistic Regression، لأنها واحدة من أقوى الخوارزميات لنموذج التصنيف الثنائي. إذا كنت لا تعرف معنى التصنيف الثنائي، فيمكنك تعلمه من هنا. الآن، دعنا نقوم ببساطة بتدريب نموذج الكشف عن الاحتيال باستخدام خوارزمية الانحدار اللوجستي وإلقاء نظرة على درجة الدقة التي سنحصل عليها باستخدام هذه الخوارزمية:

```
clf = LogisticRegression().fit(X_train, y_train)

# Make predictions on test set
y_pred = clf.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(y_pred, y_test))
```

```
1.0
```

حسنًا، ما هي آخر مرة حصلت فيها على دقة بنسبة 100في المائة. أعطى نموذج الكشف عن الاحتيال الخاص بنا دقة بنسبة 100في المائة باستخدام خوارزمية الانحدار اللوجستي.

تقييم نموذج كشف الاحتيال

الآن، دعنا نقيم أداء نموذجنا. سأستخدم خوارزمية مصفوفة الارتباك لتقييم أداء نموذجنا. يمكننا استخدام خوارزمية مصفوفة الارتباك مع كود من سطر واحد فقط:

```
# Compare test set predictions with ground truth labels
print(confusion_matrix(y_test, y_pred))
```

```
[[12753 0]
[ 0 190]]
```

لذلك من بين جميع المعاملاتفي مجموعة البيانات، تم التعرف على 190 معاملة بشكل صحيح على أنها احتيال، وتم التعرف على 12753 معاملة على أنها ليست معاملات احتيالية. آمل أن تكون قد أحببت هذه المقالة حول اكتشاف الاحتيال باستخدام التعلم الآلي.

18) تصنيف سرطان الثدي باستخدام التعلم الآلي Breast Cancer Classification using Machine Learning

في هذا البرنامج التعليمي، سننشئ نموذجًا للتنبؤ بما إذا كان لدى المريض تشخيص إيجابي لسرطان الثدي بناءً على العديد من ميزات الورم.

عرض المشكلة

قاعدة بيانات سرطان الثدي هي مجموعة بيانات متاحة للجمهور من مستودع التعلم الآلي الخاص بـ UCI. يعطي معلومات عن سمات الورم مثل حجم الورم وكثافته وملمسه.

الهدف: إنشاء نموذج تصنيف يبحثفي التنبؤات إذا كان تشخيص السرطان حميدًا أو خبيثًا بناءً على عدة ميزات.

البيانات المستخدمة: مجموعة بيانات Kaggle-Breast Cancer Prediction

الخطوة 1: استكشاف مجموعة البيانات

أولاً، دعنا نفهم مجموعة البيانات الخاصة بنا:

```
#import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
#import models from scikit learn module:
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.svm import SVC
#import Data
df_cancer = pd.read_csv('Breast_cancer_data.csv')
df_cancer.head()
#get some information about our Data-Set
df_cancer.info()
df_cancer.describe()
#visualizing data
sns.pairplot(df_cancer, hue = 'diagnosis')plt.figure(figsize=(7,7))
sns.heatmap(df_cancer['mean_radius mean_texture mean_perimeter
mean_area mean_smoothness diagnosis'.split()].corr(),
annot=True)sns.scatterplot(x = 'mean_texture', y = 'mean_perimeter',
hue = 'diagnosis', data = df_cancer)
```

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	Benign
1	20.57	17.77	132.90	1326.0	0.08474	Benign
2	19.69	21.25	130.00	1203.0	0.10960	Benign
3	11.42	20.38	77.58	386.1	0.14250	Benign
4	20.29	14.34	135.10	1297.0	0.10030	Benign

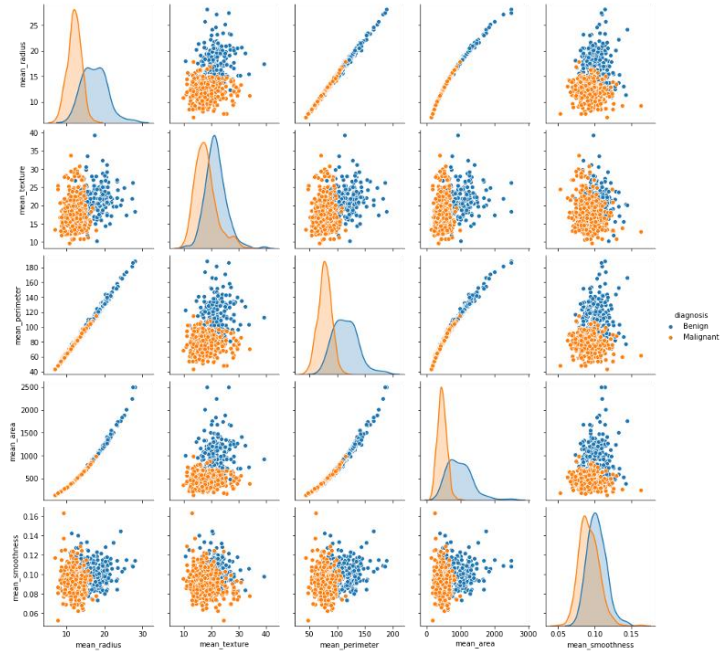
Breast_cancer_data

```
<bound method DataFrame.info of      mean_radius  mean_texture  ...  mean_smoothness  diagnosis
0          17.99         10.38  ...          0.11840     Benign
1          20.57         17.77  ...          0.08474     Benign
2          19.69         21.25  ...          0.10960     Benign
3          11.42         20.38  ...          0.14250     Benign
4          20.29         14.34  ...          0.10030     Benign
..           ...           ...  ...              ...        ...
564        21.56         22.39  ...          0.11100     Benign
565        20.13         28.25  ...          0.09780     Benign
566        16.60         28.08  ...          0.08455     Benign
567        20.60         29.33  ...          0.11780     Benign
568         7.76         24.54  ...          0.05263  Malignant

[569 rows x 6 columns]>
```

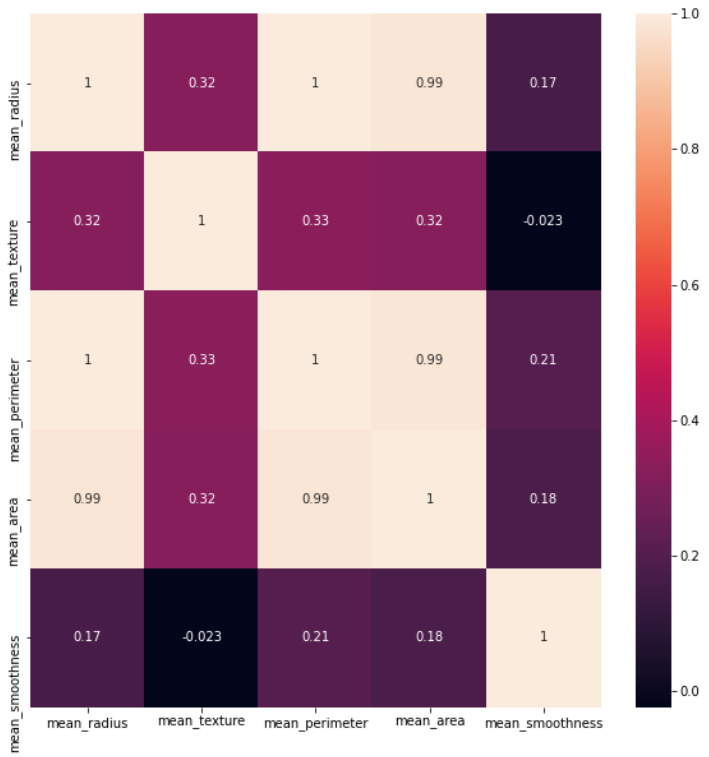
بعض المعلومات حول مجموعة البيانات

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness
count	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360
std	3.524049	4.301036	24.298981	351.914129	0.014064
min	6.981000	9.710000	43.790000	143.500000	0.052630
25%	11.700000	16.170000	75.170000	420.300000	0.086370
50%	13.370000	18.840000	86.240000	551.100000	0.095870
75%	15.780000	21.800000	104.100000	782.700000	0.105300
max	28.110000	39.280000	188.500000	2501.000000	0.163400

Data.describe( )



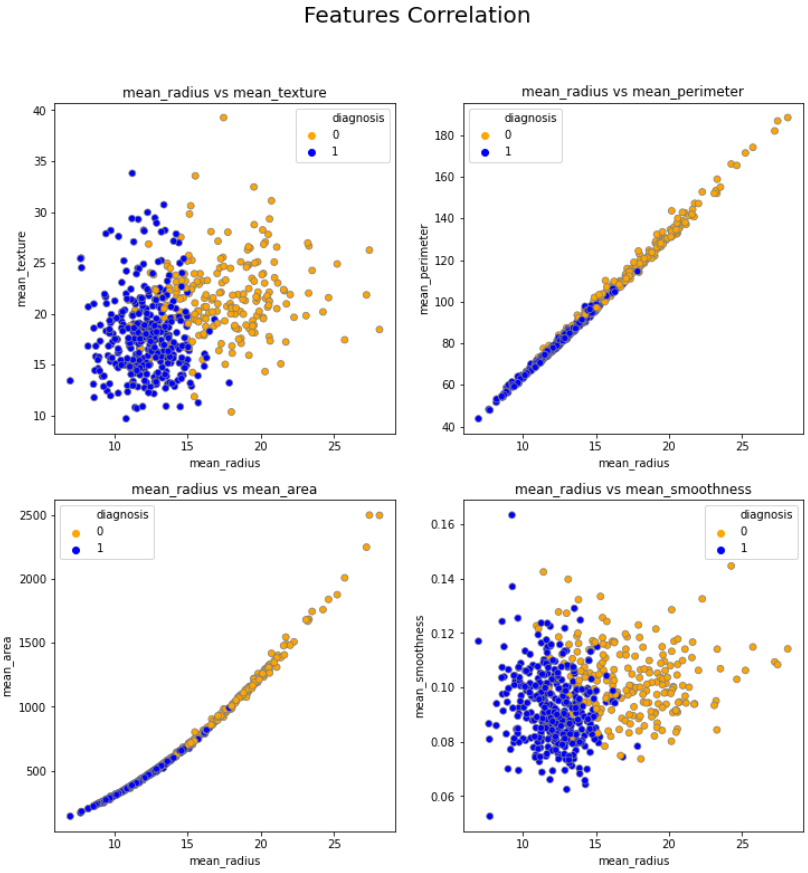
زوج من الميزات من Breast_cancer_data



الارتباط بين السمات

```
#visualizing features correlation
palette ={0 : 'orange', 1 : 'blue'}
edgecolor = 'grey'

fig = plt.figure(figsize=(12,12))
plt.subplot(221)

ax1 = sns.scatterplot(x = df_cancer['mean_radius'], y =
df_cancer['mean_texture'], hue = "diagnosis",
data = df_cancer, palette =palette, edgecolor=edgecolor)
plt.title('mean_radius vs mean_texture')

plt.subplot(222)
ax2 = sns.scatterplot(x = df_cancer['mean_radius'], y =
df_cancer['mean_perimeter'], hue = "diagnosis",
data = df_cancer, palette =palette, edgecolor=edgecolor)
plt.title('mean_radius vs mean_perimeter')

plt.subplot(223)
ax3 = sns.scatterplot(x = df_cancer['mean_radius'], y =
df_cancer['mean_area'], hue = "diagnosis",
data = df_cancer, palette =palette, edgecolor=edgecolor)
plt.title('mean_radius vs mean_area')

plt.subplot(224)
ax4 = sns.scatterplot(x = df_cancer['mean_radius'], y =
df_cancer['mean_smoothness'], hue = "diagnosis",
data = df_cancer, palette =palette, edgecolor=edgecolor)
plt.title('mean_radius vs mean_smoothness')
```

```
fig.suptitle('Features Correlation', fontsize = 20)
plt.savefig('2')
plt.show()
```



الخطوة 2: معالجة البيانات المفقودة/الفئوية

- قبل تطبيق أي طريقة، نحتاج إلى التحقق مما إذا كانت هناك أي قيم مفقودة ثم التعامل معها إذا كان الأمر كذلك.في مجموعة البيانات هذه، لا توجد قيم مفقودة – ولكن احتفظ دائمًا بعادة التحقق من القيم الخاليةفي مجموعة البيانات!
- نظرًا لأن نماذج التعلم الآلي تستند إلى معادلات رياضية، فنحن بحاجة إلى ترميز المتغيرات الفئوية. لقد استخدمت هنا ترميز التسمية label encoding نظرًا لوجود قيمتين مميزتينفي عمود التشخيص "diagnosis":

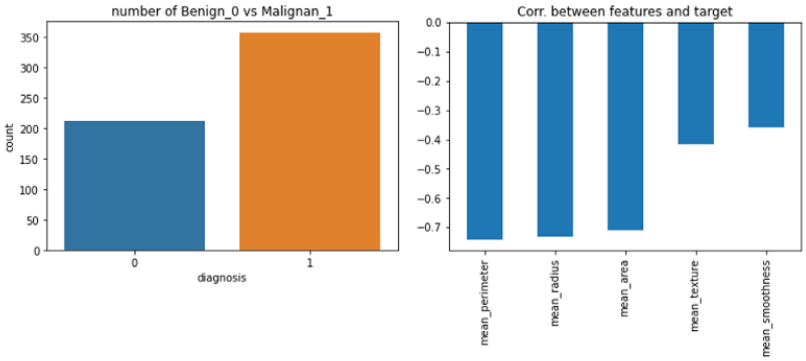
```
#check how many values are missing (NaN)
here we do not have any missing values
df_cancer.isnull().sum()
```

```
#handling categorical data
df_cancer['diagnosis'].unique()

df_cancer['diagnosis'] =
df_cancer['diagnosis'].map({'benign':0,'malignant':1})

df_cancer.head()
```

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

دعونا نستمرفي معرفة مجموعة البيانات الخاصة بنا:

```
#visualizing diagnosis column >>> 'benign':0,'malignant':1
sns.countplot(x='diagnosis',data = df_cancer)

plt.title('number of Benign_0 vs Malignan_1')#

correlation between features
df_cancer.corr()['diagnosis'][:-1].sort_values().plot(kind ='bar')
plt.title('Corr. between features and target')
```



الخطوة 3: تقسيم مجموعة البيانات إلى مجموعة تدريب ومجموعة اختبار

تنقسم البيانات إلى مجموعة التدريب Trainومجموعة الاختبار Test. نستخدم مجموعة التدريب Trainلجعل الخوارزمية تتعرف على سلوك البيانات ثم تتحقق من دقة نموذجنافي مجموعة الاختبار Test.

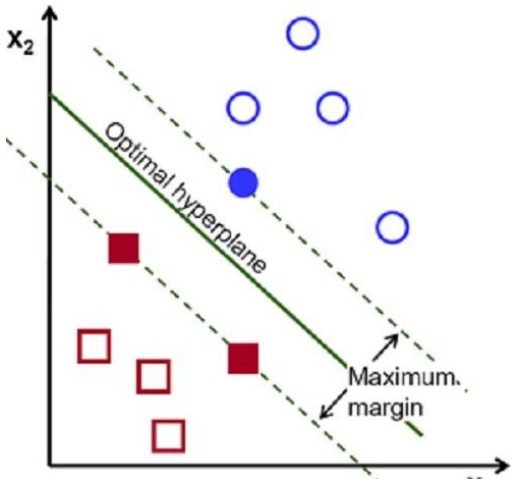
- الميزات Features (X): سيتم استخدام الأعمدة التي تم إدراجهافي نموذجنا لعمل تنبؤات.

- التنبؤ Prediction (y): المتغير المستهدف الذي ستتنبأ به الميزات.

```
#define X variables and our target(y)
X = df_cancer.drop(['diagnosis'],axis=1).values
y = df_cancer['diagnosis'].values

#split Train and Test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=101)
```

الخطوة 4: آلة دعم المتجهات لنمذجة البيانات

تعد Support Vector Machine (SVM) واحدة من أكثر خوارزميات التعلم الآلي الخاضعة للإشراف المفيدة. يمكن استخدامه لكل من مهام التصنيف والانحدار.



هناك نوعان من المفاهيم التي نحتاج أولاً إلى فهمها:

- **ما هي وظيفة SVM؟** يختار SVM المستوى الفائق الذي يقوم بأقصى قدر من الفصل بين الفئات.
- **ما هي الهوامش الصلبة والناعمة hard and soft margins؟** إذا كان من الممكن فصل البيانات خطيًا، فقد يُرجع SVM أقصى درجات الدقة (الهامش الصلب). عندما لا تكون البيانات قابلة للفصل خطيًا، فكل ما نحتاج إليه هو تخفيف الهامش للسماح بالتصنيفات الخاطئة (الهامش الناعم).
- **ما هو المعلم الفائق Hyper-parameter C؟** يمكن التحكم في عدد أخطاء التصنيف الخاطئ باستخدام المعلمة C، والتي لها تأثير مباشر على المستوى الفائق.
- **ما هي غاما ذات المعلمات الفائقة؟** تُستخدم غاما لإعطاء وزن للنقاط القريبة من متجه الدعم. بمعنى آخر، تغيير قيمة غاما سيغير شكل المستوي الفائق.

- **ما هي خدعة النواة Kernel Trick؟** إذا لم تكن بياناتنا قابلة للفصل خطيًا، فيمكننا تطبيق طريقة "Kernel Trick" التي ترسم البيانات غير الخطية إلى مساحة ذات أبعاد أعلى.

الآن دعنا نعود إلى الكود الخاص بنا!

```
#Support Vector Classification model
from sklearn.svm import SVC
svc_model = SVC()
svc_model.fit(X_train, y_train)
```

الخطوة 5: تقييم النموذج

```
from sklearn.metrics import classification_report,
confusion_matrixy_predict = svc_model.predict(X_test)
cm = confusion_matrix(y_test, y_predict)
sns.heatmap(cm, annot=True)
```

ماذا تعني نتيجة معلومات مصفوفة الارتباك confusion_matrix ؟:

- كان لدينا 143 امرأةفي مجموعة الاختبار الخاصة بنا.
- من بين 55 امرأة تم توقع عدم إصابتهن بسرطان الثدي، تم تصنيف امرأتين على أنهن لم تكن مصابات بسرطان الثديفي حين أنهن مصابات بالفعل (خطأ من النوع الأول).
- من بين 88 امرأة يُتوقع إصابتهن بسرطان الثدي، تم تصنيف 14 امرأة على أنها مصابة بسرطان الثدي لم يتم تصنيفهن (خطأ من النوع الثاني).

	precision	recall	f1-score	support
0	0.95	0.75	0.84	55
1	0.86	0.98	0.91	88
accuracy			0.89	143
macro avg	0.91	0.86	0.88	143
weighted avg	0.90	0.89	0.88	143

ماذا تعني نتيجة تقرير التصنيف هذا؟ يعني هذافي الأساس أن نموذج SVM كان قادرًا على تصنيف الأورام إلى أورام خبيثة وحميدة بدقة 89 ٪.

ملحوظة:

- Precision هي جزء من النتائج ذات الصلة.
- Recall هو جزء من جميع النتائج ذات الصلة التي تم تصنيفها بشكل صحيح.
- F1-score هي المتوسط التوافقي بين Precision وRecall الذي يتراوح بين 0 (سيئ) إلى 1 (الكمال).

الخطوة 6: ما الذي يمكننا فعله لتحسين نموذجنا؟

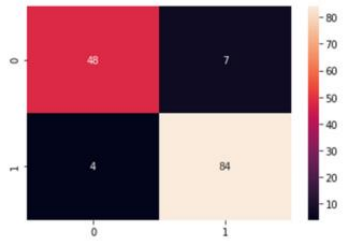
1. تسوية البيانات

سيساعدنا تحجيم الميزة على رؤية جميع المتغيرات من نفس المقياس، وبهذه الطريقة سنضع جميع القيمفي النطاق [1،0]:

```
#normalized scaler - fit&transform on train, fit only on test
from sklearn.preprocessing import MinMaxScaler
n_scaler = MinMaxScaler()

X_train_scaled = n_scaler.fit_transform(X_train.astype(np.float))
X_test_scaled = n_scaler.transform(X_test.astype(np.float))

#Support Vector Classification model -  apply on scaled data
from sklearn.svm import SVC
svc_model = SVC()
svc_model.fit(X_train_scaled, y_train)

from sklearn.metrics import classification_report, confusion_matrix
y_predict_scaled = svc_model.predict(X_test_scaled)
cm = confusion_matrix(y_test, y_predict_scaled)
sns.heatmap(cm, annot=True)

print(classification_report(y_test, y_predict_scaled))
```

ماذا تعني نتيجة معلومات مصفوفة الارتباك ؟:

- كان لدينا 143 امرأةفي مجموعة الاختبار الخاصة بنا
- من بين 55 امرأة تم توقع عدم إصابتهن بسرطان الثدي، تم تصنيف 4 نساء على أنهن لم يكن مصابات بسرطان الثديفي حين أنهن مصابات بالفعل (الخطأ من النوع 1)
- من بين 88 امرأة يتوقع إصابتهن بسرطان الثدي، تم تصنيف 7 نساء على أنهن مصابات بسرطان الثدي لم يتم تصنيفهن (خطأ من النوع 2).



	precision	recall	f1-score	support
0	0.92	0.87	0.90	55
1	0.92	0.95	0.94	88
accuracy			0.92	143
macro avg	0.92	0.91	0.92	143
weighted avg	0.92	0.92	0.92	143

ماذا تعني نتيجة تقرير التصنيف هذا؟في الأساس، هذا يعني أن نموذج SVM كان قادرًا على تصنيف الأورام إلى أورام خبيثة / حميدة بدقة 92 ٪.

2. تحسين معلمات SVM

المعلمة C: كما قلنا، تتحكمفي تكلفة التصنيف الخاطئ لبيانات التدريب.

- أصغر C: تباين أقل ولكن تحيز أعلى (هامش ناعم) وتقليل تكلفة التصنيف الخاطئ (عقوبة أقل).
- أكبر C: انحياز أقل وتباين أعلى (هامش صعب) وزيادة تكلفة التصنيف الخاطئ (أكثر صرامة).

غاما:

- غاما أصغر: تباين كبير، بعيد المدى، وحل أكثر عمومية.
- غاما أكبر: يتمتع التباين العالي والتحيز المنخفض والوصول القريب وكذلك نقاط البيانات الأقرب بوزن أعلى.

لذلك، دعنا نعثر على المعلمات المثلى لنموذجنا باستخدام بحث الشبكة grid search:

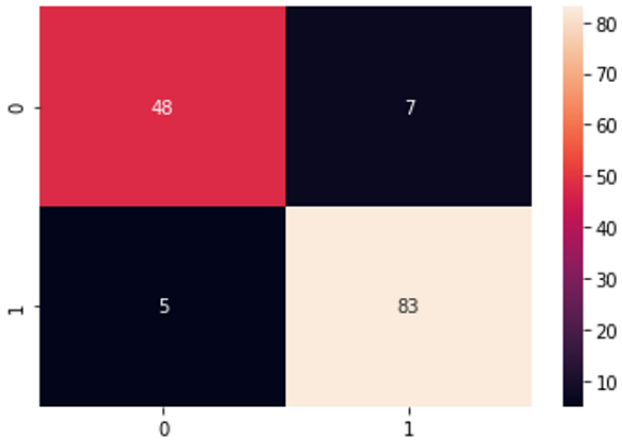
```
#find best hyper parameters
from sklearn.model_selection import GridSearchCV

param_grid = {'C':[0.1,1,10,100,1000],'gamma':[1,0.1,0.01,0.001,0.001],
'kernel':['rbf']}

grid = GridSearchCV(SVC(),param_grid,verbose = 4)
grid.fit(X_train_scaled,y_train)

grid.best_params_
grid.best_estimator_

grid_predictions = grid.predict(X_test_scaled)
cmG = confusion_matrix(y_test,grid_predictions)
sns.heatmap(cmG, annot=True)

print(classification_report(y_test,grid_predictions))
```

```
grid.best_params_
```

```
{'C': 1000, 'gamma': 0.1, 'kernel': 'rbf'}
```

```
grid.best_estimator_
```

```
SVC(C=1000, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```



كما تريفي هذه الحالة، لم ينتج عن تحسين النموذج الأخير نسبة الدقة. ومع ذلك، نجحنافي تقليل الخطأ من النوع الثاني.

19) توقع أسعار البيتكوين باستخدام التعلم الآلي Bitcoin Price Prediction using Machine Learning

في مشروع علم البيانات هذا، سوف نتوقع سعر البيتكوين للأيام الثلاثين القادمة باستخدام آلات المتجهات الداعمة SVM لنموذج التعلم الآلي (الانحدار).

يمكنك تنزيل مجموعة البيانات التي نحتاجها لهذه المهمة من هنا.

لنبدأ باستيراد المكتبات:

```
import numpy as np
import pandas as pd
df = pd.read_csv("bitcoin.csv")
df.head()
```

	Date	Price
0	5/23/2019	7881.846680
1	5/24/2019	7987.371582
2	5/25/2019	8052.543945
3	5/26/2019	8673.215820
4	5/27/2019	8805.778320

المعالجة المسبقة للبيانات

قم بإزالة عمود التاريخ date column:

```
df.drop(['Date'],1,inplace=True)
```

لنقم الآن بإنشاء متغير للتنبؤ بـ "n" الأيام الموجودة هناكفي المستقبل.

```
predictionDays = 30
 #Create another column shifted 'n'  units up
df['Prediction'] = df[['Price']].shift(-predictionDays)
 #show the first 5 rows
df.head()
```

	Price	Prediction
0	7881.846680	10701.69141
1	7987.371582	10855.37109
2	8052.543945	11011.10254
3	8673.215820	11790.91699
4	8805.778320	13016.23145

لإظهار آخر 5 صفوف من مجموعة البيانات الجديدة.

```
df.tail()
```

	Price	Prediction
362	9729.038086	NaN
363	9522.981445	NaN
364	9081.761719	NaN
365	9182.577148	NaN
366	9180.045898	NaN

قم بإنشاء مجموعة البيانات المستقلة independent data set.

```
#Create the independent dada set
#Here we will convert the data frame into a numpy array and drp the
prediction column
x = np.array(df.drop(['Prediction'],1))
#Remove the last 'n' rows where 'n' is the predictionDays
x = x[:len(df)-predictionDays]


print(x)
```

```
#Output
[[ 7881.84668 ]
 [ 7987.371582]
 [ 8052.543945]
 [ 8673.21582 ]
 [ 8805.77832 ]
 [ 8719.961914]
 [ 8659.487305]
 [ 8319.472656]
 [ 8574.501953]
```

قم بإنشاء مجموعة البيانات التابعة dependent data set.

```
 #Create the dependent data set
 #convert the data frame into a numpy array
y = np.array(df['Prediction'])
 #Get all the values except last 'n' rows
y = y[:-predictionDays]


print(y)
```

```
#Output
[10701.69141  10855.37109  11011.10254  11790.91699  13016.23145
 11182.80664  12407.33203  11959.37109  10817.15527  10583.13477
 10801.67773  11961.26953  11215.4375   10978.45996  11208.55078
 11450.84668  12285.95801  12573.8125   12156.5127   11358.66211
 11815.98633  11392.37891  10256.05859  10895.08984   9477.641602
  9693.802734 10666.48242  10530.73242  10767.13965  10599.10547
 10343.10645   9900.767578  9811.925781  9911.841797  9870.303711
  9477.677734  9552.860352  9519.145508  9607.423828 10085.62793
 10399.66895  10518.17481  10821.72656  10970.18457  11805.65332
```

تقسيم البيانات

قسّم البيانات إلى 80٪ تدريب و20٪ اختبار.

```
 #Split the data into 80% training and 20% testing
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size = 0.2)
 #set the predictionDays array equal to last 30 rows from the original
data set
predictionDays_array = np.array(df.drop(['Prediction'],1))[-
predictionDays:]
print(predictionDays_array)
```

```
#Output
[[7550.900879]
 [7569.936035]
 [7679.867188]
 [7795.601074]
 [7807.058594]
 [8801.038086]
 [8658.553711]
 [8864.766602]
 [8988.59668 ]
 [8897.46875 ]
 [8912.654297]
 [9003.070313]
 [9268.761719]
 [9951.518555]
 [9842.666016]
```

انشاء النموذج

سنقوم الآن بإنشاء نموذج التعلم الآلي:

```
from sklearn.svm import SVR
 #Create and Train the Support Vector Machine (Regression) using radial
basis function
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.00001)
svr_rbf.fit(xtrain, ytrain)
```

```
#Output
SVR(C=1000.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma=1e-
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

اختبار النموذج

سنقوم الآن بإنشاء نموذج التعلم الآلي:

```
svr_rbf_confidence = svr_rbf.score(xtest,ytest)
print('SVR_RBF accuracy :',svr_rbf_confidence)
```

```
#Output
SVR_RBF accuracy : 0.80318203039572782
```

طباعة القيم المتوقعة:

```
#print the predicted values
svm_prediction = svr_rbf.predict(xtest)
print(svm_prediction)
print()
print(ytest)
```

```
#Output
[ 8580.88704057  8598.79383546  8939.94375214  8388.92621489
  9102.56201779  8832.68229779  8329.30224101  8157.80057348
  8602.29644729  8707.90682044  7643.06939601  8408.93105022
  8917.45480981  8511.7652266   7834.15919638  8832.62858497
  8381.02268219  7098.85213417  8805.2578118   7757.01224446
  8791.58493431  8961.26396398  8218.28537299 10512.39752674
  8505.95838523  8504.09557077  8416.46565526  8812.06086838
  8565.94893198  8378.22399262  8585.8737782   7630.00945667
  9602.30696397  8934.97064742  9812.06855777  8473.66659984
  8408.82946381 10548.41305096  9362.68382564  8597.33711016
  7730.30747013  7792.1701846   8840.84467855  9893.05484852
```

طباعة تنبؤات النموذج للأيام الثلاثين القادمة:

```
#Print the model predictions for the next 30 days
svm_prediction = svr_rbf.predict(predictionDays_array)
print(svm_prediction)
print()
#Print the actual price for bitcoin for last 30 days
print(df.tail(predictionDays))
```

```
#Output
[7746.08637672 7835.76387711 8657.25433329 9554.67400231 9617.98954538
 8917.61532094 8831.29326224 8885.55655284 8640.51491415 8841.78875835
 8815.42825999 8602.53094625 8400.08270252 8426.55153101 8172.93870238
 8395.85348921 8903.73403919 8811.70139747 8917.58878224 8402.31118948
 8102.70537693 8518.83392876 8606.8071745  8195.93279966 8108.54622414
 8106.38537126 8573.49097641 8410.28935674 8307.6380027  8307.33725309]

            Price  Prediction
337   7550.900879         NaN
338   7569.936035         NaN
339   7679.867188         NaN
340   7795.601074         NaN
341   7807.058594         NaN
342   8801.038086         NaN
```

20) الكشف عن العملات المزيفة باستخدام التعلم الآلي Fake Currency Detection using Machine Learning

يُعد اكتشاف العملات المزيفة Fake Currency Detection مشكلة حقيقية لكل من الأفراد والشركات. يجد المزورون باستمرار طرقًا وتقنيات جديدة لإنتاج الأوراق النقدية المزيفة، والتي لا يمكن تمييزها أساسًا عن النقود الحقيقية. على الأقل للعين البشرية.في هذه المقالة، سأقدم لك الكشف عن العملات المزيفة باستخدام التعلم الآلي.

اكتشاف العملة المزيفة هي مهمة تصنيف ثنائيفي التعلم الآلي. إذا كانت لدينا بيانات كافية عن الأوراق النقدية الحقيقية والمزيفة، فيمكننا استخدام هذه البيانات لتدريب نموذج يمكنه تصنيف الأوراق النقدية الجديدة على أنها حقيقية أو مزيفة.

اكتشاف العملة المزيف

يمكن تنزيل مجموعة البيانات التي سأستخدمهافي هذه المهمة للكشف عن العملات المزيفة من هنا. تحتوي مجموعة البيانات على خصائص الإدخال الأربع التالية:

- تحول تباين variance الصورة إلى مويجات wavelets.
- تحول عدم تناسق asymmetry الصورة إلى مويجات wavelets.
- تحول تفرطح Kurtosis الصورة إلى مويجات wavelets.
- إنتروبيا الصورة.

القيمة المستهدفة هي ببساطة 0 للأوراق النقدية الحقيقية و1 للأوراق النقدية المزيفة. فلنبدأ الآن بمهمة اكتشاف العملة المزيفة باستخدام التعلم الآلي. سأبدأ هذه المهمة باستيراد الحزم الضرورية:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
```

الآن، دعنا نلقي نظرة على مجموعة البيانات، لا تحتوي البيانات على عناوين، لذا سأقوم أيضًا بتعيين عناوينفي العملية ثم سأطبع أول 5 صفوف من البيانات:

```
data = pd.read_csv('data_banknote_authentication.txt', header=None)
data.columns = ['var', 'skew', 'curt', 'entr', 'auth']
print(data.head())
```

```
      var    skew    curt     entr  auth
0  3.62160  8.6661 -2.8073 -0.44699     0
1  4.54590  8.1674 -2.4586 -1.46210     0
2  3.86600 -2.6383  1.9242  0.10645     0
3  3.45660  9.5228 -4.0112 -3.59440     0
4  0.32924 -4.4552  4.5718 -0.98880     0
```

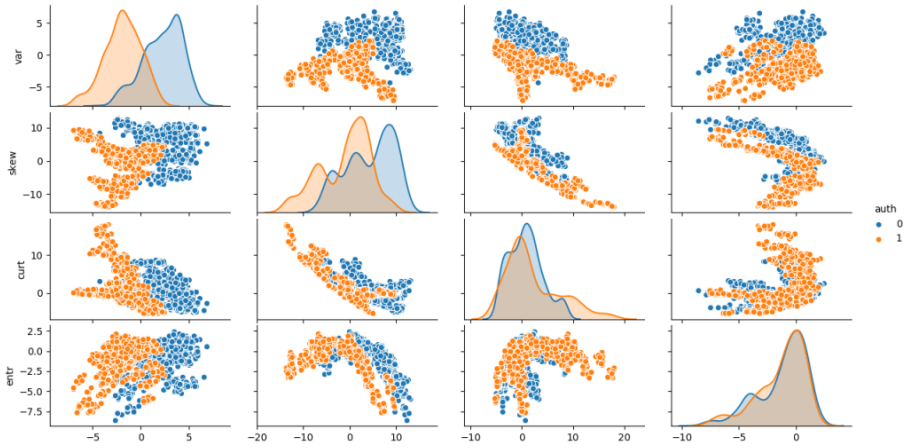
استكشاف البيانات

فلنبدأ الآنفي استكشاف مجموعة البيانات. أولاً، سأتحقق من أنواع البيانات وإذا كانت هناك أي قيم مفقودةفي البيانات:

```
print(data.info)
```

لذلك، ليس لدينا قيم مفقودةفي البيانات. يمكننا الآن رسم مخطط زوجي pair diagram للحصول على نظرة عامة على العلاقة بين جميع الكيانات. سأقوم أيضًا بتلوين الملاحظات: الأزرق للأوراق النقدية الأصلية والبرتقالي للأوراق النقدية المزيفة:

```
sns.pairplot(data, hue='auth')
plt.show()
```



من هذا المخطط الزوجي يمكننا تقديم عدة ملاحظات مثيرة للاهتمام:

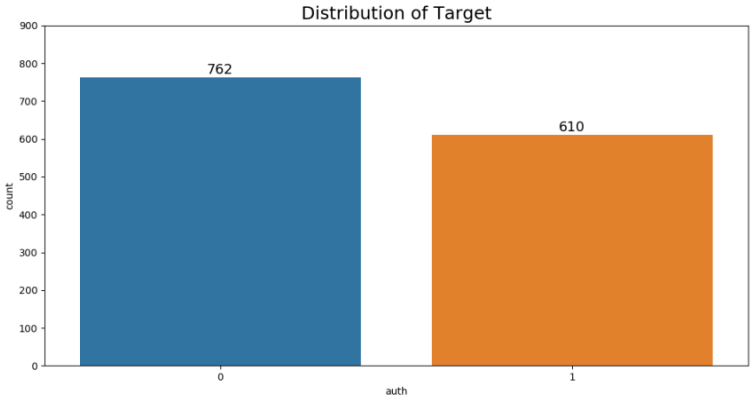
- يبدو أن توزيع كل من التباين والانحراف مختلف تمامًا بالنسبة للخاصيتين المستهدفتين، بينما يبدو التقوس والانتروبيا أكثر تشابهًا.
- هناك اتجاهات خطية وغير خطية واضحة في ميزات الإدخال.
- يبدو أن بعض الخصائص مترابطة.
- يبدو أن بعض الميزات تفصل بين الأوراق النقدية الأصلية والمزيفة جيداً.

دعنا الآن نتحقق مما إذا كانت بياناتنا متوازنة مع القيم المستهدفة:

```
plt.figure(figsize=(8,6))
```

```
plt.title('Distribution of Target', size=18)
sns.countplot(x=data['auth'])
target_count = data.auth.value_counts()
plt.annotate(s=target_count[0], xy=(-0.04,10+target_count[0]), size=14)
plt.annotate(s=target_count[1], xy=(0.96,10+target_count[1]), size=14)
plt.ylim(0,900)
plt.show()
```

مجموعة البيانات متوازنة إلى حد ما، ولكن بالنسبة لمهمة التصنيف الثنائي، نحتاج إلى موازنة ذلك تمامًا. لذلك دعونا نبدأفي المعالجة المسبقة للبيانات من خلال القيام بذلك بالضبط.



معالجة البيانات

الآن نحن بحاجة إلى موازنة بياناتنا، أسهل طريقة للقيام بذلك هي حذف عدد من مثيلات الدالة المستهدفة التي تم تمثيلها بشكل زائد بشكل عشوائي. هذا يسمى اختزال عشوائي random undersampling.

بخلاف ذلك، يمكننا أيضًا إنشاء بيانات تركيبية جديدة للفئة المستهدفة ذات التمثيل المنخفض. وهذا ما يسمى بالإفراطفي أخذ العينات oversampling.في الوقت الحالي، لنبدأ بحذف 152 ملاحظة للأوراق النقدية الفعلية بشكل عشوائي:

```
nb_to_delete = target_count[0] - target_count[1]
data = data.sample(frac=1, random_state=42).sort_values(by='auth')
data = data[nb_to_delete:]
print(data['auth'].value_counts())
```

```
1  610
0  610
Name: auth, dtype: int64
```

الآن لدينا مجموعة بيانات متوازنة تمامًا. بعد ذلك، نحتاج إلى تقسيم البيانات إلى مجموعات تدريب واختبار:

```
x = data.loc[:, data.columns != 'auth']
y = data.loc[:, data.columns == 'auth']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=42)
```

الآن سأقوم بتوحيد البيانات باستخدام طريقة StandardScalar المقدمة من Scikit-Learn:

```
scalar = StandardScaler()
scalar.fit(x_train)
x_train = scalar.transform(x_train)
x_test = scalar.transform(x_test)
```

الانحدار اللوجستي لاكتشاف العملات المزيفة

الآن، سأقوم بتدريب واختبار نموذجنا للكشف عن العملات المزيفة باستخدام خوارزمية الانحدار اللوجستي Logistic Regression. دعونا أولاً نلائم البياناتفي نموذج الانحدار اللوجستي لتدريب النموذج:

```
clf = LogisticRegression(solver='lbfgs', random_state=42,
multi_class='auto')
clf.fit(x_train, y_train.values.ravel())
```

دعنا الآن نختبر دقة نموذجنا:

```
y_pred = np.array(clf.predict(x_test))
conf_mat = pd.DataFrame(confusion_matrix(y_test, y_pred),
                        columns=["Pred.Negative", "Pred.Positive"],
                        index=['Act.Negative', "Act.Positive"])
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
accuracy = round((tn+tp)/(tn+fp+fn+tp), 4)
print(conf_mat)
print(f'\n Accuracy = {round(100*accuracy, 2)}%')
```

```
              Pred.Negative   Pred.Positive
Act.Negative     187              6
Act.Positive      0               173
Accuracy =  98.36%
```

حقق نموذج الانحدار اللوجستي دقة بلغت 98.36٪. وليس ذلك فحسب، فعندما توقع نموذجنا المزيف لاكتشاف العملة أن الأوراق النقدية حقيقية، كانت صحيحة بنسبة 100٪ من الوقت.

الآن دعونا نحاكي التنبؤ بورقة ورقية واحدة. كل ما نحتاج إلى القيام به هو استخراج الميزات وتوسيع نطاقها ودمجهافي نموذجنا المُدرَّب مسبقًا. يمكننا أيضًا فحص احتمالات الأوراق النقدية للانتماء إلى كل فئة مستهدفة:

```
new_banknote = np.array([4.5, -8.1, 2.4, 1.4], ndmin=2)
new_banknote = scalar.transform(new_banknote)
print(f'Prediction:  Class{clf.predict(new_banknote)[0]}')
print(f'Probability [0/1]:  {clf.predict_proba(new_banknote)[0]}')
```

```
Prediction: Class0
Probability [0/1]: [0.61112576 0.38887424]
```

الملخص

يتوقع نموذجنا أن هذه الورقة النقدية حقيقية. آمل أن تكون قد أحببت هذه المقالة حول الكشف عن العملات المزيفة باستخدام التعلم الآلي.

21) توقع سعر السهم باستخدام التعلم الآلي Stock Price Prediction using Machine Learning

في مشروع علوم البيانات هذا، سننشئ نموذجًا للانحدار الخطي Linear Regression ونموذجًا لانحدار شجرة القرار Decision Tree Regression للتنبؤ بسعر سهم Apple باستخدام التعلم الآلي وPython.

استيراد pandas لاستيراد ملف CSV:

```
import pandas as pd
apple = pd.read_csv("AAPL.csv")
print(apple.head())
```

	Date	Open Price	High	...	Close Price	Adj Close Price	Volume
0	5/13/2019	187.710007	189.479996	...	185.720001	183.529678	57430600
1	5/14/2019	186.410004	189.699997	...	188.660004	186.434998	36529700
2	5/15/2019	186.270004	191.750000	...	190.919998	188.668335	26544700
3	5/16/2019	189.910004	192.470001	...	190.080002	187.838257	33031400
4	5/17/2019	186.929993	190.899994	...	189.000000	186.770996	32879100

للحصول على عدد أيام التدريب:

```
print("trainging days =",apple.shape)
```

```
#Output- training days = (251, 7)
```

لرسم بيانات سعر الإغلاق:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
plt.figure(figsize=(10, 4))
plt.title("Apple's Stock Price")
plt.xlabel("Days")
plt.ylabel("Close Price USD ($)")
plt.plot(apple["Close Price"])
plt.show()
```

للحصول على سعر الإغلاق:

```
apple = apple[["Close Price"]]
print(apple.head())
```

```
   Close Price
0   185.720001
1   188.660004
2   190.919998
3   190.080002
4   189.000000
```

إنشاء متغير للتنبؤ بـ "X" أيامفي المستقبل:

```
futureDays = 25
```

إنشاء عمود هدف جديد مع تحويل "X" وحدات / أيام لأعلى:

```
apple["Prediction"] = apple[["Close Price"]].shift(-futureDays)
print(apple.head())
print(apple.tail())
```

```
   Close Price  Prediction
0   185.720001  198.449997
1   188.660004  197.869995
2   190.919998  199.460007
3   190.080002  198.779999
4   189.000000  198.580002
     Close Price  Prediction
246   293.160004         NaN
247   297.559998         NaN
248   300.630005         NaN
249   303.739990         NaN
250   310.130005         NaN
```

لإنشاء مجموعة بيانات ميزة (X) وتحويلها إلى مصفوفة عددية وإزالة "X" من الصفوف / الأيام الأخيرة:

```
import numpy as np
x = np.array(apple.drop(["Prediction"], 1))[:-futureDays]
print(x)
```

```
#Output-

[[185.720001]
[188.660004]
[190.919998]
[190.080002]
[189. ]
[183.089996]
[186.600006]
[182.779999]
[179.660004]
[178.970001]
[178.229996]
[177.380005]
[178.300003]
[175.070007]
[173.300003]
```

لإنشاء مجموعة بيانات مستهدفة (y) وتحويلها إلى مصفوفة عددية والحصول على جميع القيم المستهدفة باستثناء آخر "x" من أيام الصفوف:

```
y = np.array(apple["Prediction"])[:-futureDays]
print(y)
```

#Output-

[198.449997 197.869995 199.460007 198.779999 198.580002 195.570007
199.800003 199.740005 197.919998 201.550003 202.729996 204.410004
204.229996 200.020004 201.240005 203.229996 201.75 203.300003
205.210007 204.5 203.350006 205.660004 202.589996 207.220001
208.839996 208.669998 207.020004 207.740005 209.679993 208.779999
213.039993 208.429993 204.020004 193.339996 197. 199.039993
203.429993 200.990005 200.479996 208.970001 202.75 201.740005
206.5 210.350006 210.360001 212.639999 212.460007 202.639999
206.490005 204.160004 205.529999 209.009995 208.740005 205.699997
209.190002 213.279999 213.259995 214.169998 216.699997 223.589996
223.089996 218.75 219.899994 220.699997 222.770004 220.960007
217.729996 218.720001 217.679993 221.029999 219.889999 218.820007
223.970001 224.589996 218.960007 220.820007 227.009995 227.059998

تقسيم البيانات

قسّم البيانات إلى 75٪ تدريب و25٪ اختبار.

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25)
```

إنشاء النماذج

إنشاء نموذج انحدار لشجرة القرار.

```
 #Creating the decision tree regressor model
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor().fit(xtrain, ytrain)

 #creating the Linear Regression model
from sklearn.linear_model import LinearRegression
linear = LinearRegression().fit(xtrain, ytrain)
```

للحصول على "x" من الصفوف / الأيام الأخيرة لمجموعة بيانات الميزة:

```
xfuture = apple.drop(["Prediction"], 1)[:-futureDays]
xfuture = xfuture.tail(futureDays)
xfuture = np.array(xfuture)
print(xfuture)
```

```
#Output-

[[273.359985]
[298.809998]
[289.320007]
[302.73999 ]
[292.920013]
[289.029999]
[266.170013]
[285.339996]
[275.429993]
[248.229996]
[277.970001]
[242.210007]
[252.860001]
```

التنبؤ

لمشاهدة تنبؤ نموذج شجرة القرار:

```
treePrediction = tree.predict(xfuture)
print("Decision Tree prediction =",treePrediction)
```

لمشاهدة نموذج تنبؤ الانحدار الخطي:

```
linearPrediction = linear.predict(xfuture)
print("Linear regression Prediction =",linearPrediction)
```

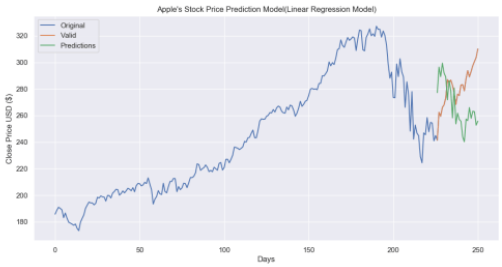
رسم التنبؤات

لنرسم تنبؤات شجرة القرار:

```
predictions = treePrediction
valid = apple[x.shape[0]:]
valid["Predictions"] = predictions
plt.figure(figsize=(10, 6))
plt.title("Apple's Stock Price Prediction Model(Decision Tree Regressor
Model)")
plt.xlabel("Days")
plt.ylabel("Close Price USD ($)")
plt.plot(apple["Close Price"])
plt.plot(valid[["Close Price", "Predictions"]])
plt.legend(["Original", "Valid", "Predictions"])
plt.show()
```

لنرسم تنبؤات النموذج الخطي

```
predictions = linearPrediction
valid = apple[x.shape[0]:]
valid["Predictions"] = predictions
plt.figure(figsize=(10, 6))
plt.title("Apple's Stock Price Prediction Model(Linear Regression Model)")
plt.xlabel("Days")
plt.ylabel("Close Price USD ($)")
plt.plot(apple["Close Price"])
plt.plot(valid[["Close Price", "Predictions"]])
plt.legend(["Original", "Valid", "Predictions"])
plt.show()
```



22) توقع الطقس باستخدام التعلم الآلي Predict Weather using Machine Learning

في هذه المقالة، سأقوم بتدريب نموذج للتنبؤ بالطقس باستخدام التعلم الآلي. سنتصرف كما لو أننا لا نستطيع الوصول إلى توقعات الطقس. لدينا إمكانية الوصول إلى قرن من المتوسطات التاريخية لدرجات الحرارة العالمية، بمافي ذلك درجات الحرارة العالمية القصوى، ودرجات الحرارة الدنيا العالمية، ودرجات حرارة الأرض والمحيطات العالمية. بعد كل هذا، نعلم أن هذه مشكلة تعلم آلة انحدار خاضعة للإشراف.

مجموعة بيانات الطقس للتنبؤ بالطقس

بادئ ذي بدء، نحتاج إلى بعض البيانات، تم إنشاء البيانات التي أستخدمها للتنبؤ بالطقس باستخدام التعلم الآلي من إحدى الجامعات البحثية المرموقةفي العالم، وسنفترض أن البيانات الموجودةفي مجموعة البيانات صحيحة. يمكنك بسهولة تنزيل هذه البيانات من هنا.

الآن، دعنا نبدأفي قراءة مجموعة البيانات:

```
import pandas as pd
global_temp = pd.read_csv("GlobalTemperatures.csv")
print(global_temp.shape)
print(global_temp.columns)
print(global_temp.info())
print(global_temp.isnull().sum())
```

تحضير البيانات

لسوء الحظ، لم نصل تمامًا إلى المرحلة التي يمكننا فيها فقط إدخال البيانات الأوليةفي نموذج وجعله يرسل ردًا. سنحتاج إلى إجراء بعض التعديلات الطفيفة لوضع بياناتنافي نموذج التعلم الآلي.

ستعتمد الخطوات الدقيقةفي إعداد البيانات على النموذج المستخدم والبيانات التي تم جمعها، ولكن ستكون هناك حاجة إلى قدر من معالجة البيانات. أولاً، سأُنشئ دالة تسمى wrangle () ساستدعي فيها على dataframe:

```
#Data Preparation
def wrangle(df):
    df = df.copy()
    df = df.drop(columns=["LandAverageTemperatureUncertainty",
"LandMaxTemperatureUncertainty",
                          "LandMinTemperatureUncertainty",
"LandAndOceanAverageTemperatureUncertainty"], axis=1)
```

نريد عمل نسخة من إطار البيانات حتى لا نتلف الأصل. بعد ذلك، سنقوم بإزالة الأعمدة ذات العلاقة الأساسية العالية high cardinality.

تشير العلاقة الأساسية العالية إلى الأعمدة التي تكون قيمها نادرة جدًا أو فريدة. نظرًا لتكرار البيانات الأساسية العاليةفي معظم مجموعات بيانات السلاسل الزمنية، سنحل هذه المشكلة مباشرةً عن طريق إزالة أعمدة عدد العناصر العالية هذه تمامًا من مجموعة البيانات الخاصة بنا حتى لا يتم الخلط بين نموذجنافي المستقبل.

الآن، سأقوم بإنشاء دالة لتحويل درجة الحرارة، ولتحويل الأعمدة إلى كائن DateTime:

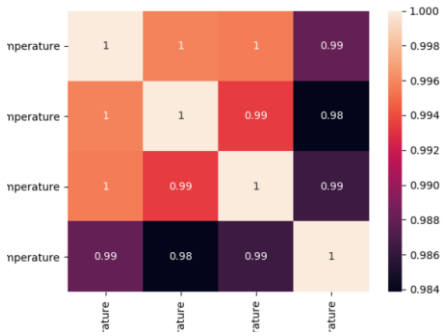
```
def converttemp(x):
        x = (x * 1.8) + 32
        return float(x)
    df["LandAverageTemperature"] =
df["LandAverageTemperature"].apply(converttemp)
    df["LandMaxTemperature"] = df["LandMaxTemperature"].apply(converttemp)
    df["LandMinTemperature"] = df["LandMinTemperature"].apply(converttemp)
    df["LandAndOceanAverageTemperature"] =
df["LandAndOceanAverageTemperature"].apply(converttemp)
    df["dt"] = pd.to_datetime(df["dt"])
    df["Month"] = df["dt"].dt.month
    df["Year"] = df["dt"].dt.year
    df = df.drop("dt", axis=1)
    df = df.drop("Month", axis=1)
    df = df[df.Year &gt;= 1850]
    df = df.set_index(["Year"])
    df = df.dropna()
    return df
global_temp = wrangle(global_temp)
print(global_temp.head())
```

بعد استدعاء دالة wrangle الخاصة بنا إلى global_temp dataframe، يمكننا الآن رؤية نسخة منظَّفة جديدة من إطار البيانات global_temp الخاص بنا بدون قيم مفقودة.

رسم البيانات

الآن، قبل المضي قدمًافي تدريب نموذج للتنبؤ بالطقس باستخدام التعلم الآلي، دعنا نرسم هذه البيانات للعثور على الارتباطات بين البيانات:

```
import seaborn as sns
import matplotlib.pyplot as plt
corrMatrix = global_temp.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```



كما يمكننا أن نرى، وكما قد خمّن بعضكم على الأرجح، فإن الأعمدة التي اخترناها للاستمرار في المضي قدمًا مرتبطة ارتباطًا وثيقًا ببعضها البعض.

فصل هدفنا للتنبؤ بالطقس

نحتاج الآن إلى فصل البيانات إلى ميزات وأهداف. الهدف، المسمى أيضًا Y، هو القيمة التي نريد توقعها،في هذه الحالة، المتوسط الفعلي لدرجة حرارة الأرض والمحيطات والميزات هي جميع الأعمدة التي يستخدمها النموذج لإجراء التنبؤ:

```
target = "LandAndOceanAverageTemperature"
y = global_temp[target]
x = global_temp[["LandAverageTemperature", "LandMaxTemperature",
"LandMinTemperature"]]
```

التقسيم الى بيانات تدريب واختبار

الآن، لإنشاء نموذج للتنبؤ بالطقس باستخدام التعلم الآلي، نحتاج إلى تقسيم البيانات باستخدام طريقة train_test_split المقدمة من scikit-Learn:

```
from sklearn.model_selection import train_test_split
xtrain, xval, ytrain, yval = train_test_split(x, y, test_size=0.25,
random state=42)
print(xtrain.shape)
print(xval.shape)
print(ytrain.shape)
print(yval.shape)
```

```
(1494, 3)
(498, 3)
(1494,)
(498,)
```

خط أساس متوسط الخطأ المطلق

قبل أن نتمكن من إجراء وتقييم أي تنبؤات على نموذج التعلم الآلي الخاص بنا للتنبؤ بالطقس، نحتاج إلى إنشاء خط أساس Baseline، وهو مقياس عاقل نأملفي التغلب عليه باستخدام نموذجنا. إذا لم يتمكن نموذجنا من التحسن من خط الأساس، فسوف يفشل ويجب أن نجرب نموذجًا مختلفًا أو نعترف بأن التعلم الآلي غير مناسب لمشكلتنا:

```
from sklearn.metrics import mean_squared_error
ypred = [ytrain.mean()] * len(ytrain)
print("Baseline MAE: ", round(mean_squared_error(ytrain, ypred), 5))
```

تدريب النموذج للتنبؤ بالطقس

الآن للتنبؤ بالطقس باستخدام التعلم الآلي، سأقوم بتدريب خوارزمية Random Forest قادرة على أداء مهام التصنيف وكذلك الانحدار:

```
from sklearn.feature_selection import SelectKBest
from sklearn.ensemble import RandomForestRegressor
forest = make_pipeline(
```

```
    SelectKBest(k="all"),
    StandardScaler(),
    RandomForestRegressor(
        n_estimators=100,
        max_depth=50,
        random_state=77,
        n_jobs=-1
    )
)
forest.fit(xtrain, ytrain)
```

تقييم نموذج التعلم الآلي للتنبؤ بالطقس

لوضع تنبؤاتنافي منظورها الصحيح، يمكننا حساب precision باستخدام متوسط النسبة المئوية للخطأ مطروحًا من 100٪:

```
import numpy as np
errors = abs(ypred - yval)
mape = 100 * (errors/ytrain)
accuracy = 100 - np.mean(mape)
print("Random Forest Model: ", round(accuracy, 2), "%")
```

```
Random Forest Model: 99.52 %
```

الملخص

لقد تعلم نموذجنا التنبؤ بالظروف الجوية باستخدام التعلم الآلي للعام المقبل بدقة تصل إلى 99٪. آمل أن تكون قد أحببت هذه المقالة حول كيفية بناء نموذج للتنبؤ بالطقس باستخدام التعلم الآلي.

23) التنبؤ بالزلازل باستخدام التعلم الآلي Earthquake Prediction using Machine Learning

في هذه المقالة، سأطلعك على كيفية إنشاء نموذج لمهمة توقع الزلازل باستخدام التعلم الآلي ولغة برمجة بايثون. يُعد التنبؤ بالزلازل أحد أكبر المشكلات التي لم يتم حلهافي علوم الأرض.

مع زيادة استخدام التكنولوجيا، زادت العديد من محطات المراقبة الزلزالية، لذلك يمكننا استخدام التعلم الآلي والأساليب الأخرى التي تعتمد على البيانات للتنبؤ بالزلازل.

نموذج التنبؤ بالزلازل مع التعلم الآلي

من المعروف أنه إذا حدثت كارثةفي منطقة ما، فمن المحتمل أن تحدث مرة أخرى. بعض المناطق بها زلازل متكررة، ولكن هذا ليس سوى مبلغ مقارن مقارنة بالمناطق الأخرى.

لذلك، فإن التنبؤ بالزلزال مع التاريخ والوقت وخط العرض وخط الطول من البيانات السابقة ليس اتجاهًا يتبع مثل الأشياء الأخرى، إنه يحدث بشكل طبيعي.

سأبدأ هذه المهمة لإنشاء نموذج للتنبؤ بالزلازل عن طريق استيراد مكتبات Python الضرورية:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

فلنقم الآن بتحميل مجموعة البيانات وقراءتها. يمكن تنزيل مجموعة البيانات التي أستخدمها هنا بسهولة من هنا:

```
data = pd.read csv("database.csv")
data.columns
```

```
Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth Error',
       'Depth Seismic Stations', 'Magnitude', 'Magnitude Type',
       'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
       'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',
       'Source', 'Location Source', 'Magnitude Source', 'Status'],
      dtype='object')
```

المعالجة المسبقة للبيانات

الآن دعنا نرى الميزات الرئيسية لبيانات الزلازل وننشئ كائنًا من هذه الميزات، أي التاريخ date والوقت time وخط العرض latitude وخط الطول longitude والعمق depth والحجم magnitude:

```
data = data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth',
'Magnitude']]
data.head()
```

	date	Time	Latitude	Longitude	Depth	Magnitude
0	01/02/1965	13:44:18	19.246	145.616	131.6	6.0
1	01/04/1965	11:29:49	1.863	127.352	80.0	5.8
2	01/05/1965	18:05:58	-20.579	-173.972	20.0	6.2
3	01/08/1965	18:49:43	-59.076	-23.557	15.0	5.8
4	01/09/1965	13:32:50	11.938	126.427	15.0	5.8

نظرًا لأن البيانات عشوائية، فنحن بحاجة إلى قياسها بناءً على مدخلات النموذج.في هذا، نقوم بتحويل التاريخ والوقت المحددين إلى وقت Unix وهو بالثواني ورقم. يمكن استخدام هذا بسهولة كمدخل للشبكة التي أنشأناها:

```
import datetime
import time

timestamp = []
for d, t in zip(data['Date'], data['Time']):
    try:
        ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S')
        timestamp.append(time.mktime(ts.timetuple()))
    except ValueError:
        # print('ValueError')
        timestamp.append('ValueError')
timeStamp = pd.Series(timestamp)
data['Timestamp'] = timeStamp.values
final_data = data.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'ValueError']
final_data.head()
```

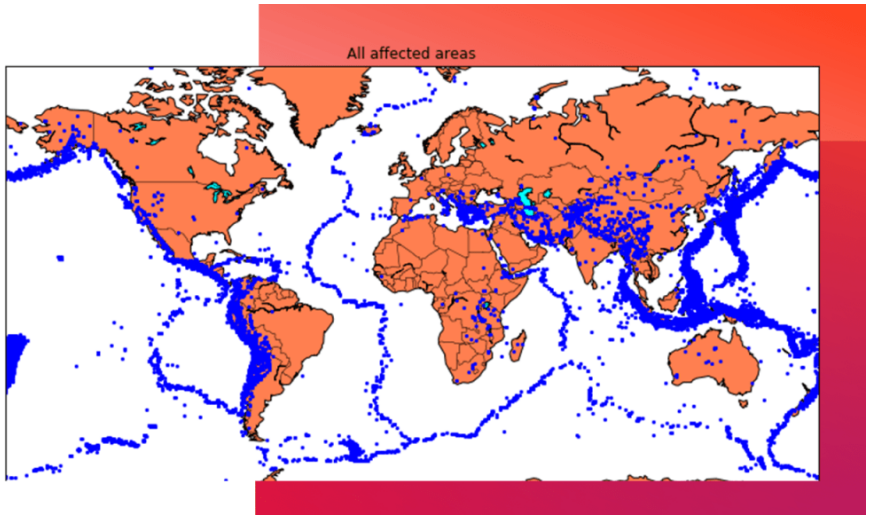
	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	-1.57631e+08
1	1.863	127.352	80.0	5.8	-1.57466e+08
2	-20.579	-173.972	20.0	6.2	-1.57356e+08
3	-59.076	-23.557	15.0	5.8	-1.57094e+08
4	11.938	126.427	15.0	5.8	-1.57026e+08

العرض المرئي للبيانات

الآن، قبل إنشاء نموذج التنبؤ بالزلازل، دعنا نرسم البيانات الموجودة على خريطة العالم التي تعرض تمثيلًا واضحًا لمكان تواتر الزلزال:

```
from mpl_toolkits.basemap import Basemap

m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-
180,urcrnrlon=180,lat_ts=20,resolution='c')

longitudes = data["Longitude"].tolist()
latitudes = data["Latitude"].tolist()
#m = Basemap(width=12000000,height=9000000,projection='lcc',
```

```
            #resolution=None,lat_1=80.,lat_2=55,lat_0=80,lon_0=-107.)
x,y = m(longitudes,latitudes)

fig = plt.figure(figsize=(12,10))
plt.title("All affected areas")
m.plot(x, y, "o", markersize = 2, color = 'blue')
m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
m.drawmapboundary()
m.drawcountries()
plt.show()
```



تقسيم مجموعة البيانات

الآن، لإنشاء نموذج التنبؤ بالزلازل، نحتاج إلى تقسيم البيانات إلى Xs و ys والتي سيتم إدخالها على التوالي في النموذج كمدخلات لتلقي الإخراج من النموذج.

المدخلات هنا هي TImestamp وLatitude وLongitude والمخرجات هي Magnitude وDepth. سأقوم بتقسيم x و y إلى تدريب واختبار مع التحقق من الصحة. تحتوي مجموعة التدريب على 80٪ ومجموعة الاختبار تحتوي على 20٪:

```
X = final_data[['Timestamp', 'Latitude', 'Longitude']]
y = final_data[['Magnitude', 'Depth']]
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, X_test.shape)
```

```
(18727, 3) (4682, 3) (18727, 2) (4682, 3)
```

الشبكة العصبية للتنبؤ بالزلازل

الآن سوف أقوم بإنشاء شبكة عصبية لتناسب البيانات من مجموعة التدريب. ستتألف شبكتنا العصبية من ثلاث طبقات كثيفة dense layers تحتوي كل منها على 2 ,16 ,16عقدة وتعيد قراءتها. سيتم استخدام Relu وsoftmax كدوال تنشيط:

```
from keras.models import Sequential
from keras.layers import Dense

def create_model(neurons, activation, optimizer, loss):
    model = Sequential()
    model.add(Dense(neurons, activation=activation, input_shape=(3,)))
    model.add(Dense(neurons, activation=activation))
    model.add(Dense(2, activation='softmax'))

    model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])

    return model
```

سأقوم الآن بتحديد المعلمات الفائقة hyperparameters بخيارين أو أكثر للعثور على أفضل ملاءمة best fit:

```
from keras.wrappers.scikit_learn import KerasClassifier

model = KerasClassifier(build_fn=create_model, verbose=0)

# neurons = [16, 64, 128, 256]
neurons = [16]
# batch_size = [10, 20, 50, 100]
batch_size = [10]
epochs = [10]
# activation = ['relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear',
'exponential']
activation = ['sigmoid', 'relu']
# optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax',
'Nadam']
optimizer = ['SGD', 'Adadelta']
loss = ['squared_hinge']

param_grid = dict(neurons=neurons, batch_size=batch_size, epochs=epochs,
activation=activation, optimizer=optimizer, loss=loss)
```

نحتاج الآن إلى العثور على أفضل نموذج ملائم للنموذج أعلاه والحصول على متوسط درجة الاختبار والانحراف المعياري لأفضل نموذج مناسب:

```
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
grid_result = grid.fit(X_train, y_train)

print("Best: %f using %s" % (grid_result.best_score_,
grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: 0.957655 using {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss':
'squared_hinge', 'neurons': 16, 'optimizer': 'SGD'} 0.333316 (0.471398) with: {'activation':
'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer':
'SGD'} 0.000000 (0.000000) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss':
'squared_hinge', 'neurons': 16, 'optimizer': 'Adadelta'} 0.957655 (0.029957) with: {'activation':
'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer': 'SGD'}
0.645111 (0.456960) with: {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss':
'squared_hinge', 'neurons': 16, 'optimizer': 'Adadelta'}
```

في الخطوة أدناه، يتم استخدام أفضل المعلمات الملائمة لنفس النموذج لحساب النتيجة باستخدام بيانات التدريب وبيانات الاختبار:

```
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(3,)))
model.add(Dense(16, activation='relu'))
model.add(Dense(2, activation='softmax'))

model.compile(optimizer='SGD', loss='squared_hinge', metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=10, epochs=20, verbose=1,
validation_data=(X_test, y_test))

[test_loss, test_acc] = model.evaluate(X_test, y_test)
print("Evaluation result on Test Data : Loss = {}, accuracy =
{}".format(test_loss, test_acc))
```

```
Evaluation result on Test Data : Loss = 0.5038455790406056, accuracy = 0.9241777017858995
```

الملخص

لذلك يمكننا أن نرىفي الناتج أعلاه أن نموذج الشبكة العصبية الخاص بنا للتنبؤ بالزلازل يعمل بشكل جيد. آمل أن تكون قد أحببت هذه المقالة حول كيفية إنشاء نموذج التنبؤ بالزلازل باستخدام التعلم الآلي ولغة برمجة Python.

24) توقع وجود أمراض القلب باستخدام التعلم الآلي Predicting presence of Heart Diseases using Machine Learning

يُستخدم التعلم الآليفي العديد من المجالات حول العالم. صناعة الرعاية الصحية ليست استثناء. يمكن أن يلعب التعلم الآلي دورًا أساسيًافي توقع وجود / عدم وجود اضطرابات حركية وأمراض القلب وغير ذلك. يمكن لمثل هذه المعلومات، إذا تم التنبؤ بها مسبقًا، أن توفر رؤى مهمة للأطباء الذين يمكنهم بعد ذلك تكييف تشخيصهم وعلاجهم على أساس كل مريض.

في هذه المقالة، سأناقش مشروعًا عملت فيه على التنبؤ بأمراض القلب المحتملة لدى الأشخاص الذين يستخدمون خوارزميات التعلم الآلي. تضمنت الخوارزميات مصنف KNN ومصنف SVM ومصنف شجرة القرار Decision Tree ومصنف الغابات العشوائية Random Forest. تم أخذ مجموعة البيانات من Kaggle. مشروعي الكامل متاحفي Heart Disease Prediction.

استيراد المكتبات

قمت باستيراد عدة مكتبات للمشروع:

- numpy: للعمل مع المصفوفات.
- pandas: للعمل مع ملفات csv وأطر البيانات.
- matplotlib: لإنشاء مخططات باستخدام pyplot ، حدد المعلمات باستخدام rcParams وقم بتلوينها باستخدام cm.rainbow.
- warnings: لتجاهل جميع التحذيرات التي قد تظهرفي النوتبوك بسبب الاستهلاك السابق / المستقبلي للميزة.
- train_test_split: لتقسيم مجموعة البيانات إلى بيانات تدريب واختبار.
- StandardScaler: لتوسيع نطاق جميع الميزات، بحيث يتكيف نموذج التعلم الآلي بشكل أفضل مع مجموعة البيانات.

بعد ذلك، قمت باستيراد جميع خوارزميات التعلم الآلي الضرورية.

```
# Basic
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

# Other libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
# Machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

استيراد مجموعة البيانات

بعد تنزيل مجموعة البيانات من Kaggle، قمت بحفظهافي دليل العمل الخاص بي باسم dataset.csv. بعد ذلك، استخدمت read_csv() لقراءة مجموعة البيانات وحفظهافي متغير مجموعة البيانات.

قبل أي تحليل، أردت فقط إلقاء نظرة على البيانات. لذلك، استخدمت طريقة info().

```
dataset.info()
```

```
#################
#### OUTPUT ####
#################
# <class 'pandas.core.frame.DataFrame'>
# RangeIndex: 303 entries, 0 to 302
# Data columns (total 14 columns):
# age          303 non-null int64
# sex          303 non-null int64
# cp           303 non-null int64
# trestbps     303 non-null int64
# chol         303 non-null int64
# fbs          303 non-null int64
# restecg      303 non-null int64
# thalach      303 non-null int64
# exang        303 non-null int64
# oldpeak      303 non-null float64
# slope        303 non-null int64
# ca           303 non-null int64
# thal         303 non-null int64
# target       303 non-null int64
# dtypes: float64(1), int64(13)
# memory usage: 33.2 KB
```

كما ترى من الإخراج أعلاه، هناك إجمالي 13 ميزة ومتغير هدف واحد. أيضًا، لا توجد قيم مفقودة، لذلك لا نحتاج إلى الاهتمام بأي قيم فارغة. بعد ذلك، استخدمت طريقة describe().

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.31
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.61
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.0
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.0
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.0
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.0

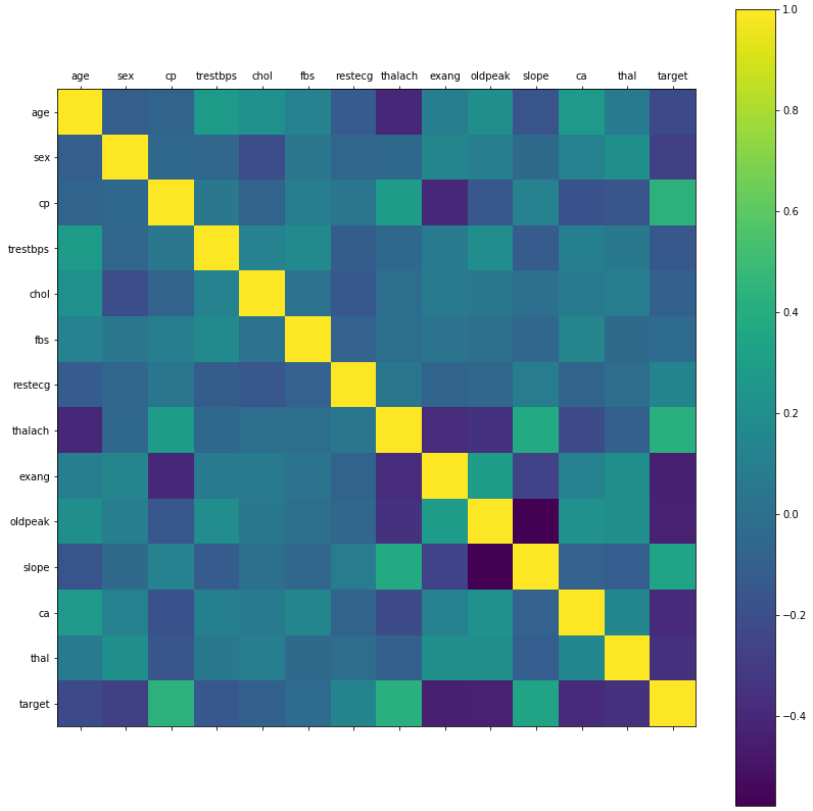
كشفت الطريقة أن نطاق كل متغير مختلف. الحد الأقصى لقيمة العمر age هو 77 ولكن بالنسبة للكوليسترول chol هو 564. وبالتالي، يجب إجراء تحجيم الميزة على مجموعة البيانات.

فهم البيانات

مصفوفة الارتباط

بادئ ذي بدء، دعنا نرى مصفوفة ارتباط correlation matrix الميزات ونحاول تحليلها. يتم تحديد حجم الشكل إلى 12 × 8 باستخدام rcParams. ثم استخدمت pyplot لإظهار مصفوفة الارتباط. باستخدام xticks و yticks ، أضفت أسماء إلى مصفوفة الارتباط. يظهر colorbar() شريط الألوان للمصفوفة.

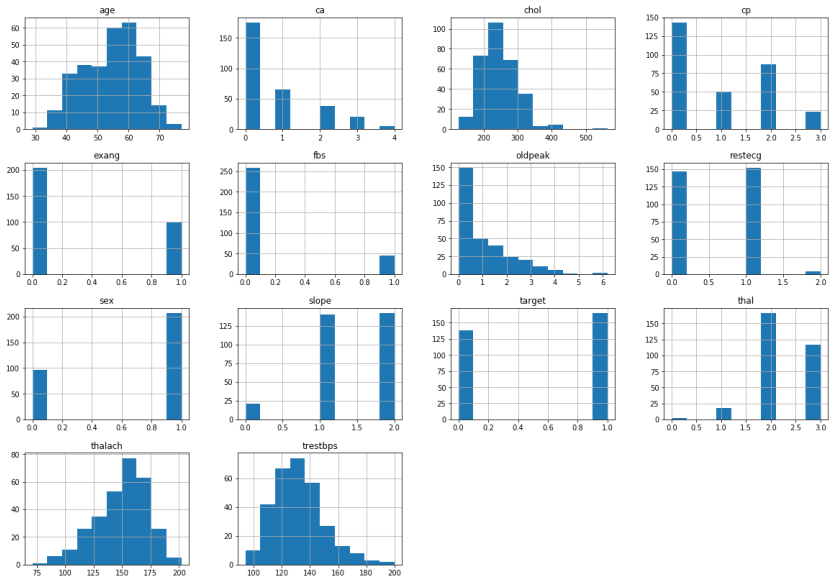
```
rcParams['figure.figsize'] = 20, 14
plt.matshow(dataset.corr())
plt.yticks(np.arange(dataset.shape[1]), dataset.columns)
plt.xticks(np.arange(dataset.shape[1]), dataset.columns)
plt.colorbar()
```



من السهل ملاحظة أنه لا توجد ميزة واحدة لها ارتباط كبير جدًا بالقيمة المستهدفة. أيضًا، بعض الميزات لها ارتباط سلبي مع القيمة المستهدفة وبعضها لها علاقة إيجابية.

الهستوكرام

أفضل جزءفي هذا النوع من المخطط هو أنه لا يتطلب الأمر سوى أمر واحد لرسم المخططات ويوفر الكثير من المعلوماتفي المقابل. فقط استخدم dataset.hist().



دعونا نلقي نظرة على المخططات. يوضح كيف يتم توزيع كل ميزة وتسمية على نطاقات مختلفة، مما يؤكد أيضًا الحاجة إلى القياس. بعد ذلك، أينما ترى أشرطة متقطعة discrete bars، فهذا يعني بشكل أساسي أن كل منها هوفي الواقع متغير فئوي. سنحتاج إلى التعامل مع هذه المتغيرات الفئوية قبل تطبيق التعلم الآلي. تحتوي التسميات المستهدفة لدينا على فئتين، صفر لعدم وجود مرض وواحد للمرض.

المخطط الشريطي للفئة المستهدفة

من الضروري حقًا أن تكون مجموعة البيانات التي نعمل عليها متوازنة تقريبًا. يمكن لمجموعة البيانات غير المتوازنة للغاية أن تجعل تدريب النموذج بأكمله عديم الفائدة، وبالتالي لن تكون ذات فائدة. دعونا نفهمها بمثال.

لنفترض أن لدينا مجموعة بيانات من 100 شخص مع 99 من غير المرضى ومريض واحد. بدون تدريب وتعلم أي شيء، يمكن للنموذج دائمًا أن يقول إن أي شخص جديد سيكون غير مريض وبدقة تبلغ 99٪. ومع ذلك، نظرًا لأننا مهتمون أكثر بتحديد الشخص الواحد الذي هو مريض، فنحن بحاجة إلى مجموعات بيانات متوازنة حتى يتعلم نموذجنا بالفعل.

```
rcParams['figure.figsize'] = 8,6
plt.bar(dataset['target'].unique(), dataset['target'].value_counts(),
color = ['red', 'green'])
plt.xticks([0, 1])
plt.xlabel('Target Classes')
plt.ylabel('Count')
plt.title('Count of each Target Class')
```

بالنسبة للمحور السيني، استخدمت unique() من العمود الهدف ثم قمت بتعيين اسمها باستخدام xticks. بالنسبة لمحور y، استخدمت value_count() للحصول على قيم كل فئة. لقد قمت بتلوين الاشرطة باللون الأخضر والأحمر.

من المخطط، يمكننا أن نرى أن الفئات متوازنة تقريبًا ونحن جيدون للمضي قدمًافي معالجة البيانات.

معالجة البيانات

للعمل مع المتغيرات الفئوية، يجب علينا تقسيم كل عمود فئوي إلى أعمدة وهمية مع الواحدات 1s والاصفار 0s.

لنفترض أن لدينا عمود الجنس Gender، بقيمتين 1 لـ للذكور و 0 للإناث. يجب تحويله إلى عمودين بالقيمة 1 حيث سيكون العمود صحيحًا و 0 حيث سيكون خطأ. ألق نظرة على الجيست Gist أدناه.

```
# Original Columm
# |  Gender  |
# |     1    |
# |     1    |
# |     0    |

# Dummy Columns
# |  Gender_0  ||  Gender_1  |
# |      0     ||      1     |
# |      0     ||      1     |
# |      1     ||      0     |
```

للقيام بذلك، نستخدم طريقة get_dummies() من pandas. بعد ذلك، نحتاج إلى توسيع نطاق مجموعة البيانات التي سنستخدم StandardScaler من أجلها. تقوم طريقة fit_transform()الخاصة بالمقياس بقياس البيانات ونقوم بتحديث الأعمدة.

```
dataset = pd.get_dummies(dataset, columns = ['sex', 'cp', 'fbs',
'restecg', 'exang', 'slope', 'ca', 'thal'])
```

الآن، سأستخدم StandardScaler من sklearn لتوسيع نطاق مجموعة البيانات الخاصة بي.

```
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] =
standardScaler.fit_transform(dataset[columns_to_scale])
```

مجموعة البيانات جاهزة الآن. يمكننا أن نبدأ بتدريب نماذجنا.

التعلم الالي

في هذا المشروع، أخذت 4 خوارزميات وقمت بتغيير معاييرها المختلفة وقارنت النماذج النهائية. لقد قسمت مجموعة البيانات إلى 67٪ بيانات تدريب و33٪ بيانات اختبار.

```
y = dataset['target']
X = dataset.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.33, random_state = 0)
```

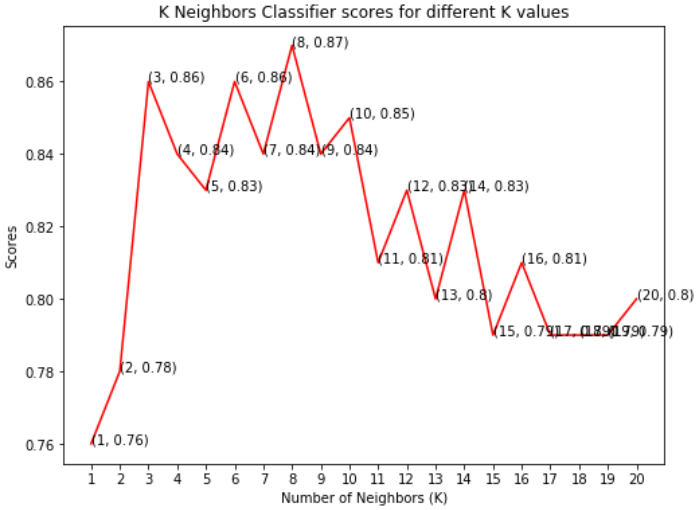
مصنف KNN

يبحث هذا المصنف عن فئات K الأقرب للجيران لنقطة بيانات معينة وبناءً على فئة الأغلبية، فإنه يقوم بتعيين فئة لنقطة البيانات هذه. ومع ذلك، يمكن أن يتنوع عدد الجيران. قمت بتنويعهم من 1 إلى 20 من الجيران وحساب درجة الاختبارفي كل حالة.

```
knn_scores = []
for k in range(1,21):
    knn_classifier = KNeighborsClassifier(n_neighbors = k)
    knn_classifier.fit(X_train, y_train)
    knn_scores.append(knn_classifier.score(X_test, y_test))
```

بعد ذلك، أرسم رسمًا بيانيًا خطيًا لعدد الجيران ودرجة الاختبار التي تم تحقيقهافي كل حالة.

```
plt.plot([k for k in range(1, 21)], knn_scores, color = 'red')
for i in range(1,21):
    plt.text(i, knn_scores[i-1], (i, knn_scores[i-1]))
plt.xticks([i for i in range(1, 21)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('K Neighbors Classifier scores for different K values')
```



كما ترى، حققنا الحد الأقصى للدرجة 87٪ عندما تم اختيار عدد الجيران ليكون 8.

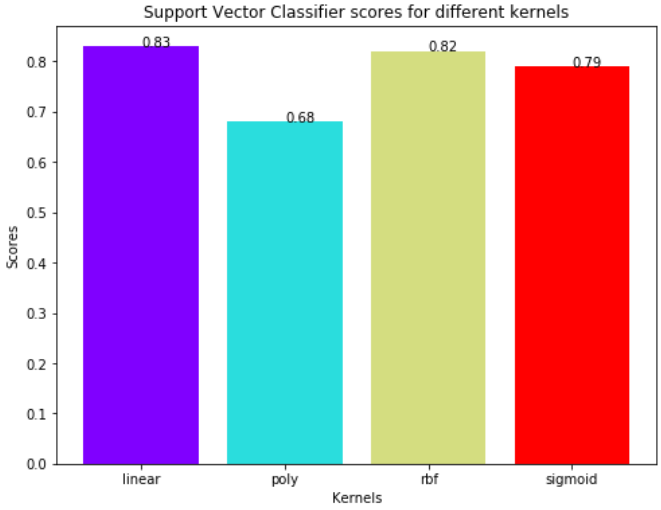
مصنف SVM

يهدف هذا المصنف إلى تكوين مستوي فائق hyperplane يمكنه فصل الفئات قدر الإمكان عن طريق ضبط المسافة بين نقاط البيانات والمستوى الفائق. هناك عدة نوى kernels على أساسها يتم تحديد المستوى الفائق. لقد جربت أربع كيرنلات وهي linear وpoly وrbf وsigmoid.

```
svc_scores = []
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    svc_classifier = SVC(kernel = kernels[i])
    svc_classifier.fit(X_train, y_train)
    svc_scores.append(svc_classifier.score(X_test, y_test))
```

بمجرد حصولي على الدرجات لكل منها، استخدمت طريقة rainbow لتحديد ألوان مختلفة لكل شريط ورسم رسمًا بيانيًا شريطيًا للدرجات التي حققتها كل منها.

```
colors = rainbow(np.linspace(0, 1, len(kernels)))
plt.bar(kernels, svc_scores, color = colors)
for i in range(len(kernels)):
    plt.text(i, svc_scores[i], svc_scores[i])
plt.xlabel('Kernels')
plt.ylabel('Scores')
plt.title('Support Vector Classifier scores for different kernels')
```



كما يتضح من الرسم أعلاه، كان أداء النواة linear هو الأفضل لمجموعة البيانات هذه وحققت درجة 83٪.

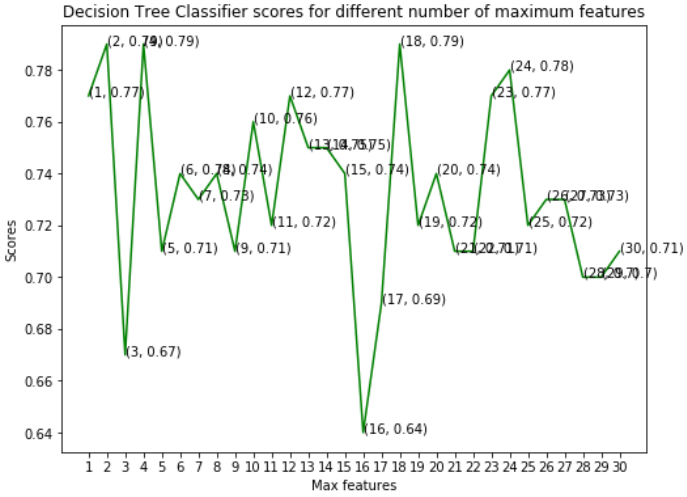
مصنف شجرة القرار

ينشئ هذا المصنف شجرة قرار decision tree بناءً على ذلك، يقوم بتعيين قيم الفئة لكل نقطة بيانات. هنا، يمكننا تغيير الحد الأقصى لعدد الميزات التي يجب مراعاتها أثناء إنشاء النموذج. أقوم بنطاق الميزات من 1 إلى 30 (إجمالي الميزاتفي مجموعة البيانات بعد إضافة أعمدة وهمية dummy columns).

```
dt_scores = []
for i in range(1, len(X.columns) + 1):
    dt_classifier = DecisionTreeClassifier(max_features = i, random_state
= 0)
    dt_classifier.fit(X_train, y_train)
    dt_scores.append(dt_classifier.score(X_test, y_test))
```

بمجرد حصولنا على النتائج، يمكننا بعد ذلك رسم كراف خطي ورؤية تأثير عدد الميزات على درجات النموذج.

```
plt.plot([i for i in range(1, len(X.columns) + 1)], dt scores, color =
'green')
for i in range(1, len(X.columns) + 1):
    plt.text(i, dt_scores[i-1], (i, dt_scores[i-1]))
plt.xticks([i for i in range(1, len(X.columns) + 1)])
plt.xlabel('Max features')
plt.ylabel('Scores')
plt.title('Decision Tree Classifier scores for different number of maximum
features')
```



من الرسم البياني الخطي أعلاه، يمكننا أن نرى بوضوح أن الحد الأقصى للدرجة هو 79٪ وقد تم تحقيقه لأقصى ميزات يتم تحديدها لتكون إما 2 أو 4 أو 18.

مصنف الغابة العشوائي

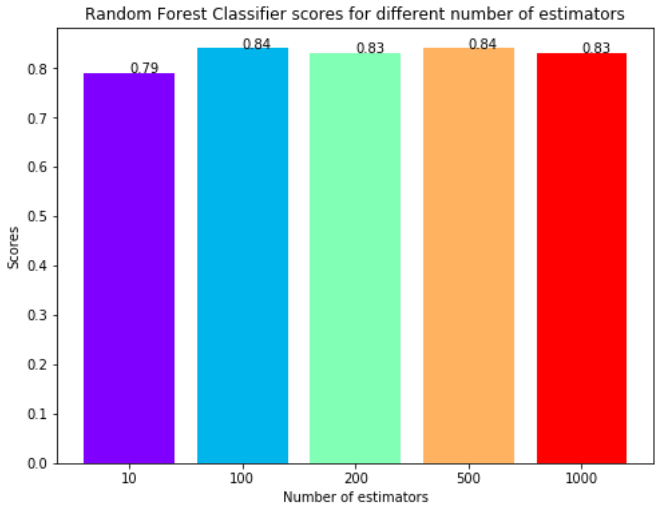
يأخذ هذا المصنف مفهوم أشجار القرار إلى المستوى التالي. يقوم بإنشاء غابة من الأشجار حيث يتم تشكيل كل شجرة عن طريق اختيار عشوائي للمعالم من إجمالي الميزات. هنا، يمكننا تغيير عدد الأشجار التي سيتم استخدامها للتنبؤ بالفئة. أحسب درجات الاختبار على 10 و100 و200 و500 و1000 شجرة.

```
rf_scores = []
estimators = [10, 100, 200, 500, 1000]
for i in estimators:
    rf classifier = RandomForestClassifier(n estimators = i, random state
= 0)
    rf_classifier.fit(X_train, y_train)
    rf_scores.append(rf_classifier.score(X_test, y_test))
```

بعد ذلك، أرسم هذه الدرجات عبر رسم بياني شريطي لمعرفة أيها أعطى أفضل النتائج. قد تلاحظ أنني لم أقم بتعيين قيم X مباشرة كمصفوفة [10, 100, 200, 500, 1000]. سيظهر مخطط مستمر من 10 إلى 1000، والتي سيكون من المستحيل فكها. لذا، لحل هذه المشكلة، استخدمت أولاً قيم X كـ [1, 2, 3, 4, 5]. ثم أعدت تسميتهم باستخدام xticks.

```
colors = rainbow(np.linspace(0, 1, len(estimators)))
plt.bar([i for i in range(len(estimators))], rf_scores, color = colors,
width = 0.8)
for i in range(len(estimators)):
    plt.text(i, rf_scores[i], rf_scores[i])
plt.xticks(ticks = [i for i in range(len(estimators))], labels =
[str(estimator) for estimator in estimators])
plt.xlabel('Number of estimators')
plt.ylabel('Scores')
```

```
plt.title('Random Forest Classifier scores for different number of
estimators')
```



من خلال إلقاء نظرة على الرسم البياني الشريطي، يمكننا أن نرى أن الحد الأقصى للدرجة البالغ 84٪ قد تم تحقيقه لكل من 100 و500 شجرة.

الملخص

تضمن المشروع تحليل مجموعة بيانات مرضى أمراض القلب مع المعالجة المناسبة للبيانات. بعد ذلك، تم تدريب 4 نماذج واختبارها بأقصى درجات على النحو التالي:

- مصنف k_اقرب الجيران (KNN): 87%.
- مصنف دعم المتجهات الناقلة (SVM): 83%.
- مصنف شجرة القرار: 79%.
- مصنف الغابة العشوائية: 84%.

سجل مصنف KNN أفضل نتيجة بنسبة 87 ٪ مع 8 من الجيران.

25) التعرف على الأرقام المكتوبة بخط اليد باستخدام التعلم الآلي
Handwritten Digit Recognition using Machine Learning

يُعد التعرف على النص المكتوب بخط اليد مشكلة يمكن إرجاعها إلى الآلات التلقائية الأولى التي احتاجت إلى التعرف على الأحرف الفرديةفي المستندات المكتوبة بخط اليد. فكر، على سبيل المثال،في الرموز البريدية الموجودة على الرسائلفي مكتب البريد والأتمتة اللازمة للتعرف على هذه الأرقام الخمسة. يُعد التعرف التام على هذه الرموز ضروريًا لفرز البريد تلقائيًا وكفاءة. من بين التطبيقات الأخرى التي قد تتبادر إلى الذهن برنامج OCR (التعرف الضوئي على الحروف). يجب أن يقرأ برنامج التعرف الضوئي على الحروف نصًا مكتوبًا بخط اليد، أو صفحات من الكتب المطبوعة، للوثائق الإلكترونية العامة التي يتم فيها تعريف كل حرف بشكل جيد. لكن مشكلة التعرف على خط اليد تعود إلى زمن بعيد، وبشكل أكثر تحديدًا إلى أوائل القرن العشرين (1920)، عندما بدأ إيمانويل غولدبرغ (1881ـ1970) دراساته بشأن هذه المسألة واقترح أن النهج الإحصائي سيكون الخيار الأمثل.

الفرضية:

توفر مجموعة بيانات Digits الخاصة بمكتبة Scikit-Learn العديد من مجموعات البيانات المفيدة لاختبار العديد من مشكلات تحليل البيانات والتنبؤ بالنتائج. يدعي بعض العلماء أنه يتوقع الرقم بدقة 95٪ من المرات. قم بإجراء تحليل البيانات لقبول هذه الفرضية أو رفضها.

المتطلبات المسبقة:

- Sklearn.
- Matplotlib.
- أساسيات التعلم الآلي.

مجموعة البيانات:

في هذا المشروع، نستخدم مجموعة بيانات Handwritten Digits وهي جاهزة بالفعلفي مكتبة sklearn. يمكننا استيراد مجموعة البيانات باستخدام الكود أدناه.

```
from sklearn import datasets
digits = datasets.load_digits()
```

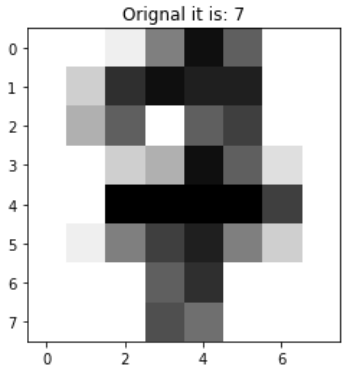
مجموعة البيانات الرقمية عبارة عن قاموس يحتوي على بيانات وأهداف وصور وأسماء ميزات ووصف لمجموعة البيانات وأسماء الهدف وما إلى ذلك.

نحن نركز بشكل رئيسي على البيانات والأهداف. نستخرج كلاهما على متغيرات مختلفة.

```
main_data = digits['data']
targets = digits['target']
```

الآن يمكننا رؤية بياناتنا باستخدام الكود التالي.

```
def view_digit(index):
    plt.imshow(digits.images[index] , cmap = plt.cm.gray_r ,
interpolation = 'nearest')
    plt.title('Orignal it is: '+ str(digits.target[index]))
    plt.show()view_digit(17)
```

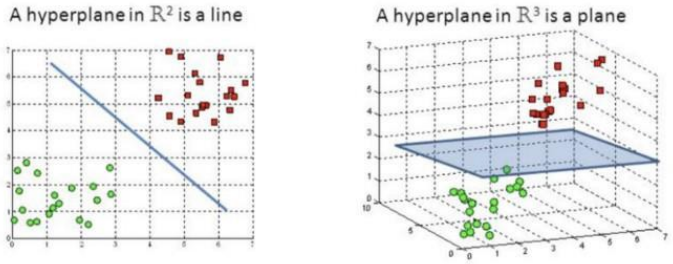


تخطيط النموذج:

لمعرفة كيفية عمل النماذج المختلفة على أحجام بيانات مختلفة، نستخدم 3 نماذج: مصنف آلة المتجهات الداعمة SVM، مصنف شجرة القرار، مصنف الغابات العشوائية.

1) مصنف الة المتجهات الداعمة:

الهدف من خوارزمية آلة المتجهات الداعمة SVM هو العثور على مستوى فائق hyperplane في فضاء N-dimensional (N ـ عدد الميزات) التي تصنف نقاط البيانات بوضوح.



Hyperplanes in 2D and 3D feature space

الكود:

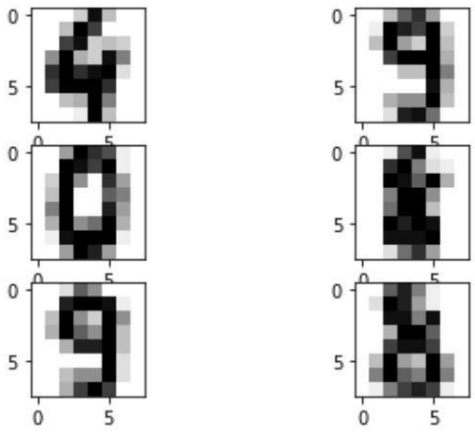
```
# import the SVC
from sklearn import svm
svc = svm.SVC(gamma=0.001 , C = 100.)
# gamma and C are hyperparameters# Training data = 1790 , Validation
data = 6
svc.fit(main_data[:1790] , targets[:1790])# predict on test data
predictions = svc.predict(main_data[1791:])# check the result
predictions , targets[1791:]
```

```
predictions , targets[1791:]
```

```
(array([4, 9, 0, 8, 9, 8]), array([4, 9, 0, 8, 9, 8]))
```

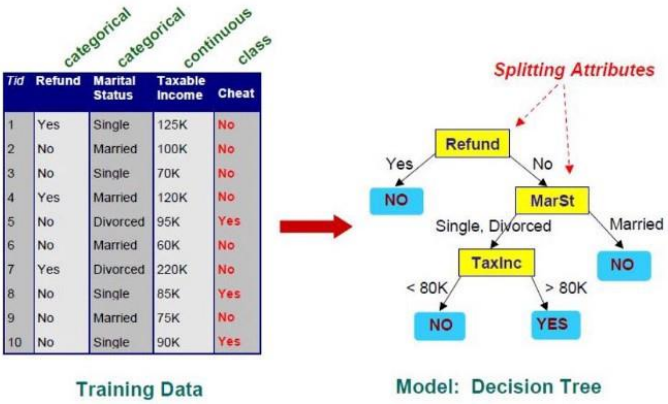
كما نرى، نستخدم بيانات عالية جدًا للتدريب وبيانات قليلة جدًا للتدريب، ويقوم مصنف SVM بعمل جيد جدًا على البيانات ونحصل على دقة بنسبة 100 ٪في بيانات الاختبار.

```
<matplotlib.image.AxesImage at 0x1fe8ee88fa0>
```



2) مصنف شجرة القرار

مصنف شجرة القرار DTC هو أسلوب تصنيف بسيط وشائع الاستخدام. يطبق فكرة مباشرة لحل مشكلة التصنيف. يطرح مصنف شجرة القرار سلسلة من الأسئلة المصممة بعناية حول سمات سجل الاختبار.في كل مرة يتلقى فيها إجابة، يتم طرح سؤال متابعة حتى يتم الوصول إلى استنتاج حول تسمية الفئة الخاصة بالسجل.



الكود:

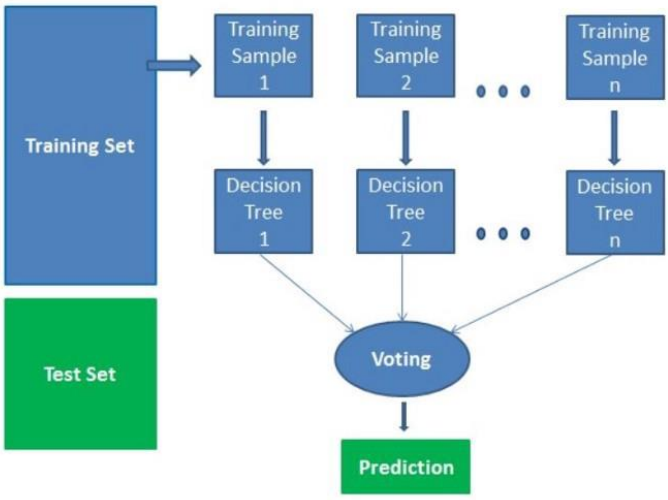
```
# import the Classifier
from sklearn.tree import DecisionTreeClassifier# Instanciate Model
# we can also use criterion = 'entropy' both lead us to nearly same
# resultdt = DecisionTreeClassifier(criterion = 'gini') # fit the data
on model
# Training Set = 1600 , Validation Set = 197
dt.fit(main_data[:1600] , targets[:1600])# prediction on test data
predictions2 = dt.predict(main_data[1601:])# We use classification
materics as accuracy_score
# import accuracy_score
from sklearn.metrics import
accuracy_scoreaccuracy_score(targets[1601:] , predictions2)
```

```
accuracy_score(targets[1601:] , predictions2)
0.7857142857142857
```

الآنفي هذا الوقت، نستخدم أحجامًا مختلفة من بيانات التدريب وبيانات التحقق من الصحة. كما نرى، يؤدي مصنف شجرة القرار أداءً ضعيفًا على البيانات. يمكننا زيادة الدقة عن طريق ضبط المعلمات الفائقة لـ DTC.

3) مصنف الغابة العشوائية

الغابات العشوائية هي خوارزمية تعلم خاضعة للإشراف. يمكن استخدامه لكل من التصنيف والانحدار. وهي أيضًا الخوارزمية الأكثر مرونة وسهولةفي الاستخدام. تتكون الغابة من الأشجار. يقال إنه كلما زاد عدد الأشجار، زادت قوة الغابة. تنشئ الغابات العشوائية أشجار قرار على عينات بيانات مختارة عشوائيًا، وتحصل على تنبؤ من كل شجرة، وتختار أفضل حل عن طريق التصويت. كما أنه يوفر مؤشرًا جيدًا لأهمية الميزة.



الكود:

```
from sklearn.ensemble import RandomForestClassifier# n_estimators
hyperparameters( default 100 )
rc = RandomForestClassifier(n estimators = 150)# Training Data = 1500
, Validation data = 297
rc.fit(main_data[:1500] , targets[:1500])predictions3 =
rc.predict(main_data[1501:])accuracy_score(targets[1501:] ,
predictions3)
```

```
accuracy_score(targets[1501:] , predictions3)
```

```
0.9222972972972973
```

كما نرى، تعمل مصنف الغالة العشوائية شكل ممتاز مع بيانات أقل مقارنة بكل من شجرة القرار وآلة المتجهات الداعمة. حصلنا على درجة دقة 92٪ باستخدام مصنف الغابة العشوائية.

الملخص:

وفقًا لفرضيتنا، يمكننا القول من خلال ضبط المعلمة الفائقة باستخدام نماذج مختلفة للتعلم الآلي أو باستخدام المزيد من البيانات، يمكننا تحقيق دقة تقارب 95٪ على مجموعة البيانات المكتوبة بخط اليد. ولكن تأكد من أن لدينا أيضًا قدرًا جيدًا من بيانات الاختبار وإلا فسيحصل النموذج على أكثر من اللازم.

26) توقع أسعار الكهرباء باستخدام التعلم الآلي Electricity Price Prediction using Machine Learning

يعتمد سعر الكهرباء على عدة عوامل. يساعد توقع سعر الكهرباء العديد من الشركات على فهم مقدار الكهرباء التي يتعين عليهم دفعها كل عام. تعتمد مهمة توقع أسعار الكهرباء على دراسة حالة تحتاج فيها إلى توقع السعر اليومي للكهرباء بناءً على الاستهلاك اليومي للآلات الثقيلة التي تستخدمها الشركات. لذلك إذا كنت تريد معرفة كيفية التنبؤ بسعر الكهرباء، فهذه المقالة مناسبة لك.في هذه المقالة، سوف أطلعك على مهمة التنبؤ بأسعار الكهرباء مع التعلم الآلي باستخدام بايثون.

توقع أسعار الكهرباء (دراسة حالة)

افترض أن عملك يعتمد على خدمات الحوسبة حيث تختلف الطاقة التي تستهلكها أجهزتك على مدار اليوم. لا تعرف التكلفة الفعلية للكهرباء التي تستهلكها الآلات على مدار اليوم، لكن المنظمة زودتك ببيانات تاريخية عن سعر الكهرباء التي تستهلكها الآلات. فيما يلي معلومات البيانات المتوفرة لدينا لمهمة التنبؤ بأسعار الكهرباء:

1. DateTime: تاريخ ووقت السجل.
2. Holiday: يحتوي على اسم العطلة إذا كان اليوم عطلة وطنية.
3. HolidayFlag: يحتوي على 1 إذا كانت عطلة البنوك وإلا 0.
4. DayOfWeek: يحتوي على قيم بين 0ـ6 حيث يكون 0 هو يوم الإثنين.
5. WeekOfYear: أسبوع من السنة.
6. Day: يوم التاريخ.
7. Month: شهر التاريخ.
8. Year: سنة التاريخ.
9. PeriodOfDay: فترة نصف ساعة من اليوم.
10. ForcastWindProduction: توقعات إنتاج الرياح.
11. SystemLoadEA توقع الحمل الوطني المتوقع.
12. SMPEA: السعر المتوقع.
13. ORKTemperature: قياس درجة الحرارة الفعلية.
14. ORKWindspeed: قياس سرعة الرياح الفعلية.
15. CO2Intensity: الكثافة الفعلية لثاني أكسيد الكربون للكهرباء المنتجة.
16. ActualWindProduction: إنتاج طاقة الرياح الفعلي.
17. SystemLoadEP2: الحمل الفعلي للنظام الوطني.
18. SMPEP2: السعر الفعلي للكهرباء المستهلكة (علامات أو قيم يمكن توقعها).

لذا فإن مهمتك هنا هي استخدام هذه البيانات لتدريب نموذج التعلم الآلي للتنبؤ بسعر الكهرباء التي تستهلكها الآلات.في القسم أدناه، سأطلعك على مهمة التنبؤ بأسعار الكهرباء باستخدام التعلم الآلي باستخدام Python.

توقع أسعار الكهرباء باستخدام لغة بايثون

سأبدأ مهمة التنبؤ بأسعار الكهرباء عن طريق استيراد مكتبات Python الضرورية ومجموعة البيانات التي نحتاجها لهذه المهمة:

```
import pandas as pd
import numpy as np
data =
pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-
data/master/electricity.csv")
print(data.head())
```

```
          DateTime Holiday  ...  SystemLoadEP2  SMPEP2
0  01/11/2011 00:00    None  ...        3159.60   54.32
1  01/11/2011 00:30    None  ...        2973.01   54.23
2  01/11/2011 01:00    None  ...        2834.00   54.23
3  01/11/2011 01:30    None  ...        2725.99   53.47
4  01/11/2011 02:00    None  ...        2655.64   39.87

[5 rows x 18 columns]
```

دعونا نلقي نظرة على جميع أعمدة مجموعة البيانات هذه:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38014 entries, 0 to 38013
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   DateTime               38014 non-null  object
 1   Holiday                38014 non-null  object
 2   HolidayFlag            38014 non-null  int64
 3   DayOfWeek              38014 non-null  int64
 4   WeekOfYear             38014 non-null  int64
 5   Day                    38014 non-null  int64
 6   Month                  38014 non-null  int64
 7   Year                   38014 non-null  int64
 8   PeriodOfDay            38014 non-null  int64
 9   ForecastWindProduction 38014 non-null  object
 10  SystemLoadEA           38014 non-null  object
 11  SMPEA                  38014 non-null  object
 12  ORKTemperature         38014 non-null  object
 13  ORKWindspeed           38014 non-null  object
 14  CO2Intensity           38014 non-null  object
 15  ActualWindProduction   38014 non-null  object
 16  SystemLoadEP2          38014 non-null  object
 17  SMPEP2                 38014 non-null  object
dtypes: int64(7), object(11)
memory usage: 5.2+ MB
```

أستطيع أن أرى أن العديد من الميزات ذات القيم العددية هي قيم سلسلةفي مجموعة البيانات وليست أعدادًا صحيحة أو قيمًا عائمة. لذا قبل المضي قدمًا، يتعين علينا تحويل قيم السلسلة النصية هذه إلى قيم عائمة float values:

```
data["ForecastWindProduction"] =
pd.to_numeric(data["ForecastWindProduction"], errors= 'coerce')
data["SystemLoadEA"] = pd.to_numeric(data["SystemLoadEA"], errors=
'coerce')
data["SMPEA"] = pd.to_numeric(data["SMPEA"], errors= 'coerce')
data["ORKTemperature"] = pd.to_numeric(data["ORKTemperature"], errors=
'coerce')
data["ORKWindspeed"] = pd.to_numeric(data["ORKWindspeed"], errors=
'coerce')
data["CO2Intensity"] = pd.to_numeric(data["CO2Intensity"], errors=
'coerce')
data["ActualWindProduction"] =
pd.to_numeric(data["ActualWindProduction"], errors= 'coerce')
data["SystemLoadEP2"] = pd.to_numeric(data["SystemLoadEP2"], errors=
'coerce')
data["SMPEP2"] = pd.to_numeric(data["SMPEP2"], errors= 'coerce')
```

دعنا الآن نلقي نظرة على ما إذا كانت مجموعة البيانات هذه تحتوي على أي قيم فارغة أم لا:

```
data.isnull().sum()
```

```
DateTime                  0
Holiday                   0
HolidayFlag               0
DayOfWeek                 0
WeekOfYear                0
Day                       0
Month                     0
Year                      0
PeriodOfDay               0
ForecastWindProduction    5
SystemLoadEA              2
SMPEA                     2
ORKTemperature          295
ORKWindspeed            299
CO2Intensity              7
ActualWindProduction      5
SystemLoadEP2             2
SMPEP2                    2
dtype: int64
```

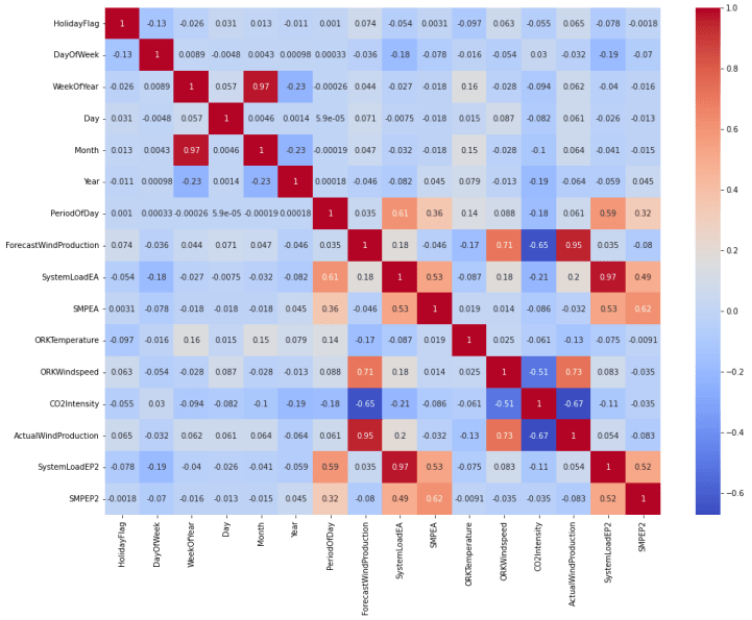
لذلك هناك بعض الأعمدة التي تحتوي على قيم خالية، سأقوم بإسقاط كل هذه الصفوف التي تحتوي على قيم خالية من مجموعة البيانات:

```
data = data.dropna()
```

دعنا الآن نلقي نظرة على الارتباط بين جميع الأعمدةفي مجموعة البيانات:

```
import seaborn as sns
import matplotlib.pyplot as plt
correlations = data.corr(method='pearson')
plt.figure(figsize=(16, 12))
```

```
sns.heatmap(correlations, cmap="coolwarm", annot=True)
plt.show()
```



نموذج توقع أسعار الكهرباء

دعنا الآن ننتقل إلى مهمة تدريب نموذج توقع أسعار الكهرباء. سأضيف هنا أولاً جميع الميزات المهمة إلى x والعمود المستهدف إلى y، وبعد ذلك سأقسم البيانات إلى مجموعات تدريب واختبار:

```
x = data[["Day", "Month", "ForecastWindProduction", "SystemLoadEA",
          "SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity",
          "ActualWindProduction", "SystemLoadEP2"]]
y = data["SMPEP2"]
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                                random_state=42)
```

نظرًا لأن هذه هي مشكلة الانحدار، سأختار هنا خوارزمية الغابة العشوائية Random Forest لتدريب نموذج التنبؤ بسعر الكهرباء:

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
model.fit(xtrain, ytrain)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

دعنا الآن ندخل جميع قيم الميزات الضرورية التي استخدمناها لتدريب النموذج وإلقاء نظرة على سعر الكهرباء الذي تنبأ به النموذج:

```
#features = [["Day", "Month", "ForecastWindProduction",
"SystemLoadEA", "SMPEA", "ORKTemperature", "ORKWindspeed",
"CO2Intensity", "ActualWindProduction", "SystemLoadEP2"]]
features = np.array ,14.8 ,9.0 ,49.56 ,4241.05 ,54.10 ,12 ,10]])
([[4426.84 ,54.0 ,491.32
model.predict(features)
```

```
array([65.1696])
```

هذه هي الطريقة التي يمكنك بها تدريب نموذج التعلم الآلي للتنبؤ بأسعار الكهرباء.

الملخص

يساعد توقع سعر الكهرباء الكثير من الشركات على فهم مقدار نفقات الكهرباء التي يتعين عليهم دفعها كل عام. آمل أن تكون قد أحببت هذه المقالة حول مهمة التنبؤ بأسعار الكهرباء مع التعلم الآلي باستخدام بايثون.

.

27) تحليل جودة المياه باستخدام التعلم الآلي Water Quality Analysis using Machine Learning

يُعد الحصول على مياه الشرب المأمونة أحد الاحتياجات الأساسية لجميع البشر. من وجهة نظر قانونية، يعتبر الحصول على مياه الشرب أحد حقوق الإنسان الأساسية. تؤثر العديد من العوامل على جودة المياه، كما أنها أحد مجالات البحث الرئيسيةفي التعلم الآلي. لذلك إذا كنت تريد معرفة كيفية إجراء تحليل جودة المياه باستخدام التعلم الآلي، فهذه المقالة مناسبة لك.في هذه المقالة، سوف أطلعك على تحليل جودة المياه باستخدام التعلم الآلي باستخدام Python.

تحليل جودة المياه

يُعد تحليل جودة المياه أحد المجالات الرئيسية للبحثفي التعلم الآلي. يُعرف أيضًا باسم تحليل قابلية المياه للشرب لأن مهمتنا هنا هي فهم جميع العوامل التي تؤثر على قابلية المياه للشرب وتدريب نموذج التعلم الآلي الذي يمكنه تصنيف ما إذا كانت عينة مياه معينة آمنة أو غير صالحة للاستهلاك.

بالنسبة لمهمة تحليل جودة المياه، سأستخدم مجموعة بيانات Kaggle التي تحتوي على بيانات حول جميع العوامل الرئيسية التي تؤثر على قابلية المياه للشرب. جميع العوامل التي تؤثر على جودة المياه مهمة للغاية، لذلك نحتاج إلى استكشاف كل ميزة من ميزات مجموعة البيانات هذه بإيجاز قبل تدريب نموذج التعلم الآلي للتنبؤ بما إذا كانت عينة المياه آمنة أو غير مناسبة للاستهلاك. يمكنك تنزيل مجموعة البيانات التي أستخدمها لمهمة تحليل جودة المياه من هنا.

تحليل جودة المياه باستخدام لغة بايثون

سأبدأ مهمة تحليل جودة المياه عن طريق استيراد مكتبات Python ومجموعة البيانات الضرورية:

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np

data = pd.read_csv("water_potability.csv")
data.head()
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

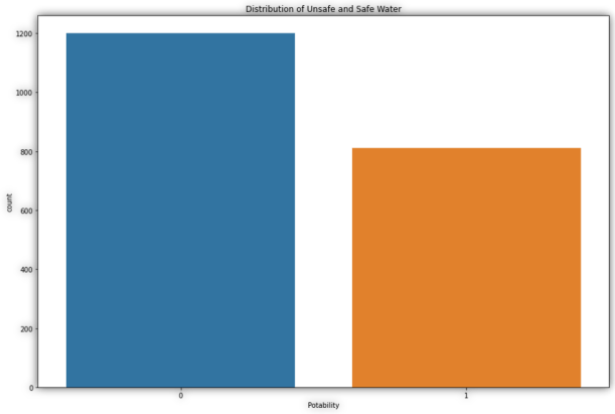
يمكنني رؤية القيم الخاليةفي المعاينة الأولى لمجموعة البيانات هذه نفسها، لذلك قبل المضي قدمًا، دعنا نزيل جميع الصفوف التي تحتوي على قيم فارغة:

```
data = data.dropna()
data.isnull().sum()
```

```
ph               0
Hardness         0
Solids           0
Chloramines      0
Sulfate          0
Conductivity     0
Organic_carbon   0
Trihalomethanes  0
Turbidity        0
Potability       0
dtype: int64
```

عمود القابلية للشرب Potability column لمجموعة البيانات هذه هو العمود الذي نحتاج إلى توقعه لأنه يحتوي على القيمتين 0 و 1 التي تشير إلى ما إذا كانت المياه صالحة للشرب (1) أو غير صالحة (0) للشرب. لذلك دعونا نرى توزيع 0 و 1في عمود "Potability ":

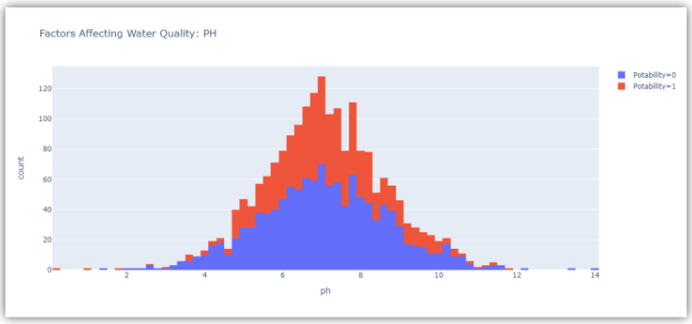
```
plt.figure(figsize=(15, 10))
sns.countplot(data.Potability)
plt.title("Distribution of Unsafe and Safe Water")
plt.show()
```



لذلك هذا شيء يجب أن تلاحظه أن مجموعة البيانات هذه غير متوازنة لأن عينات الأصفار أكثر من 1 ثانية.

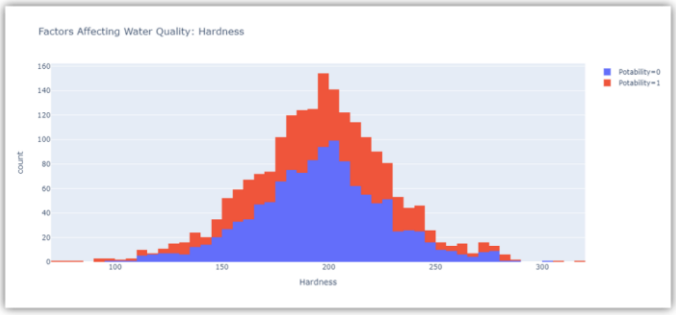
كما ذكرنا أعلاه، لا توجد عوامل لا يمكننا تجاهلها والتي تؤثر على جودة المياه، لذلك دعونا نستكشف جميع الأعمدة واحدة تلو الأخرى. لنبدأ بإلقاء نظرة على عمود الاس الهيدروجيني ph column:

```
import plotly.express as px
data = data
figure = px.histogram(data, x = "ph",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: PH")
figure.show()
```



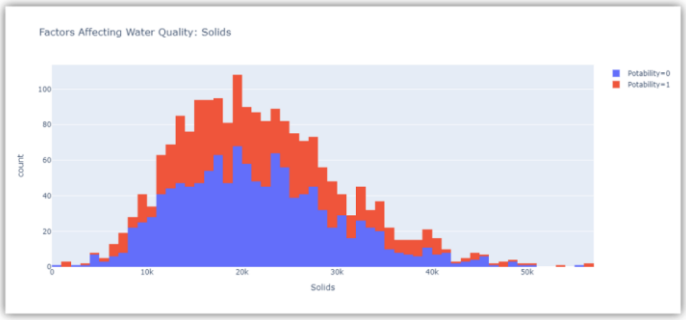
يمثل عمود الأس الهيدروجيني قيمة الأس الهيدروجيني للماء وهو عامل مهمفي تقييم التوازن الحمضي القاعدي للماء. يجب أن تكون قيمة الرقم الهيدروجيني لمياه الشرب بين 6.5 و8.5. دعونا الآن نلقي نظرة على العامل الثاني الذي يؤثر على جودة المياهفي مجموعة البيانات:

```
figure = px.histogram(data, x = "Hardness",
                      color = "Potability",
                      title= "Factors Affecting Water Quality:
Hardness")
figure.show()
```



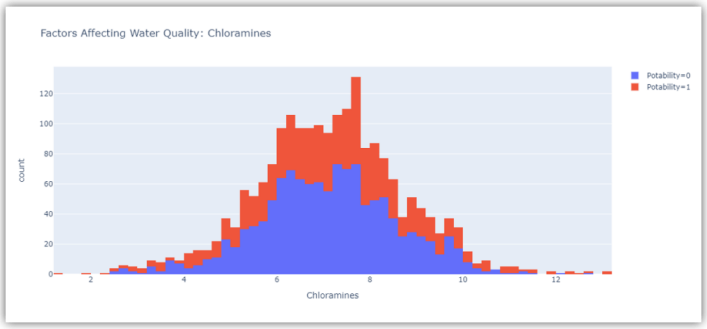
يوضح الشكل أعلاه توزيع عسرة hardness الماءفي مجموعة البيانات. تعتمد عسرة الماء عادة على مصدره، لكن الماء الذي تصل قوته إلى 120 ـ 200 ملليغرام صالح للشرب. دعنا الآن نلقي نظرة على العامل التالي الذي يؤثر على جودة المياه:

```
figure = px.histogram(data, x = "Solids",
                      color = "Potability",
                      title= "Factors Affecting Water Quality:
Solids")
figure.show()
```



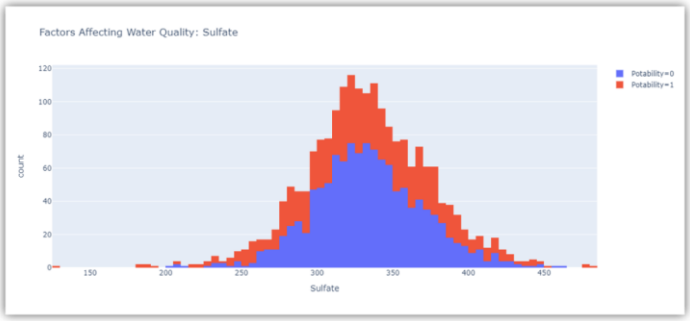
يمثل الشكل أعلاه توزيع إجمالي المواد الصلبة الذائبةفي الماءفي مجموعة البيانات. تسمى جميع المعادن العضوية وغير العضوية الموجودةفي الماء بالمواد الصلبة الذائبة. الماء الذي يحتوي على عدد كبير جداً من المواد الصلبة الذائبة شديد التمعدن. دعنا الآن نلقي نظرة على العامل التالي الذي يؤثر على جودة المياه:

```
figure = px.histogram(data, x = "Chloramines",
                      color = "Potability",
                      title= "Factors Affecting Water Quality:
Chloramines")
figure.show()
```



يمثل الشكل أعلاه توزيع الكلورامينفي الماءفي مجموعة البيانات. الكلورامين والكلور من المطهرات المستخدمةفي أنظمة المياه العامة. دعنا الآن نلقي نظرة على العامل التالي الذي يؤثر على جودة المياه:

```
figure = px.histogram(data, x = "Sulfate",
                      color = "Potability",
                      title= "Factors Affecting Water Quality:
Sulfate")
figure.show()
```

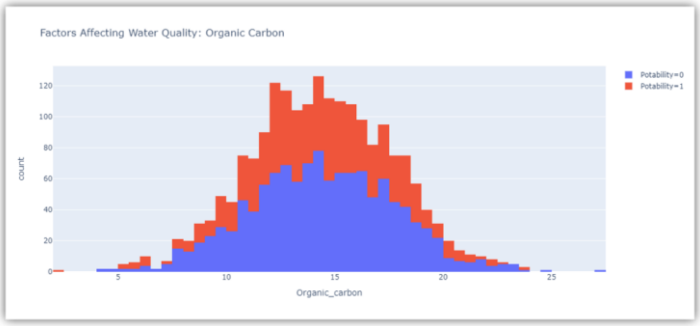


يوضح الشكل أعلاه توزيع الكبريتاتفي الماءفي مجموعة البيانات. إنها مواد موجودة بشكل طبيعي في المعادن والتربة والصخور. الماء الذي يحتوي على أقل من 500 ملليغرام من الكبريتات آمن للشرب. الآن دعونا نرى العامل التالي:

```
figure = px.histogram(data, x = "Conductivity",
                      color = "Potability",
                      title= "Factors Affecting Water Quality:
Conductivity")
figure.show()
```

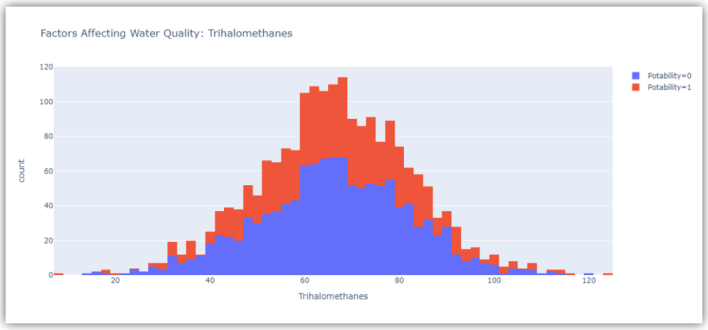
يمثل الشكل أعلاه توزيع موصلية المياهفي مجموعة البيانات. الماء موصل جيد للكهرباء، لكن أنقى أشكال الماء ليس موصلًا جيدًا للكهرباء. المياه ذات التوصيل الكهربائي أقل من 500 صالحة للشرب. الآن دعونا نرى العامل التالي:

```
figure = px.histogram(data, x = "Organic_carbon",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: Organic
Carbon")
figure.show()
```



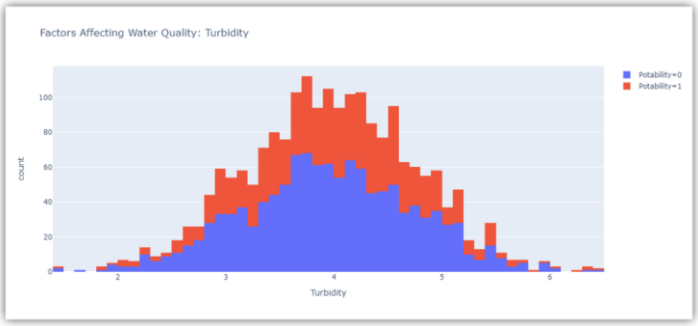
يمثل الشكل أعلاه توزيع الكربون العضويفي الماءفي مجموعة البيانات. يأتي الكربون العضوي من انهيار المواد العضوية الطبيعية والمصادر الاصطناعية. تعتبر المياه التي تحتوي على أقل من 25 ملليجرام من الكربون العضوي آمنة للشرب. دعنا الآن نلقي نظرة على العامل التالي الذي يؤثر على جودة مياه الشرب:

```
figure = px.histogram(data, x = "Trihalomethanes",
                      color = "Potability",
                      title= "Factors Affecting Water Quality:
Trihalomethanes")
figure.show()
```



يمثل الشكل أعلاه توزيع ثلاثي الميثان أو THMفي الماءفي مجموعة البيانات. THMs هي مواد كيميائية موجودةفي المياه المعالجة بالكلور. تعتبر المياه التي تحتوي على أقل من 80 ملليجرام من THMs آمنة للشرب. دعنا الآن نلقي نظرة على العامل التاليفي مجموعة البيانات الذي يؤثر على جودة مياه الشرب:

```
figure = px.histogram(data, x = "Turbidity",
                      color = "Potability",
                      title= "Factors Affecting Water Quality:
Turbidity")
figure.show()
```



يمثل الشكل أعلاه توزيع العكارة turbidityفي الماء. تعتمد عكارة الماء على عدد المواد الصلبة الموجودةفي المعلق. تعتبر المياه ذات العكارة أقل من 5 ملليغرام صالحة للشرب.

نموذج التنبؤ بجودة المياه باستخدام لغة بايثون

في القسم أعلاه، استكشفنا جميع الميزات التي تؤثر على جودة المياه. الآن، الخطوة التالية هي تدريب نموذج التعلم الآلي لمهمة تحليل جودة المياه باستخدام Python. لهذه المهمة، سأستخدم مكتبة PyCaretفي Python. إذا لم تكن قد استخدمت هذه المكتبة من قبل، فيمكنك تثبيتها بسهولة على نظامك باستخدام الأمر pip:

- pip install pycaret

قبل تدريب نموذج التعلم الآلي، دعنا نلقي نظرة على الارتباط بين جميع الميزات فيما يتعلق بعمود إمكانية الاستخدامفي مجموعة البيانات:

```
correlation = data.corr()
correlation["ph"].sort_values(ascending=False)
```

```
ph                 1.000000
Hardness           0.108948
Organic_carbon     0.028375
Trihalomethanes    0.018278
Potability         0.014530
Conductivity       0.014128
Sulfate            0.010524
Chloramines       -0.024768
Turbidity         -0.035849
Solids            -0.087615
Name: ph, dtype: float64
```

الآن فيما يلي كيف يمكنك معرفة أي خوارزمية تعلم الآلة هي الأفضل لمجموعة البيانات هذه باستخدام مكتبة PyCaretفي Python:

```
from pycaret.classification import*
clf = setup(data, target = "Potability", silent = True, session_id =
786)
compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.6830	0.7005	0.4197	0.6744	0.5133	0.2976	0.3182	0.724
qda	Quadratic Discriminant Analysis	0.6823	0.7192	0.3985	0.6883	0.5013	0.2917	0.3174	0.022
et	Extra Trees Classifier	0.6816	0.6941	0.3861	0.6858	0.4916	0.2863	0.3123	0.557
lightgbm	Light Gradient Boosting Machine	0.6652	0.6916	0.4762	0.6078	0.5324	0.2781	0.2840	0.172
gbc	Gradient Boosting Classifier	0.6602	0.6738	0.3718	0.6306	0.4667	0.2419	0.2603	0.339
nb	Naive Bayes	0.6184	0.6078	0.2478	0.5545	0.3412	0.1261	0.1462	0.019
dt	Decision Tree Classifier	0.6034	0.5895	0.5186	0.5049	0.5097	0.1775	0.1784	0.027
lr	Logistic Regression	0.5984	0.5199	0.0071	0.1900	0.0134	0.0028	0.0127	0.355
ridge	Ridge Classifier	0.5984	0.0000	0.0089	0.1583	0.0168	0.0035	0.0056	0.021
lda	Linear Discriminant Analysis	0.5977	0.4903	0.0089	0.1500	0.0167	0.0021	0.0024	0.022
ada	Ada Boost Classifier	0.5956	0.5671	0.2919	0.4896	0.3644	0.0972	0.1034	0.173
knn	K Neighbors Classifier	0.5743	0.5423	0.3644	0.4642	0.4070	0.0826	0.0846	0.121
svm	SVM - Linear Kernel	0.5194	0.0000	0.3982	0.1604	0.2287	-0.0014	-0.0104	0.027

وفقًا للنتيجة أعلاه، فإن خوارزمية الغابة العشوائية random forrest(rf)هي الأفضل لتدريب نموذج التعلم الآلي لمهمة تحليل جودة المياه. لذلك دعونا ندرب النموذج ونفحص تنبؤاته:

```
model = create_model("rf")
predict = predict_model(model, data=data)
predict.head()
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability	Label	Score
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0	0	0.87
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0	0	0.91
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	8.399735	54.917862	2.559708	0	0	0.83
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	13.789695	84.603556	2.672989	0	0	0.89
7	8.635849	203.361523	13672.091764	4.563009	303.309771	474.607645	12.363817	62.798309	4.401425	0	0	0.94

النتائج المذكورة أعلاه تبدو مرضية. آمل أن تكون قد أحببت مشروع التعلم الآلي هذا حول تحليل جودة المياه باستخدام Python.

الملخص

هذه هي الطريقة التي يمكنك بها تحليل جودة المياه وتدريب نموذج التعلم الآلي لتصنيف المياه الآمنة وغير الآمنة للشرب. يُعد الحصول على مياه الشرب المأمونة أحد الاحتياجات الأساسية لجميع البشر. من وجهة نظر قانونية، يعتبر الحصول على مياه الشرب أحد حقوق الإنسان الأساسية. تؤثر العديد من العوامل على جودة المياه، كما أنها أحد مجالات البحث الرئيسيةفي التعلم الآلي.

28) توقع التأمين باستخدام التعلم الآلي Insurance Prediction using Machine Learning

التأمين Insurance هو عقد يحصل بموجبه الفرد على الحماية المالية ضد الخسائر من شركة التأمين ضد مخاطر الخسائر المالية كما هو مذكورفي التأمين. توجد أنواع عديدة من التأمين اليوم وهناك العديد من الشركات التي تقدم خدمات التأمين. تحتاج هذه الشركات دائمًا إلى التنبؤ بما إذا كان الشخص سيشتري التأمين أم لا حتى يتمكن من توفير الوقت والمال للعملاء الأكثر ربحية. لذلك إذا كنت تريد معرفة كيف يمكننا استخدام التعلم الآلي للتنبؤ بما إذا كان الفرد سيشتري التأمين أم لا، فهذه المقالة مناسبة لك.في هذه المقالة، سوف أطلعك على مهمة توقع التأمين مع التعلم الآلي باستخدام Python.

توقع التأمين مع التعلم الآلي

مهمة التنبؤ بالتأمين Insurance Prediction هي شيء يضيف قيمة إلى كل شركة تأمين. يستخدمون بيانات من قاعدة بياناتهم حول كل شخص اتصلوا بهم للترويج لخدمات التأمين ومحاولة العثور على الأشخاص الأكثر احتمالية الذين يمكنهم شراء التأمين. يساعد هذا الشركة على استهداف العملاء الأكثر ربحية ويوفر الوقت والمال لشركة التأمين.

في القسم أدناه، سوف آخذك خلال مهمة توقع التأمين باستخدام التعلم الآلي باستخدام Python. بالنسبة لمهمة التنبؤ بالتأمين مع التعلم الآلي، قمت بجمع مجموعة بيانات من Kaggle حول العملاء السابقين لشركة تأمين السفر. مهمتنا هنا هي تدريب نموذج التعلم الآلي للتنبؤ بما إذا كان الفرد سيشتري بوليصة التأمين من الشركة أم لا.

توقع التأمين باستخدام بايثون

لنبدأ مهمة التنبؤ بالتأمين باستخدام التعلم الآلي عن طريق استيراد مكتبات Python ومجموعة البيانات الضرورية:

مجموعة البيانات

```
import pandas as pd
data = pd.read_csv("TravelInsurancePrediction.csv")
data.head()
```

	Unnamed: 0	Age	...	EverTravelledAbroad	TravelInsurance
0	0	31	...	No	0
1	1	31	...	No	0
2	2	34	...	No	1
3	3	28	...	No	0
4	4	28	...	No	0

[5 rows x 10 columns]

لا فائدة من العمود غير المسمى Unnamedفي مجموعة البيانات هذه، لذا سأزيله من البيانات:

```
data.drop(columns=["Unnamed: 0"], inplace=True)
```

دعنا الآن نلقي نظرة على بعض الأفكار الضرورية للحصول على فكرة حول نوع البيانات التي نعمل معها:

```
data.isnull().sum()
```

```
Age                    0
Employment Type        0
GraduateOrNot          0
AnnualIncome           0
FamilyMembers          0
ChronicDiseases        0
FrequentFlyer          0
EverTravelledAbroad    0
TravelInsurance        0
dtype: int64
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1987 entries, 0 to 1986
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Age                  1987 non-null   int64
 1   Employment Type      1987 non-null   object
 2   GraduateOrNot        1987 non-null   object
 3   AnnualIncome         1987 non-null   int64
 4   FamilyMembers        1987 non-null   int64
 5   ChronicDiseases      1987 non-null   int64
 6   FrequentFlyer        1987 non-null   object
 7   EverTravelledAbroad  1987 non-null   object
 8   TravelInsurance      1987 non-null   int64
dtypes: int64(5), object(4)
memory usage: 139.8+ KB
```

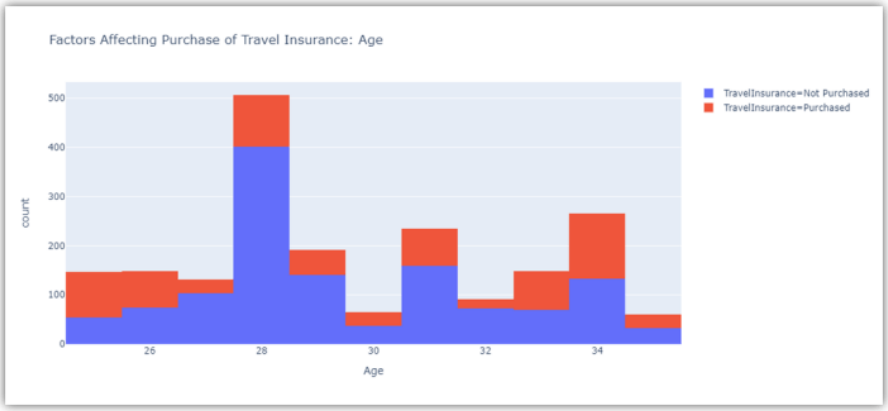
في مجموعة البيانات هذه، التسميات التي نريد توقعها موجودةفي عمود "TravelInsurance". تم ذكر القيم الموجودةفي هذا العمود على أنها 0 و 1 حيث يعني 0 عدم الشراء ويعني 1 أنه تم الشراء. لفهم أفضل عند تحليل هذه البيانات، سأحول 1 و 0 إلى مشتراة purchased وغير مشتراة not purchased:

```
data["TravelInsurance"] = data["TravelInsurance"].map({0: "Not
Purchased", 1: "Purchased"})
```

لنبدأ الآن بإلقاء نظرة على عمود العمر age column لمعرفة مدى تأثير العمر على شراء بوليصة التأمين:

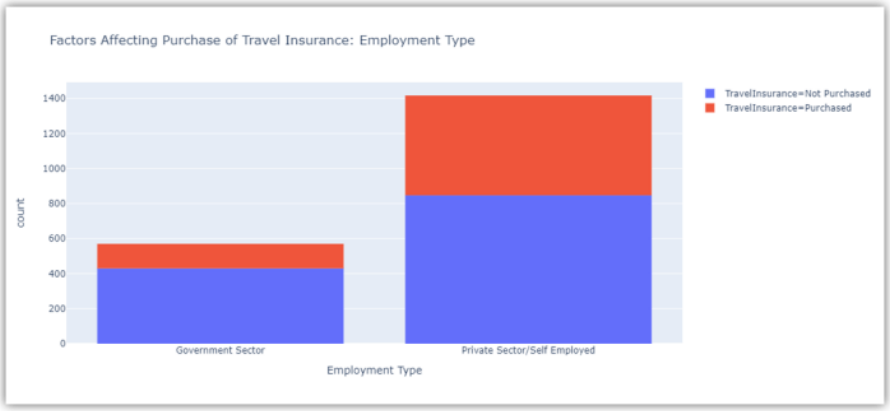
```
import plotly.express as px
data = data
```

```
figure = px.histogram(data, x = "Age",
                      color = "TravelInsurance",
                      title= "Factors Affecting Purchase of Travel
Insurance: Age")
figure.show()
```



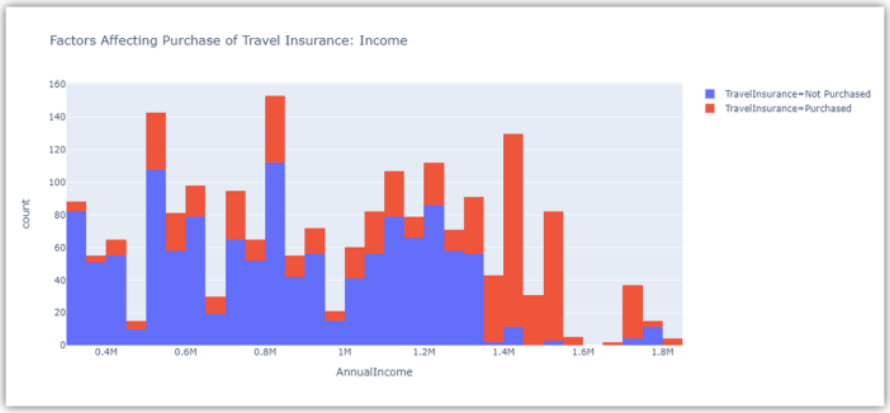
وفقًا للمخطط أعلاه، من المرجح أن يشتري الأشخاص الذين يبلغون من العمر 34 عامًا بوليصة تأمين ويقل احتمال شراء الأشخاص حول 28 عامًا بوليصة التأمين. دعنا الآن نرى كيف يؤثر نوع عمل الشخص على شراء بوليصة التأمين:

```
import plotly.express as px
data = data
figure = px.histogram(data, x = "Employment Type",
                      color = "TravelInsurance",
                      title= "Factors Affecting Purchase of Travel
Insurance: Employment Type")
figure.show()
```



وفقًا للمخطط أعلاه، من المرجح أن يكون لدى الأشخاص العاملين في القطاع الخاص أو العاملين لحسابهم الخاص بوليصة تأمين. دعنا الآن نرى كيف يؤثر الدخل السنوي للفرد على شراء بوليصة التأمين:

```
import plotly.express as px
data = data
figure = px.histogram(data, x = "AnnualIncome",
                      color = "TravelInsurance",
                      title= "Factors Affecting Purchase of Travel
Insurance: Income")
figure.show()
```



وفقًا للمخطط أعلاه، فإن الأشخاص الذين لديهم دخل سنوي يزيد عن 1400000 هم أكثر عرضة لشراء بوليصة التأمين.

تعتمد مجموعة البيانات التي نستخدمها على مشتريات تأمين السفر travel insurance، لذا فمن الأرجح أن الأشخاص الذين يكسبون دخلاً أعلى يسافرون أكثر، ونتيجة لذلك، من المرجح أن يشتروا تأمين السفر. هذه هي الطريقة التي يمكنك بها استكشاف كل عمود من هذه البيانات بسهولة. الآن في القسم أدناه، سأطلعك على كيفية تدريب نموذج التعلم الآلي للتنبؤ بما إذا كان الشخص سيشتري تأمين السفر أم لا.

نموذج التنبؤ بالتأمين

سأحول جميع القيم الفئوية إلى 1 و 0 أولاً لأن جميع الأعمدة مهمة لتدريب نموذج التنبؤ بالتأمين:

```
import plotly.express as px
data = data
figure = px.histogram(data, x = "AnnualIncome",
                      color = "TravelInsurance",
                      title= "Factors Affecting Purchase of Travel
Insurance: Income")
figure.show()
```

الآن دعنا نقسم البيانات ونقوم بتدريب النموذج باستخدام خوارزمية تصنيف شجرة القرار:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.10,
random_state=42)
model = DecisionTreeClassifier()
model.fit(xtrain, ytrain)
predictions = model.predict(xtest)
```

```
0.8190954773869347
```

يعطي النموذج درجة تزيد عن 80٪ وهي ليست سيئة لهذا النوع من المشاكل. هذه هي الطريقة التي يمكنك بها تدريب نموذج التعلم الآلي لمهمة التنبؤ بالتأمين باستخدام Python.

الملخص

هذه هي الطريقة التي يمكنك من خلالها تحليل أي نوع من الأشخاص من المرجح أن يشتري بوليصة تأمين وتدريب نموذج التعلم الآلي لنفسه. مهمة التنبؤ بالتأمين هي شيء يضيف قيمة إلى كل شركة تأمين. يستخدمون بيانات من قاعدة بياناتهم حول كل شخص اتصلوا بهم للترويج لخدمات التأمين ومحاولة العثور على الأشخاص الأكثر احتمالية الذين يمكنهم شراء التأمين. آمل أن تكون قد أحببت هذه المقالة حول مهمة توقع التأمين باستخدام التعلم الآلي باستخدام بايثون.

29) توقع درجات الطلاب باستخدام التعلم الآلي Student Grades Prediction using Machine Learning

في الدراسات العليا، يجد العديد من الطلاب صعوبةفي تحقيق درجات جيدة لأنهم لا يحصلون على دعم كبيرفي دورات التعليم العالي مقارنة بالدعم الذي يتلقاه الطلابفي المدارس. يمكننا استخدام التعلم الآلي لمهمة التنبؤ بدرجات الطالب حتى يتمكن المدرسون من مساعدة الطلاب على الاستعداد للموضوعات التي تم توقع انخفاض درجات الطلاب فيها.في هذه المقالة، سوف أطلعك على مهمة التنبؤ بدرجات الطلاب باستخدام التعلم الآلي باستخدام Python.

توقع درجات الطالب

الجامعات هي أماكن مرموقة للغاية للوصول إلى التعليم العالي. لكن مقدار الرسوم التي تفرضها الجامعات اليوم لا يساوي أبدًا الدعم الذي تقدمه للطلاب. يحتاج بعض الطلاب إلى الكثير من الاهتمام من المعلمين لأنه إذا لم يتم إيلاء اهتمام خاص لأولئك الطلاب الذين لم يحصلوا على درجات جيدة، فقد يكون ذلك ضارًا بحالتهم العاطفية وحياتهم المهنية على المدى الطويل.

باستخدام خوارزميات التعلم الآلي، يمكننا التنبؤ بمدى جودة أداء الطلاب حتى نتمكن من مساعدة الطلاب الذين من المتوقع أن تكون درجاتهم منخفضة. يعتمد توقع درجات الطلاب على مشكلة الانحدارفي التعلم الآلي.في القسم أدناه، سوف آخذك خلال مهمة التنبؤ بدرجات الطلاب باستخدام التعلم الآلي باستخدام Python.

توقع درجات الطالب باستخدام لغة بايثون

آمل أن تكون قد فهمت الآن سبب حاجتنا للتنبؤ بدرجات الطالب. دعنا الآن نرى كيف يمكننا استخدام التعلم الآلي لمهمة التنبؤ بدرجات الطلاب باستخدام Python. سأبدأ هذه المهمة عن طريق استيراد مكتبات Python ومجموعة البيانات اللازمة:

مجموعة البيانات.

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.utils import shuffle

data = pd.read_csv("student-mat.csv")
data.head()
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	course	mother
1	GP	F	17	U	GT3	T	1	1	at_home	other	course	father
2	GP	F	15	U	LE3	T	1	1	at_home	other	other	mother
3	GP	F	15	U	GT3	T	4	2	health	services	home	mother
4	GP	F	16	U	GT3	T	3	3	other	other	home	father

تستند مجموعة البيانات التي أستخدمها لمهمة التنبؤ بدرجات الطلاب إلى إنجازات طلاب المدارس البرتغالية.في مجموعة البيانات هذه، يمثل G1 درجات الفترة الأولى، ويمثل G2 درجات الفترة الثانية، ويمثل G3 الدرجات النهائية. الآن دعنا نجهز البيانات ودعنا نرى كيف يمكننا توقع الدرجات النهائية للطلاب:

```
data = data[["G1", "G2", "G3", "studytime", "failures", "absences"]]
predict = "G3"
x = np.array(data.drop([predict], 1))
y = np.array(data[predict])

from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2)
```

في الكود أعلاه، قمت أولاً بتحديد الأعمدة الضرورية التي نحتاجها لتدريب نموذج التعلم الآلي لمهمة التنبؤ بدرجات الطلاب. ثم أعلنت أن عمود G3 هو التسمية المستهدفة لدينا، ثم قسمت مجموعة البيانات إلى اختبار بنسبة 20٪ وتدريب بنسبة 80٪. دعنا الآن نرى كيفية تدريب نموذج الانحدار الخطي linear regression لمهمة التنبؤ بدرجات الطلاب:

```
linear_regression = LinearRegression()
linear_regression.fit(xtrain, ytrain)
accuracy = linear_regression.score(xtest, ytest)
print(accuracy)
```

```
0.8432876775479776
```

أعطى نموذج الانحدار الخطي دقة حوالي 84٪ وهي ليست سيئةفي هذه المهمة. الآن دعنا نلقي نظرة على التنبؤات التي قدمها نموذج التنبؤ بالدرجات للطلاب:

```
predictions = linear_regression.predict(xtest)
for i in range(len(predictions)):
    print(predictions[x], xtest[x], [ytest[x]])
```

```
[[16.16395534 14.23423176 14.08532841  5.28096434 14.23423176]
 [16.16395534 16.16395534 14.08532841  5.28096434  7.97291422]
 [14.52779998 11.92149651 14.08532841  9.13993948  4.71694746]
 ...
 [ 4.71694746 11.92149651  3.9451298   9.13993948  9.13993948]
 [12.56424351  4.92497623  3.9451298   5.28096434  5.28096434]
 [11.92149651  9.05247158  3.9451298   5.28096434 16.16395534]] [[[15 16  2  0  2]
  [15 14  2  0  2]
  [15 14  3  0  6]
  [ 7  6  2  0 10]
  [15 14  2  0  2]]....
```

الملخص

هذه هي الطريقة التي يمكنك بها تدريب نموذج الانحدار الخطي لمهمة تنبؤ درجات الطلاب باستخدام التعلم الآلي باستخدام Python. يمكنك القيام بالكثير باستخدام مجموعة البيانات هذه، ويمكنك العثور على المعلومات الكاملة حول مجموعة البيانات هذه من هنا.

30) توقع أسعار الطيران باستخدام التعلم الآلي Flight Price Prediction using Machine Learning

في هذه المقالة، سنقوم بتنفيذ نموذج توقع أسعار الطيران باستخدام تقنيات مختلفة، كما سنقوم ببعض تصورات البيانات لفهم بياناتنا بشكل أفضل.

قم بإنشاء بيئة conda وقم بتثبيت المكتبات المطلوبة:

```
conda create -n fpp python=3.9
conda activate fpp
pip install flask flask_cors pandas seaborn sklearn openpyxl
flask run
```

الخطوة 1: استيراد المكتبات المطلوبة لتنبؤ أسعار الطيران.

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
from sklearn.model_selection import train_test_split,
RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor,
ExtraTreesRegressor
import pickle
```

الخطوة 2: قراءة بيانات التدريب.

```
train_data = pd.read_excel('Flight Dataset/Data_Train.xlsx')
train_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

الخطوة 3: التحقق من القيم في عمود الوجهة.

```
train_data['Destination'].value_counts()
```

- سيذهب الحد الأقصى من الأشخاص إلى كوتشي تليها بنغالور ثم دلهيفي مجموعة البيانات الخاصة بنا.

```
Cochin       4537
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

الخطوة 3.5 - دمج دلهي ونيودلهي.

```
def newd(x):
    if x=='New Delhi':
        return 'Delhi'
```

```
    else:
        return x

train_data['Destination'] = train_data['Destination'].apply(newd)
```

- كما رأينا أعلاه، وجهتنا هي دلهي ونيودلهي لذلك قمنا بدمجهما.

الخطوة 4: التحقق من معلومات بيانات التدريب الخاصة بنا.

```
train_data.info()
```

- يمكننا أن نرى أن المسار وإجمالي التوقفات بها 11 قيم فارغة (NULL) لكل منهما.
- لذلك سنقوم بإسقاط قيم NULL أكثر.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

الخطوة 5: جعل أعمدة اليوم والشهر أعمدة Datetime.

```
train_data['Journey_day'] =
pd.to_datetime(train_data['Date_of_Journey'],format='%d/%m/%Y').dt.day
train_data['Journey_month'] =
pd.to_datetime(train_data['Date_of_Journey'],format='%d/%m/%Y').dt.mon
th

train_data.drop('Date_of_Journey',inplace=True,axis=1)

train_data.head()
```

- سنقوم باستخراج يوم الرحلة وشهر الرحلة من تاريخ الرحلة ونقوم بعمل عمودين لهما كما هو موضح أدناه.
- ثم سنقوم بإسقاط عمود تاريخ الرحلة journey column.

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month
0	IndiGo	Banglore	Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	3
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662	1	5
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882	9	6
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218	12	5
4	IndiGo	Banglore	Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302	1	3

الخطوة 6: استخلاص الساعات والدقائق من الوقت.

```
train_data['Dep_hour'] =
pd.to_datetime(train_data['Dep_Time']).dt.hour
train_data['Dep_min'] =
pd.to_datetime(train_data['Dep_Time']).dt.minute
train_data.drop('Dep_Time',axis=1,inplace=True)

train_data['Arrival_hour'] =
pd.to_datetime(train_data['Arrival_Time']).dt.hour
train_data['Arrival_min'] =
pd.to_datetime(train_data['Arrival_Time']).dt.minute
train_data.drop('Arrival_Time',axis=1,inplace=True)

train_data.head()
```

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival
0	IndiGo	Banglore	Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	22	20	1	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	5	5	50	13	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	9	6	9	25	4	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	12	5	18	5	23	
4	IndiGo	Banglore	Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	1	3	16	50	21	

- كما هو مذكور أعلاه، سنستخرج ساعة المغادرة ودقائق المغادرة من وقت المغادرة departure time.
- ونفس الشيء سيتم القيام به لوقت الوصول arrival time.
- وبعد ذلك، سنقوم بإسقاط كلا العمودين.

الخطوة 7: التحقق من القيم في عمود المدة.

```
train_data['Duration'].value_counts()
```

- هذه هي فترات الرحلات الجوية.
- 550 رحلة مدتها ساعتان و50 دقيقة وهكذا.

```
2h 50m     550
1h 30m     386
2h 55m     337
2h 45m     337
2h 35m     329
          ...
4h 10m       1
41h 20m      1
31h 30m      1
29h 30m      1
30h 10m      1
Name: Duration, Length: 368, dtype: int64
```

الخطوة 8: إسقاط عمود Duration واستخراج المعلومات المهمة منه.

```
duration = list(train_data['Duration'])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if 'h' in duration[i]:
            duration[i] = duration[i] + ' 0m'
        else:
            duration[i] = '0h ' + duration[i]

duration_hour = []
duration_min = []

for i in duration:
    h,m = i.split()
    duration_hour.append(int(h[:-1]))
    duration_min.append(int(m[:-1]))

train_data['Duration_hours'] = duration_hour
train_data['Duration_mins'] = duration_min

train_data.drop('Duration',axis=1,inplace=True)
train_data.head()
```

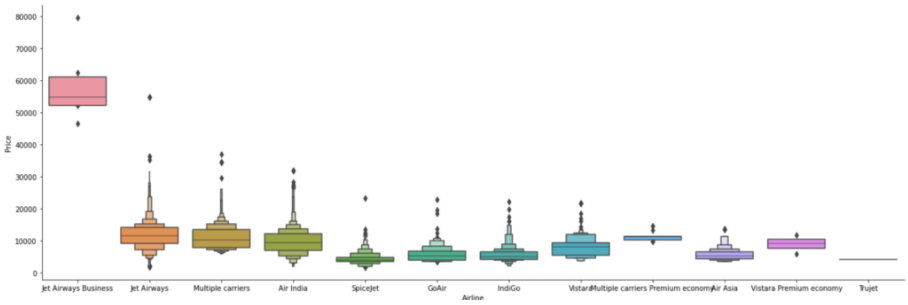
- **السطر الأول:** إنشاء قائمة بجميع الفترات الموجودةفي البيانات.
- **السطر 3ـ8**: نقوم فقط بإحضار كل فترة بنفس التنسيق. قد تكون هناك حالة عندما تكون فترة الرحلة 30 دقيقة فقط، لذا سنكتبها على أنها "0 س 30 د" وقد تكون هناك أيضًا حالات مثل ساعتان، لذا سنكتبها على أنها "2 س 0 د".
- **السطر 13ـ16**: قم ببساطة بتقسيمه إلى مكونين، ساعة ودقيقة.
- **السطر 18ـ19**: أضف عمودين "Duration_hours" و "Duration_mins".
- **السطر 21**: قم بإسقاط عمود "Duration " الأصلي.

ination	Route	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins
w Delhi	BLR → DEL	non-stop	No info	3897	24	3	22	20	1	10	2	50
angiore	CCU → IXR → BBI → BLR	2 stops	No info	7662	1	5	5	50	13	15	7	25
Cochin	DEL → LKO → BOM → COK	2 stops	No info	13882	9	6	9	25	4	25	19	0
angiore	CCU → NAG → BLR	1 stop	No info	6218	12	5	18	5	23	30	5	25
w Delhi	BLR → NAG → DEL	1 stop	No info	13302	1	3	16	50	21	35	4	45

الخطوة 9: رسم المخطط لشركة الطيران مقابل السعر.

```
sns.catplot(x='Airline',y='Price',data=train_data.sort_values('Price',
ascending=False),kind='boxen',aspect=3,height=6)
```

- من المخطط أدناه يمكننا أن نستنتج أن أعمال Jet Airways هي أغلى الخطوط الجوية.



الخطوة 10: قم بإنشاء أعمدة وهمية من عمود شركة الطيران.

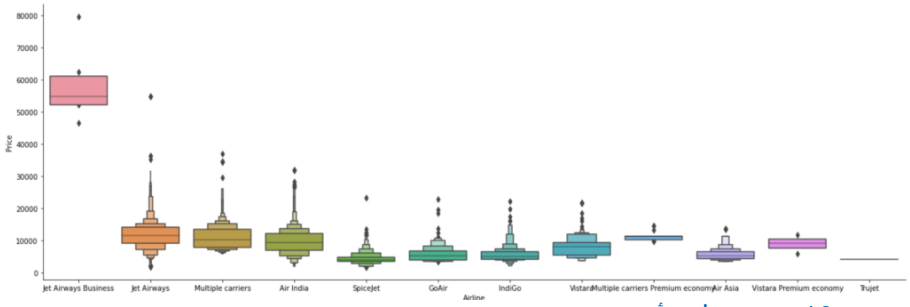
```
airline = train_data[['Airline']]
airline = pd.get_dummies(airline,drop_first=True)
```

- نظرًا لأن Airline عبارة عن عمود فئوي، فسنقوم بإخراج أعمدة وهمية dummy columns منه.

الخطوة 11: رسم المصدر مقابل السعر.

```
# If we are going from Banglore the prices are slightly higher as
compared to other cities
sns.catplot(x='Source',y='Price',data=train_data.sort_values('Price',a
scending=False),kind='boxen',aspect=3,height=4)
```

- يوضح المخطط أدناه أنه إذا كنت مسافرًا من بنغالور، فبغض النظر عن المكان الذي يتعين عليك فيه دفع أكبر مبلغ من المال.



الخطوة 12: قم بإنشاء أعمدة وهمية من عمود المصدر.

```
source = train_data[['Source']]
source = pd.get_dummies(source,drop_first=True)
source.head()
```

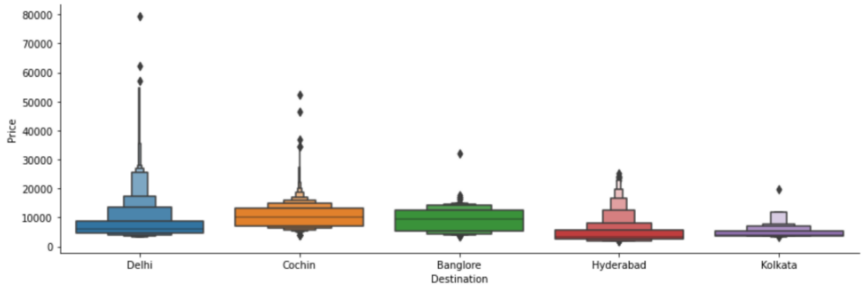
- نظرًا لأن المصدر هو عمود فئوي، لذلك سنقوم بإخراج أعمدة وهمية منه.

	Source_Chennai	Source_Delhi	Source_Kolkata	Source_Mumbai
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	0

الخطوة 13: رسم مخطط الوجهة مقابل السعر.

```
# If we are going to New Delhi the prices are slightly higher as
compared to other cities
sns.catplot(x='Destination',y='Price',data=train_data.sort_values('Pri
ce',ascending=False),kind='boxen',aspect=3,height=4)
```

- يوضح المخطط أدناه أنه إذا كنت ذاهبًا إلى نيودلهي، بغض النظر عن المكان، يجب عليك دفع أكبر مبلغ من المال.



الخطوة 14: قم بإنشاء أعمدة وهمية من عمود الوجهة.

```
destination = train_data[['Destination']]
destination = pd.get_dummies(destination,drop_first=True)
destination.head()
```

- نظرًا لأن الوجهة هي أيضًا عمود فئوي، لذلك سنقوم بإخراج أعمدة وهمية منه.

	Destination_Cochin	Destination_Delhi	Destination_Hyderabad	Destination_Kolkata
0	0	1	0	0
1	0	0	0	0
2	1	0	0	0
3	0	0	0	0
4	0	1	0	0

الخطوة 15: إسقاط الاعمدة غير المرغوب بها.

```
train_data.drop(['Route','Additional_Info'],inplace=True,axis=1)
```

الخطوة 16: التحقق من القيم في عمود إجمالي التوقفات.

```
train_data['Total_Stops'].value_counts()
```

```
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```

الخطوة 17: تحويل التسميات إلى أرقام في عمود Total_stops.

```
# acc to the data, price is directly prop to the no. of stops
train_data['Total_Stops'].replace({'non-stop':0,'1 stop':1,'2
stops':2,'3 stops':3,'4 stops':4},inplace=True)
train_data.head()
```

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_m
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	10	2	
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	15	7	
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	25	19	
3	IndiGo	Kolkata	Banglore	1	6218	12	5	18	5	23	30	5	
4	IndiGo	Banglore	New Delhi	1	13302	1	3	16	50	21	35	4	

الخطوة 18: التحقق من أشكال إطارات البيانات الأربعة الخاصة بنا.

```
print(airline.shape)
print(source.shape)
print(destination.shape)
print(train_data.shape)
```

- تحتوي كل إطارات البيانات الأربعة هذه على نفس عدد الصفوف، مما يعني أننا قمنا بكل شيء بشكل صحيح.
- والآن يمكننا الانضمام إليهم.

```
(10682, 11)
(10682, 4)
(10682, 5)
(10682, 13)
```

الخطوة 19: اجمع كل إطارات البيانات الأربعة.

```
data_train = pd.concat([train_data,airline,source,destination],axis=1)
data_train.drop(['Airline','Source','Destination'],axis=1,inplace=True
)
data_train.head()
```

- انضم إلى جميع إطارات البيانات الأربعة.
- قم بإسقاط أعمدة الخطوط الجوية Airline والمصدر Source والوجهة Destination.

	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins	...	Airline_Vistara Premium economy	Source_
0	0	3897	24	3	22	20	1	10	2	50	...	0	
1	2	7662	1	5	5	50	13	15	7	25	...	0	
2	2	13882	9	6	9	25	4	25	19	0	...	0	
3	1	6218	12	5	18	5	23	30	5	25	...	0	
4	1	13302	1	3	16	50	21	35	4	45	...	0	

5 rows × 30 columns

الخطوة 20: أخذ بيانات التدريب.

```
X = data_train.drop('Price',axis=1)
X.head()
```

الخطوة 21: خذ تسميات بيانات التدريب.

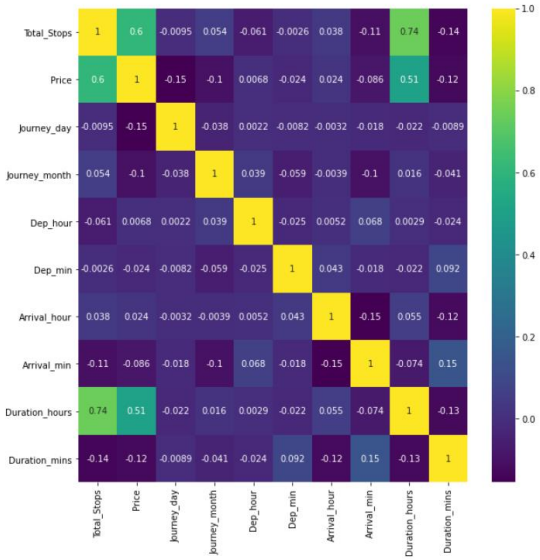
```
y = data_train['Price']
y.head()
```

```
0     3897
1     7662
2    13882
3     6218
4    13302
Name: Price, dtype: int64
```

الخطوة 22: التحقق من الارتباطات بين الأعمدة.

```
plt.figure(figsize=(10,10))
sns.heatmap(train_data.corr(),cmap='viridis',annot=True)
```

- مجرد التحقق من الارتباط بين السمات المختلفة لبيانات التدريب.
- يمكننا أن نرى أن Total_stops يرتبط ارتباطًا وثيقًا بـ Duration_hours وهو أمر واضح جدًا. إذا زاد عدد التوقفات، فستزيد أيضًا مدة الرحلة.
- أيضًا، يرتبط السعر ارتباطًا وثيقًا بإجمالي نقاط التوقف لأنهفي حالة زيادة التوقفات، فإن ذلك يتطلب أيضًا كمية كبيرة من الوقود، وهذا من شأنه زيادة السعر.



الخطوة 23: جرب أولاً نموذج ExtraTreesRegressor للتنبؤ بسعر الرحلة.

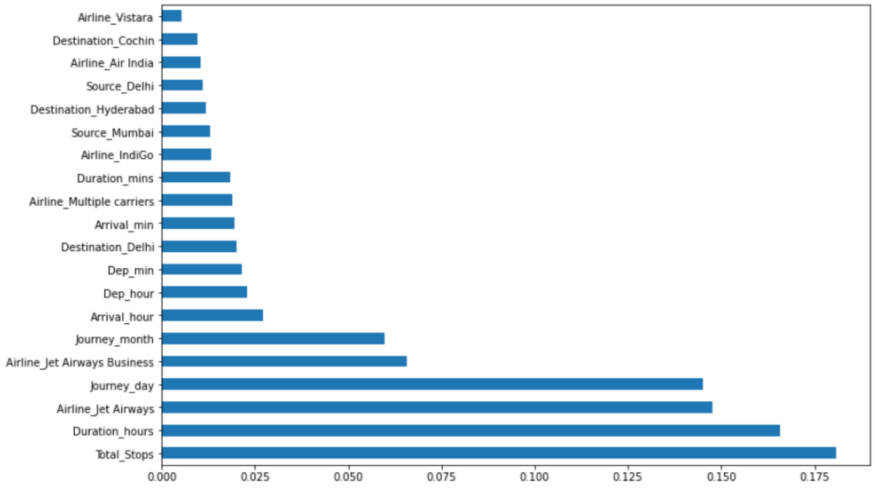
```
reg = ExtraTreesRegressor()
reg.fit(X,y)

print(reg.feature_importances_)
```

- دعونا نلائم بياناتنافي ExtraTreeRegressor ونحلل أهمية الميزات.

```
[1.80724905e-01 1.45097438e-01 5.97393938e-02 2.29072716e-02
 2.14236783e-02 2.72528146e-02 1.95937136e-02 1.65708966e-01
 1.83608184e-02 1.04019292e-02 1.77576707e-03 1.32195548e-02
 1.47739235e-01 6.58602376e-02 1.88771000e-02 9.89160674e-04
 3.74142540e-03 2.58326660e-05 5.38193708e-03 3.83297614e-05
 8.57103668e-04 1.09368849e-02 4.13799449e-03 1.29028159e-02
 9.47279541e-03 1.99945794e-02 1.18121866e-02 1.02613024e-03]
```

الخطوة 24: التحقق من أهمية الميزة التي قدمتها ExtraTreeRegressor.

- Total_stops هي الميزة ذات الأهمية القصوىفي تحديد السعر كما رأينا أعلاه أيضًا.
- بعد ذلك، يلعب يوم الرحلة Journey Day أيضًا دورًا كبيرًافي تحديد السعر. الأسعار أعلى بشكل عامفي عطلات نهاية الأسبوع.



الخطوة 25: تقسيم بياناتنا إلى بيانات تدريب واختبار.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 42)
```

الخطوة 26: تدريب نموذج الانحدار للغابات العشوائية للتنبؤ بسعر الطيران.

```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200,
num = 12)]

# Number of features to consider at every split
max_features = ['auto', 'sqrt']

# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
```

```
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]

# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}


# Random search of parameters, using 5 fold cross validation, search
across 100 different combinations
rf_random = RandomizedSearchCV(estimator = RandomForestRegressor(),
param_distributions = random_grid,
                               scoring='neg_mean_squared_error',
n_iter = 10, cv = 5,
                               verbose=1, random_state=42, n_jobs = 1)
rf_random.fit(X_train,y_train)
```

- نحن هنا نستخدم RandomizedSearchCV الذي يجرب المجموعات بشكل عشوائي ويرى أيها هو الأفضل منها.
- لقد أعلنا عن معلمات RandomForestRegressor التي نريد تجربتها.

الخطوة 27: التحقق من أفضل المعلمات التي حصلنا عليها باستخدام البحث العشوائي.

```
rf_random.best_params_
```

```
{'n_estimators': 700,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

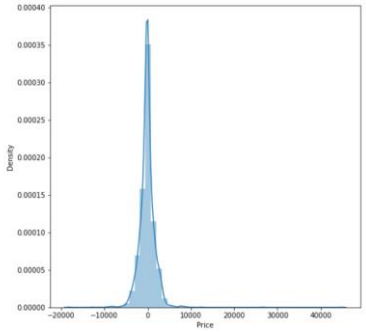
الخطوة 28: أخذ التنبؤات

```
# Flight Price Prediction
prediction = rf_random.predict(X_test)
```

الخطوة 29: رسم مخططات القيم المتبقية.

```
plt.figure(figsize = (8,8))
sns.distplot(y_test-prediction)
plt.show()
```

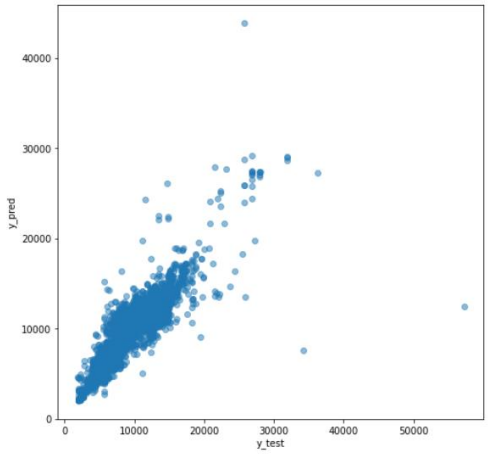
- كما يمكننا أن نرى أن معظم القيم المتبقية هي 0، مما يعني أن نموذجنا يتم تعميمه جيداً.



الخطوة 30: رسم y_test مقابل التنبؤات.

```
plt.figure(figsize = (8,8))
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```

- ببساطة نخطط لتوقعاتنا مقابل القيم الحقيقية.
- من الناحية المثالية، يجب أن يكون خطًا مستقيمًا.



الخطوة 31: طباعة المقاييس.

```
print('r2 score: ', metrics.r2_score(y_test,y_pred))
```

```
r2 score:  0.8114033815110862
```

الخطوة 32: حفظ النموذج.

```
file = open('flight_rf.pkl', 'wb')
pickle.dump(rf_random, file)
```

31) توقع فئة الوزن باستخدام التعلم الآلي Weight Category prediction using Machine Learning

لذلكفي مقالة اليوم، سنقوم بالتنبؤ بفئة الوزن لشخص معطى الطول والوزن والجنس بمساعدة خوارزمية Random Forest.

كود توقع فئة الوزن

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score

data = pd.read_csv('500_Person_Gender_Height_Weight_Index.csv')
print(data.describe())

def give_names_to_indices(ind):
    if ind==0:
        return 'Extremely Weak'
    elif ind==1:
        return 'Weak'
    elif ind==2:
        return 'Normal'
    elif ind==3:
        return 'OverWeight'
    elif ind==4:
        return 'Obesity'
    elif ind==5:
        return 'Extremely Obese'


data['Index'] = data['Index'].apply(give_names_to_indices)

sns.lmplot('Height','Weight',data,hue='Index',size=7,aspect=1,fit_reg=
False)

people = data['Gender'].value_counts()

categories = data['Index'].value_counts()

# STATS FOR MEN
data[data['Gender']=='Male']['Index'].value_counts()

# STATS FOR WOMEN
data[data['Gender']=='Female']['Index'].value_counts()

data2 = pd.get_dummies(data['Gender'])
data.drop('Gender',axis=1,inplace=True)
data = pd.concat([data,data2],axis=1)

y=data['Index']
data =data.drop(['Index'],axis=1)

scaler = StandardScaler()
data = scaler.fit_transform(data)
```

```
data=pd.DataFrame(data)

X_train, X_test, y_train, y_test = train_test_split(data, y,
test_size=0.3, random_state=101)

param_grid = {'n_estimators':[100,200,300,400,500,600,700,800,1000]}
grid_cv =
GridSearchCV(RandomForestClassifier(random_state=101),param_grid,verbo
se=3)

grid_cv.fit(X_train,y_train)

print(grid_cv.best_params_)
# weight category prediction
pred = grid_cv.predict(X_test)


print(classification_report(y_test,pred))
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print('Acuuracy is --> ',accuracy_score(y_test,pred)*100)
print('\n')

def lp(details):
    gender = details[0]
    height = details[1]
    weight = details[2]

    if gender=='Male':

details=np.array([[np.float(height),np.float(weight),0.0,1.0]])
    elif gender=='Female':

details=np.array([[np.float(height),np.float(weight),1.0,0.0]])

    y_pred = grid_cv.predict(scaler.transform(details))
    return (y_pred[0])



#Live predictor
your_details = ['Male',175,80]
print(lp(your_details))
```

- **السطر 1ـ8**: استيراد المكتبات المطلوبة.
- **السطر 10**: قراءة بياناتنا.

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3

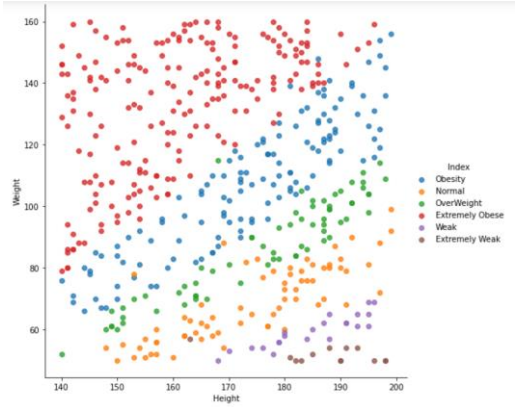
- **السطر 11**: وصف بياناتنا.

	Height	Weight	Index
count	500.000000	500.000000	500.000000
mean	169.944000	106.000000	3.748000
std	16.375261	32.382607	1.355053
min	140.000000	50.000000	0.000000
25%	156.000000	80.000000	3.000000
50%	170.500000	106.000000	4.000000
75%	184.000000	136.000000	5.000000
max	199.000000	160.000000	5.000000

- **السطر 13ـ25**: أنشئ دالة لتحويل القيم العددية لعمود الفهرس إلى قيم فئوية.
- **السطر 28:** قم بتطبيق هذه الدالة لإجراء تغييرات.

	Gender	Height	Weight	Index
0	Male	174	96	Obesity
1	Male	189	87	Normal
2	Female	185	110	Obesity
3	Female	195	104	OverWeight
4	Male	149	61	OverWeight

- **السطر 30**: لنرسم الطول مقابل الوزن ولونهما وفقًا لفئة الوزن. نحن نستخدم sns.lmplot التي هي مجرد مخطط مبعثر أو يمكننا أن نقول مخطط الانحدار.



- **السطر 32**: دعونا نحلل أعداد القيم value counts لعمود الجنس Gender column.

```
Female    255
Male      245
Name: Gender, dtype: int64
```

- **السطر 34**: دعنا نحلل عدد قيم value counts عمود الفهرس Index column.

```
Extremely Obese    198
Obesity            130
Normal              69
OverWeight          68
Weak                22
Extremely Weak      13
Name: Index, dtype: int64
```

- **السطر 36ـ40**: دعنا نحلل توزيع فئة الوزن وفقًا للجنس.

```
Extremely Obese    105
Obesity             59
OverWeight          32
Normal              28
Weak                15
Extremely Weak       6
Name: Index, dtype: int64
```

- **السطر 42ـ44**: الآن عندما ننتهي من استكشاف البيانات، فلنبدأفي بناء النموذج. لا يمكننا إعطاء نموذجنا الميزات الفئوية. لذلك سننشئ متغيرات وهمية dummy variables من عمود الجنس، ثم سنقوم بإسقاط عمود الجنس الأصلي ودمج هذين المتغيرين الوهميين مع إطار البيانات الرئيسي لدينا.

Female	Male
0	1
0	1
1	0
1	0
0	1

	Height	Weight	Index	Female	Male
0	174	96	Obesity	0	1
1	189	87	Normal	0	1
2	185	110	Obesity	1	0
3	195	104	OverWeight	1	0
4	149	61	OverWeight	0	1

- **السطر 46**: إنشاء بيانات y التي ستكون عمود الفهرس الخاص بنا، لأن هذا هو ما نريد أن يتنبأ به نموذجنا. (قيم الإخراج)
- **السطر 47**: إنشاء بيانات X التي سيتم تدريب نموذجنا عليها. (قيم الإدخال).

	Height	Weight	Female	Male
0	174	96	0	1
1	189	87	0	1
2	185	110	1	0
3	195	104	1	0
4	149	61	0	1

- **السطر 49ـ51**: نعلن عن مقياس قياسي Standard Scaler ونقوم بتوسيع نطاق بياناتنا لإحضار كل شيء على نفس المقياس scale / النطاق range. سيساعد هذا في زيادة دقة نموذجنا وسيساعد أيضًا في تدريب أسرع.

	0	1	2	3
0	0.247939	-0.309117	-1.020204	1.020204
1	1.164872	-0.587322	-1.020204	1.020204
2	0.920357	0.123647	0.980196	-0.980196
3	1.531645	-0.061823	0.980196	-0.980196
4	-1.280283	-1.391027	-1.020204	1.020204

- **السطر 53**: دعنا نقسم بياناتنا لأغراض التدريب والاختبار بنسب 70٪ و30٪ على التوالي.
- **السطر 55ـ56**: كان بإمكاننا ببساطة الإعلان عن نموذج الغابة العشوائية بدون أي رقم. من المقدرين ولكن باستخدام Grid Search CV، يمكننا تدريب نموذج الغابة العشوائية الخاص بنا على قيم n_estimators متعددة مثل Random Forest الغابة العشوائية الأولى سيكون لها n_estimators كـ 100، في المرة الثانية سيكون لها 200 وهكذا يمكننا التحقق من ذلك n_estimator الذي يعطي أعلى دقة. سيصبح Grid_cv تلقائيًا أفضل نموذج غابة عشوائي.
- **السطر 58**: تركيب بيانات التدريب الخاصة بنا على نموذج التنبؤ بفئة الوزن.
- **السطر 60**: دعونا نرى أفضل المعلمات.

```
{'n_estimators': 200}
```

- **السطر 62**: دعونا نجعل تنبؤات فئة الوزن على بيانات الاختبار.
- السطر 65ـ70: لنطبع تقرير التصنيف classification report ومصفوفة الارتباك confusion matrix ودرجة الدقة accuracy score لنموذج التنبؤ بفئة الوزن.

```
                    precision    recall  f1-score   support

Extremely Obese        0.91      0.97      0.94        63
 Extremely Weak        1.00      1.00      1.00         1
         Normal        0.92      0.96      0.94        23
        Obesity        0.78      0.82      0.79        38
     OverWeight        0.92      0.58      0.71        19
           Weak        0.83      0.83      0.83         6

       accuracy                           0.87       150
      macro avg        0.89      0.86      0.87       150
   weighted avg        0.88      0.87      0.87       150


[[61  0  0  2  0  0]
 [ 0  1  0  0  0  0]
 [ 0  0 22  0  0  1]
 [ 6  0  0 31  1  0]
 [ 0  0  1  7 11  0]
 [ 0  0  1  0  0  5]]

Acuuracy is -->  87.33333333333333
```

- **السطر 72ـ83**: دالة ستؤدي جميع عمليات المعالجة المسبقة للتنبؤ الحيفي الخطوة التالية.
- **السطر 87ـ89**: التنبؤ المباشر. ما عليك سوى إنشاء مصفوفة "your_details" بحيث يكون العنصر الأول كنوع الجنس، وثانيًا مثل الارتفاع، والثالث كوزن، ثم مرره إلى الدالة أعلاه للمعالجة المسبقة والتنبؤات.

```
OverWeight
```

32) التنبؤ بضريبة المنزل باستخدام التعلم الآلي House Tax Prediction using Machine Learning

في هذه المقالة، سنقوم بتنفيذ توقع ضريبة المنزل باستخدام خوارزمية الغابة العشوائية Random Forest. سنستخدم بيانات الإسكان الشهيرةفي بوسطن لحل هذه المشكلة.

الخطوة 1: استيراد الحزم المطلوبة.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

%matplotlib inline
```

الخطوة 2: قراءة بياناتنا.

```
data = pd.read csv('HousingData.csv')
data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	NaN	36.2

الخطوة 3: وصف بياناتنا.

```
data.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	
count	486.000000	486.000000	486.000000	486.000000	506.000000	506.000000	486.000000	506.000000	506.000000	506.000000	506.000000	506.000000	486.0
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519	3.795043	9.549407	408.237154	18.455534	356.674032	12.7
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	27.999513	2.105710	8.707259	168.537116	2.164946	91.294864	7.1
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.7
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	45.175000	2.100175	4.000000	279.000000	17.400000	375.377500	7.1
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	76.800000	3.207450	5.000000	330.000000	19.050000	391.440000	11.4
75%	3.560262	12.500000	18.100000	0.000000	0.624000	6.623500	93.975000	5.188425	24.000000	666.000000	20.200000	396.225000	16.9
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.9

الخطوة 4: تحقق من معلومات بياناتنا.

```
data.info()
```

- كما نرى أدناه، يوجد 14 أعمدة لا تحتوي على قيم 506، مما يعني أنها تحتوي على بعض القيم الخالية.
- لهذا السبب نحتاج إلى ملء هذه القيم الخالية لجعل كل عمود بحجم متساوٍ.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    int64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

الخطوة 5: تعبئة القيم الخالية.

```
col = ['CRIM','ZN','INDUS','CHAS','AGE','LSTAT']
for c in col:
    data[c].fillna(data[c].mean(),inplace=True)
```

- هنا نقوم ببساطة بملء الحقول الفارغة بمتوسط هذا العمود.
- لقد تجاوزنا "inplace = True" لأننا نريد إجراء هذا التغييرفي مجموعة البيانات الأصلية.

الخطوة 6: الآن تحقق مرة أخرى من معلومات بياناتنا.

```
data.info()
```

- يمكننا أن نرى أدناه، أن جميع الأعمدة تحتوي الآن على 506 قيم غير فارغة.
- هذه خطوة مهمة للغايةفي المعالجة المسبقة للبيانات. يجب علينا إما إزالة الإدخالات بقيم فارغة أو ملئها ببعض القيم الافتراضية أو يمكننا أيضًا ملئها بمتوسط القيمة كما فعلنافي هذه الحالة.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    int64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

الخطوة 7: تحقق من ارتباط الحقل المستهدف "TAX" بالميزات الأخرى.

```
data.corr()['TAX'].sort_values(ascending=False)
```

- نحن هنا نتحقق فقط من ارتباط العمود المستهدف لدينا وهو "TAX" مع الأعمدة الأخرى.
- يمكننا أن نرى أن لها أعلى ارتباط مع عمود "RAD". هذا يعني أن قيمة الضريبة تتناسب طرديًا مع قيمة RAD.

```
TAX        1.000000
RAD        0.910228
INDUS      0.731055
NOX        0.668023
CRIM       0.580595
LSTAT      0.536110
AGE        0.509114
PTRATIO    0.460853
CHAS      -0.032304
RM        -0.292048
ZN        -0.312371
B         -0.441808
MEDV      -0.468536
DIS       -0.534432
Name: TAX, dtype: float64
```

الخطوة 8: المعالجة المسبقة لبياناتنا.

```
from sklearn.preprocessing import StandardScaler

X = data.drop('TAX',axis=1)
y = data['TAX']

scaler = StandardScaler()
X = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
```

- نقوم بإنشاء X، وهي ميزاتنا / جميع الأعمدة باستثناء التسميات / عمود الضرائب.
- نقوم بإنشاء y وهو عمود الضريبة فقط.
- نقوم بتوسيع نطاق بيانات X باستخدام Standard Scaler لإخراج كل شيء إلى نفس المقياس [0ـ1].
- ثم نقوم ببساطة بتقسيم بياناتنا إلى 70ـ30 نسبًا لبيانات التدريب والاختبار على التوالي باستخدام train-test-split.

الخطوة 9: تدريب نموذج التنبؤ بالضريبة الخاص بنا.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV

rfc = RandomForestRegressor()
params = {'n_estimators':[100,200,300,400,500,600,700,800,900,1000]}

grid_model = GridSearchCV(rfc, params,verbose=2)
grid_model.fit(X_train,y_train)

pred = grid_model.predict(X_test)

print('Random Forest accuracy is --> ',r2_score(y_test,pred)*100)
```

- لقد اخترنا Random Forest Regressor لهذا البيان الخاص بالمشكلة.
- لقد جربت أيضًا الانحدار الخطي وانحدار لاسو وانحدار ريدج ولكن جميعها أعطت درجة r2 أقل من Random Forest.
- نحن هنا نستخدم Grid Search CV للعثور على أفضل قيمة لمعلمة "n_estimator" الخاصة بـ Random Forest.

```
Random Forest accuracy is -->  97.94199488798098
```

الخطوة 10: التحقق من أفضل المعلمات لنموذج التنبؤ بضرائب المنزل.

```
grid_model.best_params_
```

- باستخدام Grid Search CV، توصلنا إلى أن أفضل قيمة لـ n_estimator ستكون 700.

```
{'n_estimators': 700}
```

الخطوة 11: فقط شاهد النتائج.

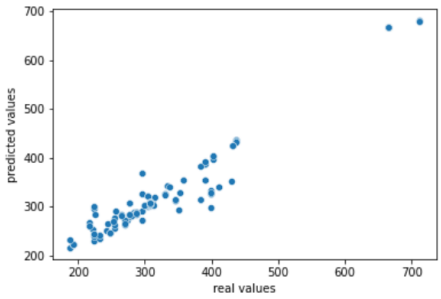
```
res = pd.DataFrame()
res['Y_Test'] = y_test
res['PRED'] = pred
res.head()
```

- مجرد خطوة بسيطة حيث نرى ما هو توقعنا وما يجب أن يكون.

	Y_Test	PRED
173	296	369.225714
274	254	255.244286
491	711	680.581429
72	305	302.314286
452	666	666.000000

الخطوة 12: رسم نتائج التنبؤ بضريبة المنزل.

```
sns.scatterplot(y_test,pred)
plt.xlabel('real values')
plt.ylabel('predicted values')
```



تنزيل الكود المصدر للتنبؤ بضريبة المنزل من هنا.

33) التعرف على الوجوه باستخدام التعلم الآلي Face Recognition using Machine Learning

مقدمة

لذلكفي مقالة اليوم، سنرى كيف يمكننا أداء التعرف على الوجوه باستخدام KNN (خوارزمية K-Nearest Neighbours) وHaar cascades. Haar cascades سريعة جداً مقارنة بالطرق الأخرى لاكتشاف الوجوه (مثل MTCNN) ولكن مع مقايضة دقيقة. دقتها أقل قليلاً عند مقارنتها بهؤلاء الأولاد الكبار مثل MTCNNs.

سنرى سكربتينفي مقالة اليوم:

- الأول هو إضافة وجه جديد.
- والثاني هو التعرف على الوجوهفي الوقت الحقيقي باستخدام KNN.

الكود لإضافة وجه جديد:

```
import cv2
import numpy as np
import os
import pickle

face_data = []
i = 0

cam = cv2.VideoCapture(0)

facec =
cv2.CascadeClassifier('data/haarcascade_frontalface_default.xml')

name = input('Enter your name --> ')
ret = True

# Face Recognition using KNN
while(ret):
    ret, frame = cam.read()
    if ret == True:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        face_coordinates = facec.detectMultiScale(gray, 1.3, 4)

        for (x, y, w, h) in face_coordinates:
            faces = frame[y:y+h, x:x+w, :]
            resized_faces = cv2.resize(faces, (50, 50))

            if i % 10 == 0 and len(face_data) < 10:
                face_data.append(resized_faces)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        i += 1

        cv2.imshow('frames', frame)

        if cv2.waitKey(1) == 27 or len(face data) >= 10:
            break
    else:
```

```
        print('error')
        break


cv2.destroyAllWindows()
cam.release()


face_data = np.asarray(face_data)
face_data = face_data.reshape(10, -1)

if 'names.pkl' not in os.listdir('data/'):
    names = [name]*10
    with open('data/names.pkl', 'wb') as f:
        pickle.dump(names, f)
else:
    with open('data/names.pkl', 'rb') as f:
        names = pickle.load(f)

    names = names + [name]*10
    with open('data/names.pkl', 'wb') as f:
        pickle.dump(names, f)


if 'faces.pkl' not in os.listdir('data/'):
    with open('data/faces.pkl', 'wb') as w:
        pickle.dump(face_data, w)
else:
    with open('data/faces.pkl', 'rb') as w:
        faces = pickle.load(w)

    faces = np.append(faces, face_data, axis=0)
    with open('data/faces.pkl', 'wb') as w:
        pickle.dump(faces, w)
```

شرح الكود:

- **السطر 1ـ4**: استيراد المكتبات المطلوبة للتعرف على الوجوه باستخدام KNN.
- **السطر 6ـ7**: تهيئة المتغيرات.
- **السطر 9**: إنشاء كائن VideoCapture للوصول إلى كاميرا الويب. يتم تمرير الوسيطة 0 عندما نريد استخدام كاميرا الويب المدمجة لجهاز الكمبيوتر / الكمبيوتر المحمول، استخدم 1 إذا كنت تريد استخدام الكاميرا الخارجية.
- **السطر 11**:في هذا السطر، نقوم بإنشاء كائن Haarcascade لاكتشاف الوجوهفي الإطار.
- **السطر 13**: نسأل عن اسم الشخص الذي يضيف وجهه.
- **السطر 14**: دعونا نضبط ret = True (مجرد إجراء شكلي لبدء الحلقة اللانهائية).
- لنبدأ الحلقة لإجراء التعرف على الوجوه باستخدام KNN.
- **السطر 18**: نستخدم cam.read() لقراءة الإطار الحالي من كاميرا الويب.
- **السطر 19**: تقول هذه العبارة أننا إذا كنا نحصل على إطارات من كاميرا الويب دون أي خطأ، فتابع المزيد، لأنهفي هذه الحالة، سيكون ret صحيحًا.

- **السطر 20**: تحويل الصورة من RBG إلى تدرج رمادي لأن Haar Cascades تكتشف الوجوهفي الصور ذات التدرج الرمادي بكفاءة.
- **السطر 22**: دعنا نكتشف الوجوه باستخدام DiscoverMultiscale، الآن لدينا إحداثيات وجهنا مثل (x، y، w، h) حيث (x,y) هي إحداثيات أعلى يسار المستطيل حول الوجه، w هو العرض وh هو ارتفاع المستطيل.
- **السطر 24**: دعونا نجتاز traverse الوجوه.
- **السطر 25**: لنستخرج الوجه من الإطار ونغيّر حجمه إلى 50 × 50.
- **السطر 28ـ29**: نقوم بتخزين الوجوهفي مصفوفة بيانات الوجه. نحتاج فقط إلى 10 وجوه لهذا السبب نتحقق من الحالة len (face_data) <10 ونحفظ الوجوه بعد كل 10 إطارات حتى نتمكن من الحصول على بعض الصور المتنوعة وليس الصور من نفس النوع / الوضع. نحن نقوم بذلك لجعل نموذجنا أكثر قوة.
- **السطر 30**: نقوم برسم مستطيل حول الوجه لعرضهفي السطر 33.
- **السطر 31**: قم ببساطة بزيادة قيمة i التي تتعقب رقم الإطار.
- **السطر 33**: يظهر أخيرًا إطارنا مع مستطيل حول الوجه المكتشف.
- **السطر 35ـ36**: إذا قام شخص ما بالضغط على مفتاح ESC أو كان عدد الصور المخزنة يساوي 10 يكسر الكود ويخرج.

- **السطر 37ـ39**: وإلا، فكسر الكود برسالة خطأ.
- **السطر 42ـ43**: فقط بعض الإجراءات الشكلية، ودمر جميع نوافذ الصور المفتوحة، ثم حرر كائن الكاميرا.
- **السطر 46ـ47**: لدينا 10 صور بحجم 50 × 50 × 3في face_data، دعنا نحولها إلى المصفوفة، ونشكلها بشكل صحيح. بعد هذه الخطوة، سيكون لدينا مصفوفة بيانات

الوجه بالشكل x750010 (10 صفوف، كل صف يصور صورة واحدة) حيث تصور 10 عدد الصور و7500 تصور الصورة المسطحة نفسها (50 × 50 × 3) (الهيكل الموضح أدناه).

- **السطر 49–52**: إذا لم يكن لدينا "names.pkl"في مجلد البيانات لدينا حتى الآن، فهذا يعني أنه الوجه الأول الذي نضيفه. لذا قم بإنشاء ملف "names.pkl" يحتوي على نفس الاسم 10 مرات (لأننا نقوم أيضًا بحفظ 10 صور لكل شخص).
- **السطر 53–59**: حالة أخرى، تعني أن لدينا "names.pkl"، وهذا يعني أنه ليس الوجه الأول الذي نضيفه، لذلك فقط قم بتحميل "names.pkl" وأضف 10 إدخالات لاسم وجهنا الحالي واحفظه باسم "names.pkl".

Sam
Sam
Sam
Sam
Sam
Sam
Sam
Sam
Sam
Sam
Nick
Nick

Structure of names.pkl

- **السطر 62–71**: كما فعلنا أعلاه للأسماء، هنا نقوم بذلك من أجل face_data.
- يقول **السطر 69** إضافة صف بيانات الوجه بشكل صفي.

Face 1 of sam	7500 columns depicting this face
Face 2 of sam	7500 columns depicting this face
Face 3 of sam	7500 columns depicting this face
Face 4 of sam	7500 columns depicting this face
Face 5 of sam	7500 columns depicting this face
Face 6 of sam	7500 columns depicting this face
Face 7 of sam	7500 columns depicting this face
Face 8 of sam	7500 columns depicting this face
Face 9 of sam	7500 columns depicting this face
Face 10 of sam	7500 columns depicting this face
Face 1 of nick	7500 columns depicting this face
Face 2 of nick	7500 columns depicting this face

Structure of faces.pkl

كود للتعرف المباشر على الوجوه باستخدام KNN:

```
import cv2
import numpy as np
import pickle
from sklearn.neighbors import KNeighborsClassifier

with open('data/faces.pkl', 'rb') as w:
    faces = pickle.load(w)

with open('data/names.pkl', 'rb') as f:
    labels = pickle.load(f)

facec =
cv2.CascadeClassifier('data/haarcascade_frontalface_default.xml')

cam = cv2.VideoCapture(0)

print('Shape of Faces matrix --> ', faces.shape)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(faces,labels)

# Face Recognition using KNN
while True:
    ret, fr = cam.read()
    if ret == True:
        gray = cv2.cvtColor(fr, cv2.COLOR_BGR2GRAY)
        face_coordinates = facec.detectMultiScale(gray, 1.3, 5)

        for (x, y, w, h) in face_coordinates:
            fc = fr[y:y + h, x:x + w, :]
            r = cv2.resize(fc, (50, 50)).flatten().reshape(1,-1)
            text = knn.predict(r)
            cv2.putText(fr, text[0], (x, y-10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2)
            cv2.rectangle(fr, (x, y), (x + w, y + w), (0, 0, 255), 2)

        cv2.imshow('livetime face recognition', fr)
        if cv2.waitKey(1) == 27:
            break
    else:
        print("error")
        break

cv2.destroyAllWindows()
```

شرح الكود:

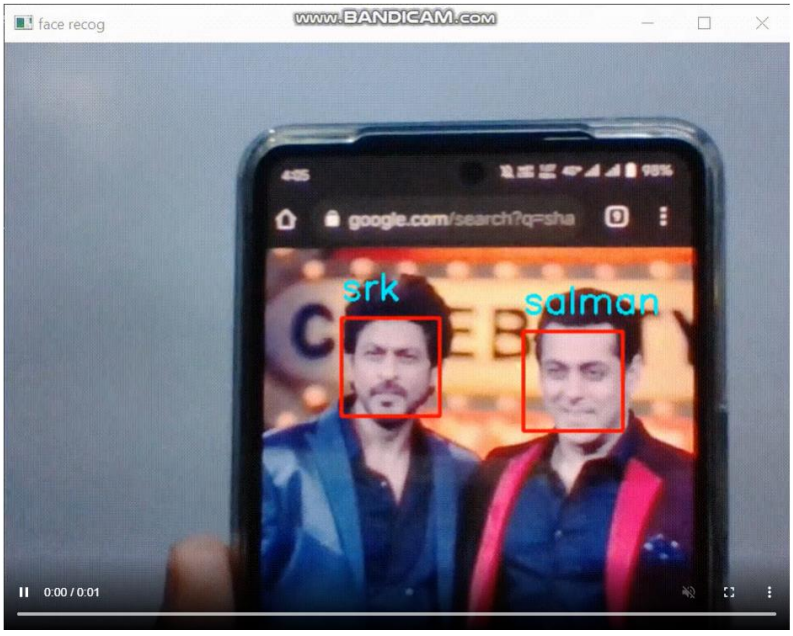
- **السطر 1ـ4**: استيراد المكتبات المطلوبة.
- **السطر 6ـ7**: تحميل بيانات الوجوه (X_train).
- **السطر 9ـ10**: تحميل بيانات الأسماء / التسميات (y_train).
- **السطر 12ـ14**: لقد ناقشنا أعلاه.
- **السطر 16**: دعنا نتحقق من شكل بيانات / مصفوفة الوجوه.

```
Shape of Faces matrix -->  (20, 7500)
```

- إنه X750020 لأنه،في وقت كتابة هذه المدونة، كانت تحتوي على وجهين فقط (10 صور لكل منهما).
- **السطر 17ـ18**: أعلنا عن كائن knn من فئة KNeighborsClassifier() مع n_neighbours = 5 ، مما يعني أنه سيتحقق فقط من أقرب 5 جيران لإعطاء النتائج.
- **السطر 21ـ29**: لقد ناقشنا هذا سابقًا. نحن فقط نجتاز الإطارات ونكتشف الوجوه فيها وتغيير حجمها.
- **السطر 30**: نقوم بعمل تنبؤات باستخدام knn.predict (r) ، حيث r هي الصورة / المصفوفة التي تم تغيير حجمها وتسويتها من 7500 نقطة.
- **السطر 31ـ32**: نقوم برسم المستطيل حول الوجه والاسمفي النافذة النهائية.
- **السطر 34**: عرض نتائجنا.
- **السطر 35ـ36**: إذا ضغطنا على مفتاح ESC، فيكسر الكود.

نتائج التعرف على الوجوه باستخدام KNN:

- أضفت وجهين، شاروخان وسلمان خان.



تنزيل الكود المصدري للتعرف على الوجوه باستخدام KNN من هنا.

34) تحليل ABC باستخدام التعلم الآلي ABC Analysis using Machine Learning

يفترض تحليل ABC أن العناصر المدرة للدخلفي المخزون تتبع توزيع Pareto، حيث تولد نسبة صغيرة جدًا من العناصر معظم الدخل.في هذه المقالة، سوف أطلعك على كيفية إجراء تحليل ABC باستخدام التعلم الآلي.

اصطلاحات تحليل ABC

باستخدام اصطلاحات تحليل ABC، يتم تخصيص حرف لصنف المخزون بناءً على أهميته:

- تمثل السلع A 20٪ من السلع، لكنها تساهم بنسبة 70٪ من الإيرادات.
- تمثل السلع B 30٪ من السلع، لكنها تساهم بنسبة 25٪ من الإيرادات.
- تمثل السلع C 50٪ من المقالات السلع لكنها تساهم بنسبة 5٪ من الإيرادات.

ضعفي اعتبارك أن هذه الأرقام تقريبية وستختلف بشكل كبير اعتمادًا على التوزيع الفعلي للمبيعات. الخلاصة الرئيسية هي أن العناصر A تشكل نسبة صغيرة من المخزون ولكنها تساهم أكثرفي الدخل، بينما تشكل العناصر C نسبةكبيرة من المخزون ولكنها تساهم بشكل أقلفي الدخل والعناصر B موجودةفي مكان ما حول الإجازاتفي المنتصف.

أهمية تحليل ABC

تعتمد استراتيجيات تخطيط المخزون والتخزين للمؤسسة على تحليل ABC لاتخاذ أي قرارات رئيسية. على سبيل المثال، يريد مدير المستودعات عادةً العناصر A الأقرب إلى أرصفة الشحن لتقليل الوقت الذي يستغرقه استلامها. هذا يزيد الإنتاجية ويقلل من تكاليف العمالة.

تحليل ABC مع تعلم الآلة

تأتي البيانات المستخدمةفي هذا المشروع من مجموعة بيانات معروفة لمتاجر التجزئة على الإنترنت. تتضمن مجموعة البيانات مبيعات الملابس عبر الإنترنت فقط طوال فصل الصيف. الأهم من ذلك، أنه يوضح عدد الوحدات المباعة والسعر المباع، مما يؤدي إلى تحقيق الإيرادات لكل عنصر. يمكن تنزيل مجموعة البيانات بسهولة من هنا.

الهدف من هذا المشروع هو فرز جميع عناصر مجموعة البياناتفي تصنيف ABC بناءً على أهميتها. عند عرض النتائج، يجب أن يكون هناك عدد قليل نسبيًا من العناصر A التي تولد غالبية الدخل وعددًا كبيرًا من العناصر C التي لا تولد الكثير من الدخل.

تحضير البيانات

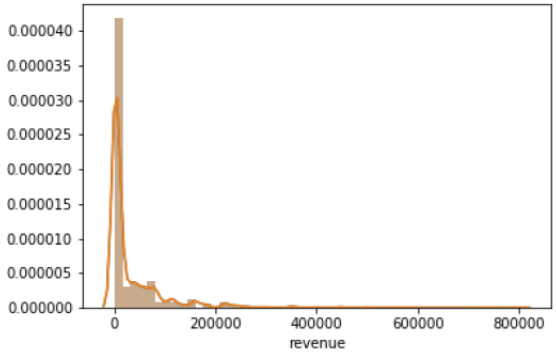
الآن، لنبدأ بهذه المهمةفي إعداد البيانات. سأبدأ هذا باستيراد الحزم الضرورية وقراءة مجموعة البيانات:

```
# Import libraries
import pandas as pd
import numpy as np
# read the data to a dataframe
df = pd.read_csv("Summer_Sales.csv")
```

سأضيف عمودًا جديدًا إلى بيانات الإيرادات عن طريق ضرب عدد الوحدات المباعةفي السعر. من الممكن أن يكون السعر قد تغير بمرور الوقت، خاصةً عند حدوث مبيعات فلاش، ولكن بدون بيانات إضافية لتحليلها، يُفترض أن جميع العناصر المباعة بسعر واحد ثابت:

```
df["revenue"] = df["units_sold"] * df["price"]
```

الآن، دعنا نتخيل الإيرادات باستخدام حزمة seabornفي python:



يوضح الرسم البياني أعلاه توزيع باريتو الموجودفي البيانات. تنتج الغالبية العظمى من المقالات أقل من 200000 يوروفي المبيعات.في الوقت نفسه، يُظهر أن بعض العناصر يتم بيعها بمبلغ يتراوح بين 400000 و800000 يورو، وهو ما يساهمفي غالبية الإيرادات.

الآن، سأقوم بتعريف دالة لتصنيف مقدار الدخل الناتج عن عنصر ما إلى صناديق، وبعد ذلك سأقوم بتطبيقه على البيانات:

```
def bins(x):
    for bar in range(20000, 820000, 20000):
        if x &lt;= bar:
            return bar
# Create new column to apply the bin function
df["rev_dist"] = df["revenue"].apply(lambda x: bins(x))
```

سأقوم الآن بإنشاء جدول محوري لسرد عدد العناصر التي تقعفي كل فئة:

```
df["count"] = 1
# Create a pivot table of the revenue distributions
```

```
pivot_table = pd.pivot_table(df, index = ["rev_dist"], values =
["count"], aggfunc = np.sum)
```

تطبيق خوارزمية التعلم الآلي

لتدريب النموذج بشكل صحيح، لا يكفي مجرد إلقاء نظرة على الدخل الناتج عن كل عنصر. يجب أن يعرف أيضًا كيف يتم توزيع الدخل. يوفر هذا الجدول المحوري مجموعة بيانات يمكن إدارتها للغاية بحيث يمكن للنموذج التدرب عليها. سأستخدم خوارزمية K-Means Clustering لهذه المهمة لتحليل ABC:

```
# import model from SKLearn
from sklearn.cluster import KMeans
# K -clusters is equal to 3 because things will be sorted into A, B,
and C
kmeans = KMeans(n_clusters=3)
kmeans.fit(pivot_table)
```

سأضيف الآن عمودًا جديدًا إلى الجدول المحوري يعطي تصنيف النموذج. وتجدر الإشارة إلى أنه افتراضيًا، ستعمل خوارزمية K-Means الخاصة بـ scikit-Learn على ترتيب العناصر على مقياس رقمي بدلاً من المقياس الأبجدي المستخدمفي تحليل ABC. لذلك، سيتم تصنيف كل صف على أنه صفر أو واحد أو اثنان:

```
pivot_table["category"] = kmeans.labels_
```

الآن، سأحدد قاموسًا جديدًا لتصنيف كل صف لمهمة تحليل ABC:

```
ABC_dict = {
    0: "A",
    1: "C",
    2: "B"
}
pivot table["ABC"] = pivot table["category"].apply(lambda x:
ABC_dict[x])
```

الآن، تذكر أن النموذج تم تدريبه على طاولة محورية. لم يتم حتى الآن تخصيص تصنيف ABC للعناصر. بدلاً من ذلك، تم تخصيص تصنيف دخل لها:

```
df = pd.merge(df, pivot_table, on = "rev_dist", how ="left")
```

هذا يعني أنه على الرغم من أننا لا نعرف على الفور العناصر التي تندرج ضمن الفئة A، فإننا نعلم أن بعض تصنيفات الدخل مصنفة كعناصر A. نتيجة لذلك، يمكننا فقط دمج إطار البيانات الرئيسي وجدول PivotTable لمنح كل عنصر تصنيف ABC الخاص به.

عند تحليل التوزيع النهائي للعناصر وجد أن:

- تمثل العناصر A 11.4٪ من السلع، ولكن 61.7٪ من حجم الأعمال.
- تمثل العناصر B 20.5٪ من العناصر، ولكن 30.7٪ من حجم الأعمال.
- تمثل السلع C 68.1٪ من السلع، ولكن 7.6٪ من حجم الأعمال.

35) توقع أمراض الكلى المزمنة باستخدام التعلم الآلي Predicting Chronic Kidney Disease using Machine Learning

في هذه المقالة، سنستعرض مجموعة بيانات أمراض الكلى المزمنة ونجري التحليل الكامل لنفس الهدف الرئيسي هو التنبؤ بما إذا كان الفرد سيصاب بمرض كلوي مزمن أم لا بناءً على البيانات المقدمة.

استيراد المكتبات اللازمة

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV, train_test_split,
cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
import scipy.stats as stats
import seaborn as sns

%matplotlib inline
```

الآن، دعونا نقرأ مجموعة البيانات الخاصة بنا:

```
df = pd.read_csv('kidney.csv')
data = df
data.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wbcc	rbcc	htn	dm	cad	appet	pe	ane	class
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44.0	7800.0	5.2	yes	yes	no	good	no	no	ckd
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38.0	6000.0	NaN	no	no	no	good	no	no	ckd
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31.0	7500.0	NaN	no	yes	no	poor	no	yes	ckd
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32.0	6700.0	3.9	yes	no	no	poor	yes	yes	ckd
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35.0	7300.0	4.6	no	no	no	good	no	no	ckd

الآن، دعونا نرى شكل مجموعة البيانات الخاصة بنا:

```
data.shape
```

```
(400, 25)
```

معالجة البيانات

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   age     391 non-null    float64
 1   bp      388 non-null    float64
 2   sg      353 non-null    float64
 3   al      354 non-null    float64
 4   su      351 non-null    float64
 5   rbc     248 non-null    object 
 6   pc      335 non-null    object 
 7   pcc     396 non-null    object 
 8   ba      396 non-null    object 
 9   bgr     356 non-null    float64
 10  bu      381 non-null    float64
 11  sc      383 non-null    float64
 12  sod     313 non-null    float64
 13  pot     312 non-null    float64
 14  hemo    348 non-null    float64
 15  pcv     329 non-null    float64
 16  wbcc    294 non-null    float64
 17  rbcc    269 non-null    float64
 18  htn     398 non-null    object 
 19  dm      398 non-null    object 
 20  cad     398 non-null    object 
 21  appet   399 non-null    object 
 22  pe      399 non-null    object 
 23  ane     399 non-null    object 
 24  class   400 non-null    object 
dtypes: float64(14), object(11)
memory usage: 78.2+ KB
```

الآن، دعونا نرى بياناتنا إحصائيًا:

```
df.describe()
```

	age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo	pcv
count	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000	312.000000	348.000000	329.000000
mean	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.627244	12.526437	38.884498
std	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752	3.193904	2.912587	8.990105
min	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000	2.500000	3.100000	9.000000
25%	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000	3.800000	10.300000	32.000000
50%	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000	4.400000	12.650000	40.000000
75%	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000	4.900000	15.000000	45.000000
max	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000	47.000000	17.800000	54.000000

الآن، دعنا نرى العدد الإجمالي للقيم الخالية التي تحملها كل ميزة:

```
age       9
bp       12
sg       47
al       46
su       49
rbc     152
pc       65
pcc       4
ba        4
bgr      44
bu       19
sc       17
sod      87
pot      88
hemo     52
pcv      71
wbcc    106
rbcc    131
htn       2
dm        2
cad       2
appet     1
pe        1
ane       1
class     0
dtype: int64
```

مصفوفة الارتباط ورسم المصفوفة

```
df.corr()
```

	age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo	pcv	wbcc	rbcc
age	1.000000	0.159480	-0.191096	0.122091	0.220866	0.244992	0.196985	0.132531	-0.100046	0.058377	-0.192928	-0.242119	0.118339	-0.268896
bp	0.159480	1.000000	-0.218836	0.160689	0.222576	0.160193	0.188517	0.146222	-0.116422	0.075151	-0.306540	-0.326319	0.029753	-0.261936
sg	-0.191096	-0.218836	1.000000	-0.469760	-0.296234	-0.374710	-0.314295	-0.361473	0.412190	-0.072787	0.602582	0.603560	-0.236215	0.579476
al	0.122091	0.160689	-0.469760	1.000000	0.269305	0.379464	0.453528	0.399198	-0.459896	0.129038	-0.634632	-0.611891	0.231989	-0.566437
su	0.220866	0.222576	-0.296234	0.269305	1.000000	0.717827	0.168583	0.223244	-0.131776	0.219450	-0.224775	-0.239189	0.184893	-0.237448
bgr	0.244992	0.160193	-0.374710	0.379464	0.717827	1.000000	0.143322	0.114875	-0.267848	0.066966	-0.306189	-0.301385	0.150015	-0.281541
bu	0.196985	0.188517	-0.314295	0.453528	0.168583	0.143322	1.000000	0.586368	-0.323054	0.357049	-0.610360	-0.607621	0.050462	-0.579087
sc	0.132531	0.146222	-0.361473	0.399198	0.223244	0.114875	0.586368	1.000000	-0.690158	0.326107	-0.401670	-0.404193	-0.006390	-0.400852
sod	-0.100046	-0.116422	0.412190	-0.459896	-0.131776	-0.267848	-0.323054	-0.690158	1.000000	0.097887	0.365183	0.376914	0.007277	0.344873
pot	0.058377	0.075151	-0.072787	0.129038	0.219450	0.066966	0.357049	0.326107	0.097887	1.000000	-0.133746	-0.163182	-0.105576	-0.158309
hemo	-0.192928	-0.306540	0.602582	-0.634632	-0.224775	-0.306189	-0.610360	-0.401670	0.365183	-0.133746	1.000000	0.895382	-0.169413	0.798880
pcv	-0.242119	-0.326319	0.603560	-0.611891	-0.239189	-0.301385	-0.607621	-0.404193	0.376914	-0.163182	0.895382	1.000000	-0.197022	0.791625
wbcc	0.118339	0.029753	-0.236215	0.231989	0.184893	0.150015	0.050462	-0.006390	0.007277	-0.105576	-0.169413	-0.197022	1.000000	-0.158163
rbcc	-0.268896	-0.261936	0.579476	-0.566437	-0.237448	-0.281541	-0.579087	-0.400852	0.344873	-0.158309	0.798880	0.791625	-0.158163	1.000000

دعنا نتعرف على عدد كل فئة:

```
df['class'].value_counts()
```

```
ckd        250

notckd     150

Name: class, dtype: int64
```

الاستدلال: من الناتج أدناه يمكننا استنتاج أن هذا قريب من "مجموعة بيانات غير متوازنة".

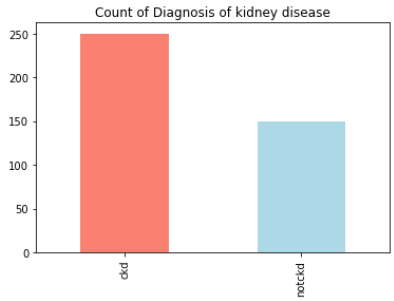
تمثيل متغير الهدف بالنسبة المئوية:

```
countNoDisease = len(df[df['class'] == 0])
countHaveDisease = len(df[df['class'] == 1])
print("Percentage of Patients Haven't Heart Disease:
{:.2f}%".format((countNoDisease / (len(df['class']))*100)))
print("Percentage of Patients Have Heart Disease:
{:.2f}%".format((countHaveDisease / (len(df['class']))*100)))
```

```
Percentage of Patients Haven't Heart Disease: 0.00%

Percentage of Patients Have Heart Disease: 0.00%
```

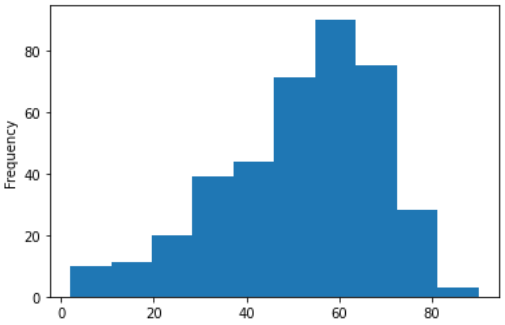
فهم موازنة البيانات بصريا:

```
df['class'].value_counts().plot(kind='bar',color=['salmon','lightblue'
],title="Count of Diagnosis of kidney disease")
```



هنا سوف نتحقق من توزيع عمود العمر age column:

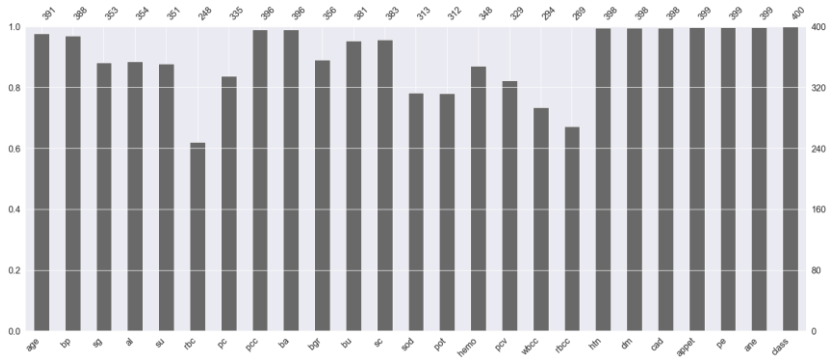
```
df['age'].plot(kind='hist')
```



الاستدلال: هنا بمساعدة المدرج التكراري يمكننا أن نرى أن الأشخاص الذين تتراوح أعمارهم بين 50 و60 عامًا منتشرون على نطاق واسعفي مجموعة البيانات هذه.

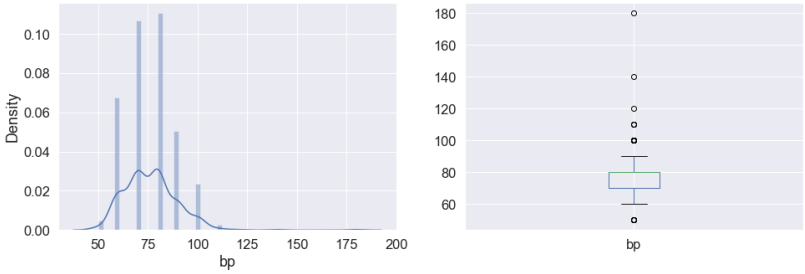
نحن هنا نرسم الرسم البياني لرؤية القيم الخاليةفي مجموعة البيانات.

```
p = msno.bar(data)
```



الاستدلال: هنا أي ميزات لم تمس علامة 400في الأعلى تحتوي على قيم فارغة.

```
plt.subplot(121), sns.distplot(data['bp'])
plt.subplot(122), data['bp'].plot.box(figsize=(16,5))
plt.show()
```



الاستدلال: هنافي الرسم البياني أعلاه يمكننا أن نرى توزيع ضغط الدم bp وأيضًافي الرسم البياني الفرعي، من الواضح أن عمود bp يحتوي على بعض القيم المتطرفة فيه.

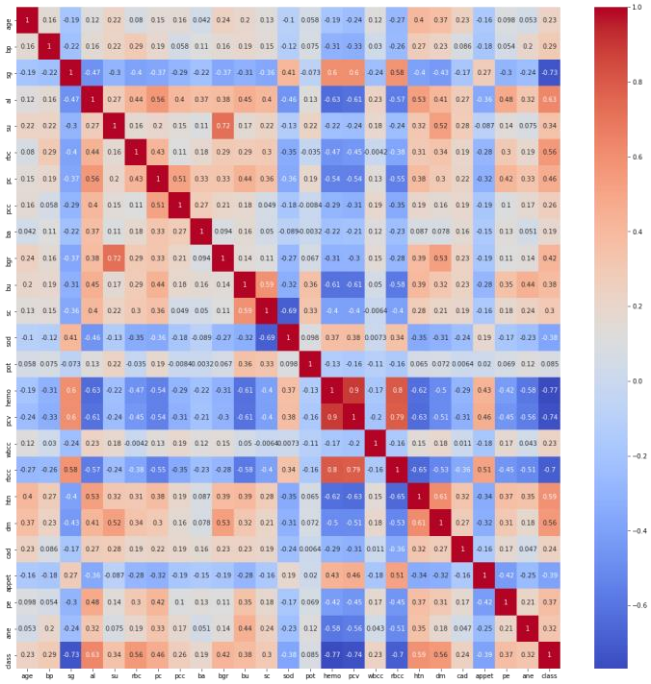
الآن سنقوم بتحويل القيم الفئوية (الكائن) إلى القيم الفئوية (int).

```
data['class'] = data['class'].map({'ckd':1,'notckd':0})
data['htn'] = data['htn'].map({'yes':1,'no':0})
data['dm'] = data['dm'].map({'yes':1,'no':0})
data['cad'] = data['cad'].map({'yes':1,'no':0})
data['appet'] = data['appet'].map({'good':1,'poor':0})
data['ane'] = data['ane'].map({'yes':1,'no':0})
data['pe'] = data['pe'].map({'yes':1,'no':0})
data['ba'] = data['ba'].map({'present':1,'notpresent':0})
data['pcc'] = data['pcc'].map({'present':1,'notpresent':0})
data['pc'] = data['pc'].map({'abnormal':1,'normal':0})
data['rbc'] = data['rbc'].map({'abnormal':1,'normal':0})
data['class'].value_counts()
```

```
1    250

0    150

Name: class, dtype: int64
```

إيجاد علاقة الارتباط بين المخططات:

```
plt.figure(figsize = (19,19))
sns.heatmap(data.corr(), annot = True, cmap = 'coolwarm') # looking
for strong correlations with "class" row
```



تحليل البيانات الاستكشافية (EDA)

دعونا نرى شكل مجموعة البيانات مرة أخرى بعد التحليل:

```
data.shape
```

```
(400, 25)
```

الآن دعونا نرى الأعمدة النهائية الموجودةفي مجموعة البيانات الخاصة بنا:

```
data.columns
```

```
Index(['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
       'sc', 'sod', 'pot', 'hemo', 'pcv', 'wbcc', 'rbcc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane', 'class'],
      dtype='object')
```

الآن اسقاط القيم الخالية:

```
data.shape[0], data.dropna().shape[0]
```

```
(400, 158)
```

هنا من الناتج أعلاه، يمكننا أن نرى أن هناك 158 قيمة فارغةفي مجموعة البيانات. الآن يتبقى لنا خياران يمكننا إما إسقاط جميع القيم الفارغة أو الاحتفاظ بها عندما نتخلى عن قيم NA لذلك يجب أن نفهم أن مجموعة البيانات الخاصة بنا ليست كبيرة جدًا وإذا أسقطنا هذه القيم الفارغة فسيكون ذلك متساويًا أصغرفي هذه الحالة إذا قدمنا بيانات أقل جدًا إلى نموذج التعلم الآلي الخاص بنا، فسيكون الأداء أقل أيضًا ولا نعرف حتى الآن أن هذه القيم الفارغة مرتبطة ببعض الميزات الأخرىفي مجموعة البيانات.

لذلكفي هذا الوقت، سأحتفظ بهذه القيم وأرى كيف سيعمل النموذجفي مجموعة البيانات هذه.

أيضًا عندما نعمل على بعض مشاريع الرعاية الصحية حيث سنتوقع ما إذا كان الشخص يعاني من هذا المرض أم لا، هناك شيء واحد يجب أن نضعهفي ذهني وهو أن تقييم النموذج يجب أن يحتوي على أقل أخطاء إيجابية كاذبة.

```
data.dropna(inplace=True)
data.shape
```

```
(158, 25)
```

بناء النموذج

1) الانحدار اللوجستي

```
from sklearn.linear_model import LogisticRegression
```

```
logreg = LogisticRegression()

X = data.iloc[:,:-1]
y = data['class']

X_train, X_test, y_train, y_test = train_test_split(X,y, stratify = y,
shuffle = True)

logreg.fit(X_train,y_train)
```

```
LogisticRegression()
```

دقة التدريب:

```
logreg.score(X_train,y_train)
```

```
1.0
```

دقة الاختبار:

```
logreg.score(X_test,y_test)
```

```
0.975
```

طباعة دقة التدريب والاختبار:

```
from sklearn.metrics import accuracy_score, confusion_matrix

print('Train Accuracy: ', accuracy_score(y_train, train_pred))
print('Test Accuracy: ', accuracy_score(y_test, test_pred))
```

```
Train Accuracy:  1.0
Test Accuracy:  0.975
```

تُظهر الخلية أدناه معاملات كل متغير.

(مثال على قراءة المعامِلات من الانحدار اللوجستي: زيادة وحدة واحدةفي العمر تجعل الفرد حوالي e ^ 0.14 مرة أكثر عرضة للإصابة بمرض الكلى المزمن،في حين أن زيادة وحدة واحدة في ضغط الدم تجعل الفرد حوالي e ^ -0.07 مرة من المحتمل أن يكون مصابًا بمرض الكلى المزمن.

```
pd.DataFrame(logreg.coef_, columns=X.columns)
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	hemo	pcv	wbcc	rbcc
0	0.28582	-0.132118	0.002671	0.309953	0.010789	0.019856	0.091069	0.003106	0.006829	0.414045	...	-0.282868	-0.62111	0.001157	-0.141783

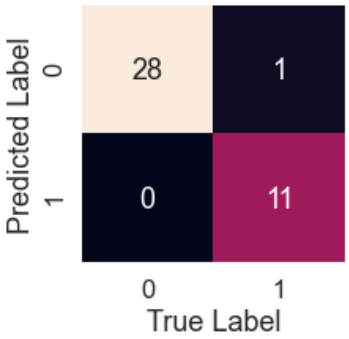
1 rows × 24 columns

مصفوفة الارتباك

```
sns.set(font_scale=1.5)
```

```
def plot_conf_mat(y_test,y_preds):
    """
    This function will be heloing in plotting the confusion matrix by
using seaborn
    """

    fig,ax=plt.subplots(figsize=(3,3))

ax=sns.heatmap(confusion_matrix(y_test,y_preds),annot=True,cbar=False)
    plt.xlabel("True Label")
    plt.ylabel("Predicted Label")
```

```
log_pred = logreg.predict(X_test)
plot_conf_mat(y_test, log_pred)
```




```
tn, fp, fn, tp = confusion_matrix(y_test, test_pred).ravel()

print(f'True Neg: {tn}')
print(f'False Pos: {fp}')
print(f'False Neg: {fn}')
print(f'True Pos: {tp}')
```

```
True Neg: 28
False Pos: 1
False Neg: 0
True Pos: 11
```

2) مصنف KNN

إنها ممارسة جيدة أولاً لتحقيق التوازن بين الفصل جيداً قبل استخدام KNN، حيث نعلم أنهفي حالة الفئات غير المتوازنة، لا يؤدي KNN أداءً جيداً.

```
df["class"].value_counts()
```

```
0    115

1     43

Name: class, dtype: int64
```

الآن دعونا نربط متغيرات الفئة لدينا معًا.

```
balanced_df = pd.concat([df[df["class"] == 0], df[df["class"] ==
1].sample(n = 115, replace = True)], axis = 0)
balanced_df.reset_index(drop=True, inplace=True)
balanced_df["class"].value_counts()
```

```
1    115

0    115

Name: class, dtype: int64
```

فلنحجم البيانات الآن:

```
ss = StandardScaler()
ss.fit(X_train)
X_train = ss.transform(X_train)
X_test = ss.transform(X_test)
```

الآن سنقوم بضبط نموذج KNN للحصول على دقة أفضل:

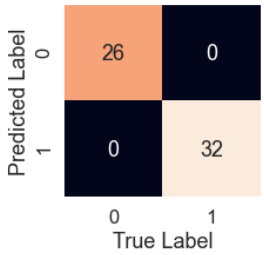
```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier()

params = {
    "n_neighbors":[3,5,7,9],
    "weights":["uniform","distance"],
    "algorithm":["ball_tree","kd_tree","brute"],
    "leaf_size":[25,30,35],
    "p":[1,2]
}

gs = GridSearchCV(knn, param_grid=params)
model = gs.fit(X_train,y_train)
preds = model.predict(X_test)
accuracy_score(y_test, preds)
```

```
1.0
```

مصفوفة الارتباك لنموذج KNN:

```
knn_pred = model.predict(X_test)
plot_conf_mat(y_test, knn_pred)
```




```
tn, fp, fn, tp = confusion_matrix(y_test, preds).ravel()
```

```
print(f'True Neg: {tn}')
print(f'False Pos: {fp}')
print(f'False Neg: {fn}')
print(f'True Pos: {tp}')
```

```
True Neg: 26
False Pos: 0
False Neg: 0
True Pos: 32
```

أهمية الميزة

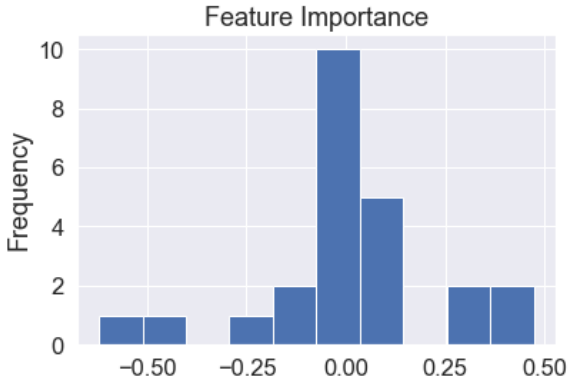
```
feature_dict=dict(zip(df.columns,list(logreg.coef_[0])))
feature_dict
```

هنا سوف نحصل على المعامل من الميزات التي ستخبرنا بوزن كل ميزة.

```
{'age': 0.2858203378727209,
 'bp': -0.13211767022170745,
 'sg': 0.0026714534270341444,
 'al': 0.3099528545093234,
 'su': 0.010788785962584712,
 'rbc': 0.01985635073828642,
 'pc': 0.09106895589320387,
 'pcc': 0.003106393240262293,
 'ba': 0.006828878616469952,
 'bgr': 0.4140451203997997,
 'bu': 0.47262371944289844,
 'sc': 0.12893875993072498,
 'sod': -0.4419699201228987,
 'pot': 0.05909714695858163,
 'hemo': -0.28286805186344094,
 'pcv': -0.6211104727832718,
 'wbcc': 0.001157338688486265,
 'rbcc': -0.1417833283935927,
 'htn': 0.08881269207443204,
 'dm': 0.08689401433102413,
 'cad': 0.0018059681932075433,
 'appet': -0.004481530609769657,
 'pe': 0.0051126943850270425,
 'ane': 0.00349950552414094}
```

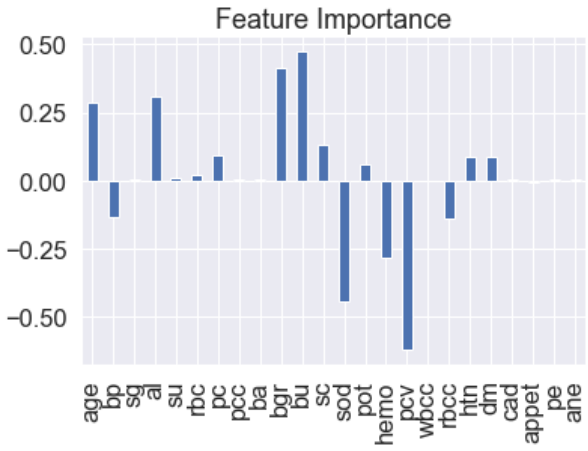
رسم أهمية الميزة:

```
feature_df=pd.DataFrame(feature_dict,index=[0])
feature_df.T.plot(kind="hist",legend=False,title="Feature Importance")
```



رسم أهمية الميزة ـ تبديل:

```
feature_df=pd.DataFrame(feature_dict,index=[0])
feature_df.T.plot(kind="bar",legend=False,title="Feature Importance")
```



حفظ النموذج

```
import pickle

# Now with the help of pickle model we will be saving the trained
model
saved_model = pickle.dumps(logreg)

# Load the pickled model
logreg_from_pickle = pickle.loads(saved_model)

# Now here we will load the model
logreg_from_pickle.predict(X_test)
```

```
array([1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1], dtype=int64)
```

الكود المصدري للمشروع من هنا.

36) تصنيف الفواكه باستخدام التعلم الالي Fruits Classification using Machine Learning

في هذا المنشور، سننفذ العديد من خوارزميات التعلم الآليفي Python باستخدام Scikit-Learn، أداة التعلم الآلي الأكثر شيوعًافي Python. استخدام مجموعة بيانات بسيطة لمهمة تدريب المصنف للتمييز بين أنواع الفاكهة المختلفة.

الغرض من هذا المنشور هو تحديد خوارزمية التعلم الآلي الأنسب للمشكلة المطروحة؛ وبالتالي، نريد مقارنة الخوارزميات المختلفة واختيار أفضلها أداءً. هيا بنا نبدأ!

البيانات

تم إنشاء مجموعة بيانات الفواكه بواسطة الدكتور إيان موراي من جامعة إدنبرة. اشترى بضع عشرات من البرتقال والليمون والتفاح من أصناف مختلفة، وسجل قياساتهافي جدول. ثم قام الأساتذةفي جامعة ميشيغان بتنسيق بيانات الفاكهة بشكل طفيف ويمكن تنزيلها من هنا.

دعونا نلقي نظرة على الصفوف القليلة الأولى من البيانات.

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as pltfruits =
pd.read_table('fruit_data_with_colors.txt')
fruits.head()
```

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79

يمثل كل صف من مجموعة البيانات قطعة واحدة من الفاكهة كما هو موضحفي العديد من الميزات الموجودةفي أعمدة الجدول.

لدينا 59 قطعة من الفاكهة و7 ميزاتفي مجموعة البيانات:

```
print(fruits.shape)
```

```
(59, 7)
```

لدينا أربعة أنواع من الفاكهةفي مجموعة البيانات:

```
print(fruits['fruit_name'].unique())
```

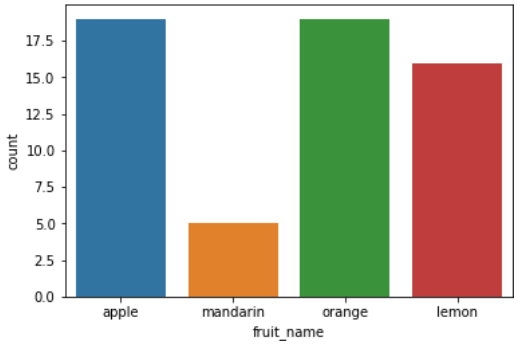
```
['apple' 'mandarin' 'orange' 'lemon']
```

البيانات متوازنة جداً باستثناء اليوسفي mandarin. علينا فقط أن نذهب معها.

```
print(fruits.groupby('fruit_name').size())
```

```
fruit_name
apple       19
lemon       16
mandarin     5
orange      19
dtype: int64
```

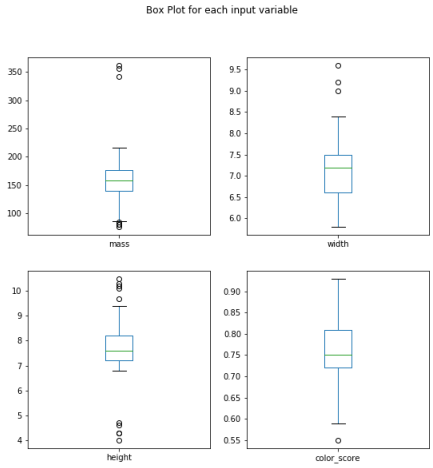
```
import seaborn as sns
sns.countplot(fruits['fruit_name'],label="Count")
plt.show()
```



الرسم البياني

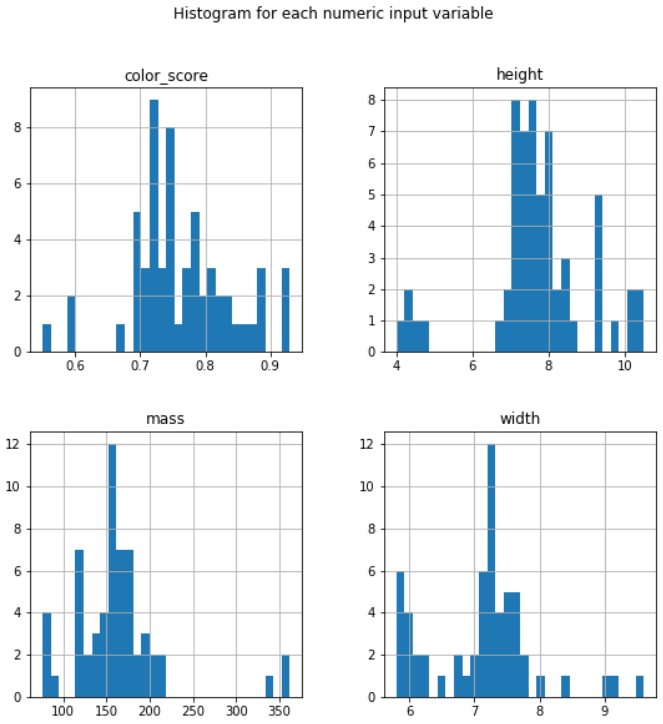
- المخطط الصندوقي لكل متغير رقمي سيعطينا فكرة أوضح عن توزيع متغيرات الإدخال:

```
fruits.drop('fruit_label', axis=1).plot(kind='box', subplots=True,
layout=(2,2), sharex=False, sharey=False, figsize=(9,9),
title='Box Plot for each input variable')
plt.savefig('fruits_box')
plt.show()
```



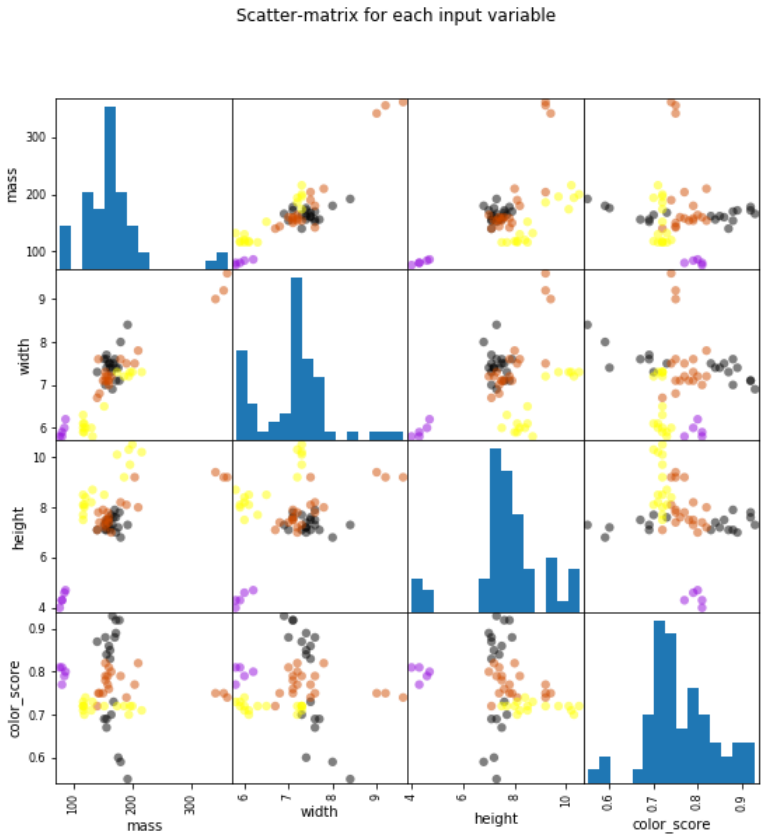
- يبدو أن درجة اللون ربما لها توزيع غاوسي قريب.

```
import pylab as pl
fruits.drop('fruit_label' ,axis=1).hist(bins=30, figsize=(9,9))
pl.suptitle("Histogram for each numeric input variable")
plt.savefig('fruits_hist')
plt.show()
```



- ترتبط بعض أزواج السمات (الكتلة والعرض). هذا يشير إلى وجود ارتباط كبير وعلاقة يمكن التنبؤ بها.

```
from pandas.tools.plotting import scatter matrix
from matplotlib import cmfeature_names = ['mass', 'width', 'height',
'color_score']
X = fruits[feature_names]
y = fruits['fruit_label']cmap = cm.get_cmap('gnuplot')
scatter = pd.scatter_matrix(X, c = y, marker = 'o', s=40,
hist kwds={'bins':15}, figsize=(9,9), cmap = cmap)
plt.suptitle('Scatter-matrix for each input variable')
plt.savefig('fruits_scatter_matrix')
```



الملخص الإحصائي

	fruit_label	mass	width	height	color_score
count	59.000000	59.000000	59.000000	59.000000	59.000000
mean	2.542373	163.118644	7.105085	7.693220	0.762881
std	1.208048	55.018832	0.816938	1.361017	0.076857
min	1.000000	76.000000	5.800000	4.000000	0.550000
25%	1.000000	140.000000	6.600000	7.200000	0.720000
50%	3.000000	158.000000	7.200000	7.600000	0.750000
75%	4.000000	177.000000	7.500000	8.200000	0.810000
max	4.000000	362.000000	9.600000	10.500000	0.930000

يمكننا أن نرى أن القيم العددية ليس لها نفس المقياس. سنحتاج إلى تطبيق التحجيم scaling على مجموعة الاختبار التي قمنا بحسابها لمجموعة التدريب.

إنشاء مجموعات التدريب والاختبار وتطبيق القياس

```
from sklearn.model_selection import train_test_splitX_train, X_test,
y_train, y_test = train_test_split(X, y, random_state=0)from
sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

بناء النماذج

الانحدار اللوجستي

```
from sklearn.linear_model import LogisticRegressionlogreg =
LogisticRegression()
logreg.fit(X_train, y_train)print('Accuracy of Logistic regression
classifier on training set: {:.2f}'
     .format(logreg.score(X_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'
     .format(logreg.score(X_test, y_test)))
```

```
Accuracy of Logistic regression classifier on training set: 0.70
Accuracy of Logistic regression classifier on test set: 0.40
```

شجرة القرار

```
from sklearn.tree import DecisionTreeClassifierclf =
DecisionTreeClassifier().fit(X_train, y_train)print('Accuracy of
Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.73
```

KNN

```
from sklearn.neighbors import KNeighborsClassifierknn =
KNeighborsClassifier()
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.95
Accuracy of K-NN classifier on test set: 1.00
```

LDA

```
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysislda = LinearDiscriminantAnalysis()
lda.fit(X train, y train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(X_test, y_test)))
```

```
Accuracy of LDA classifier on training set: 0.86
Accuracy of LDA classifier on test set: 0.67
```

GNB

```
from sklearn.naive_bayes import GaussianNBgnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'
     .format(gnb.score(X_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'
     .format(gnb.score(X_test, y_test)))
```

```
Accuracy of GNB classifier on training set: 0.86
Accuracy of GNB classifier on test set: 0.67
```

SVM

```
from sklearn.svm import SVCsvm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(X train, y train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
     .format(svm.score(X_test, y_test)))
```

```
Accuracy of SVM classifier on training set: 0.61
Accuracy of SVM classifier on test set: 0.33
```

كانت خوارزمية KNN هي النموذج الأكثر دقة الذي جربناه. توفر مصفوفة الارتباك إشارة إلى عدم حدوث خطأفي مجموعة الاختبار. ومع ذلك، كانت مجموعة الاختبار صغيرة جداً.

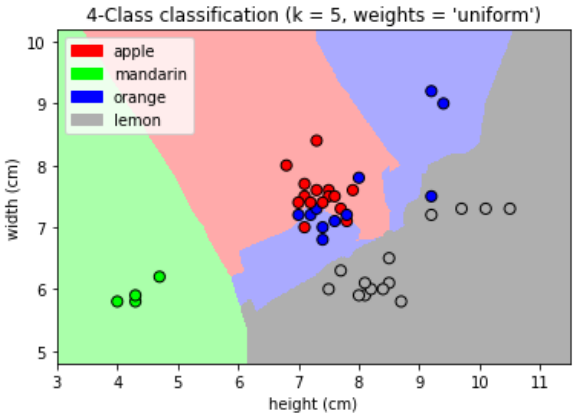
```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
pred = knn.predict(X_test)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

```
[[4 0 0 0]
 [0 1 0 0]
 [0 0 8 0]
 [0 0 0 2]]
             precision    recall  f1-score   support

          1       1.00      1.00      1.00         4
          2       1.00      1.00      1.00         1
          3       1.00      1.00      1.00         8
          4       1.00      1.00      1.00         2

avg / total       1.00      1.00      1.00        15
```

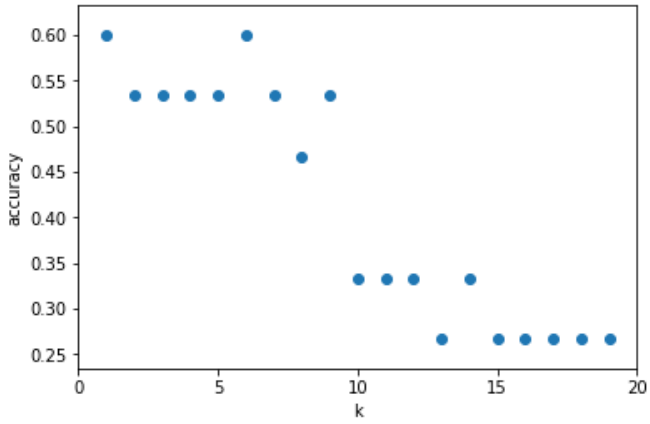
رسم حدود القرار لمصنف KNN

```
import matplotlib.cm as cm
from matplotlib.colors import ListedColormap, BoundaryNorm
import matplotlib.patches as mpatches
import matplotlib.patches as mpatchesX = fruits[['mass', 'width',
'height', 'color_score']]
y = fruits['fruit_label']
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)def plot_fruit_knn(X, y, n_neighbors, weights):
    X_mat = X[['height', 'width']].as_matrix()
    y_mat = y.as_matrix()# Create color maps
    cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA',
'#AAAAFF','#AFAFAF'])
    cmap_bold  = ListedColormap(['#FF0000', '#00FF00',
'#0000FF','#AFAFAF'])clf = neighbors.KNeighborsClassifier(n neighbors,
weights=weights)
```

```
    clf.fit(X_mat, y_mat)# Plot the decision boundary by assigning a
color in the color map
    # to each mesh point.

    mesh_step_size = .01  # step size in the mesh
    plot_symbol_size = 50

    x_min, x_max = X_mat[:, 0].min() - 1, X_mat[:, 0].max() + 1
    y_min, y_max = X_mat[:, 1].min() - 1, X_mat[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, mesh_step_size),
                         np.arange(y_min, y_max, mesh_step_size))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])# Put the result
into a color plot
    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)# Plot training points
    plt.scatter(X_mat[:, 0], X_mat[:, 1], s=plot_symbol_size, c=y,
cmap=cmap_bold, edgecolor = 'black')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())patch0 =
mpatches.Patch(color='#FF0000', label='apple')
    patch1 = mpatches.Patch(color='#00FF00', label='mandarin')
    patch2 = mpatches.Patch(color='#0000FF', label='orange')
    patch3 = mpatches.Patch(color='#AFAFAF', label='lemon')
    plt.legend(handles=[patch0, patch1, patch2,
patch3])plt.xlabel('height (cm)')
plt.ylabel('width (cm)')
plt.title("4-Class classification (k = %i, weights = '%s')"
          % (n_neighbors, weights))
plt.show()plot_fruit_knn(X_train, y_train, 5, 'uniform')
```




```
k_range = range(1, 20)
scores = []for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    scores.append(knn.score(X_test, y_test))
plt.figure()
plt.xlabel('k')
plt.ylabel('accuracy')
plt.scatter(k_range, scores)
plt.xticks([0,5,10,15,20])
```



بالنسبة لمجموعة البيانات المحددة هذه، نحصل على أعلى دقة عندما تكون k = 5.

الملخص

في هذا المنشور، ركزنا على دقة التنبؤ. هدفنا هو تعلم نموذج له أداء تعميم جيد. مثل هذا النموذج يزيد من دقة التنبؤ. حددنا خوارزمية التعلم الآلي الأنسب للمشكلة المطروحة (أي تصنيف أنواع الفاكهة)؛ لذلك، قمنا بمقارنة الخوارزميات المختلفة واختيار أفضل الخوارزميات أداءً.

يمكن العثور على رمز المصدر الذي أنشأ هذا المنشور هنا.

37) تصنيف النص باستخدام التعلم الآلي Text Classification using Machine learning

مكان واحدفي علم البيانات Data Science حيث غالبًا ما يستخدم multinomial naive Bayesفي تصنيف النص text classification، حيث ترتبط الميزات بعدد الكلمات أو التكرار داخل المستندات المراد تصنيفها.

في مشروع علم البيانات هذا، سنستخدم ميزات عدد الكلمات المتناثر sparse word count features من مجموعة مجموعات الأخبار العشرين لإظهار كيف يمكننا تصنيف هذه المستندات القصيرة إلى فئات.

لنقم بتنزيل البيانات وإلقاء نظرة على أسماء الأهداف:

```
from sklearn.datasets import fetch_20newsgroups
data = fetch_20newsgroups()
print(data.target_names)
```

```
#Output-

['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware',
'comp.sys.mac.hardware', 'comp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles',
'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.space',
'soc.religion.christian', 'talk.politics.guns', 'talk.politics.mideast', 'talk.politics.misc',
'talk.religion.misc']
```

للتبسيط، سنختار عددًا قليلاً فقط من هذه الفئات، وننزل مجموعة التدريب والاختبار:

```
categories = ['talk.religion.misc', 'soc.religion.christian',
'sci.space', 'comp.graphics']
train = fetch_20newsgroups(subset='train', categories=categories)
test = fetch_20newsgroups(subset='test', categories=categories)
```

هنا إدخال تمثيلي من البيانات:

```
print(train.data[5])
```

```
#Output-

From: dmcgee@uluhe.soest.hawaii.edu (Don McGee)
Subject: Federal Hearing
Originator: dmcgee@uluhe
Organization: School of Ocean and Earth Science and Technology
Distribution: usa
Lines: 10
```

Fact or rumor....? Madalyn Murray O'Hare an atheist who eliminated the use of the bible reading and prayer in public schools 15 years ago is now going to appear before the FCC with a petition to stop the reading of the Gospel on the airways of America. And she is also campaigning to remove Christmas programs, songs, etc from the public schools. If it is true then mail to Federal Communications Commission 1919 H Street Washington DC 20054 expressing your opposition to her request. Reference Petition number

2493.

من أجل استخدام هذه البيانات للتعلم الآلي، نحتاج إلى أن نكون قادرين على تحويل محتوى كل سلسلة إلى متجه من الأرقام. لهذا سنستخدم متجه TF – IDF، وننشئ خط أنابيب يربطه بمصنف Bayes السذج متعدد الحدود:

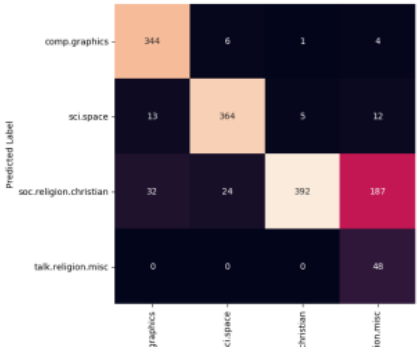
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline

model = make_pipeline(TfidfVectorizer(), MultinomialNB())
```

باستخدام خط الأنابيب هذا، يمكننا تطبيق النموذج على بيانات التدريب، والتنبؤ بتسميات بيانات الاختبار:

```
model.fit(train.data, train.target)
labels = model.predict(test.data)
```

الآن بعد أن توقعنا تسميات بيانات الاختبار، يمكننا تقييمها لمعرفة المزيد عن أداء المقدر. على سبيل المثال، إليك مصفوفة الارتباك confusion matrix بين التسميات الحقيقية والمتوقعة لبيانات الاختبار:

```
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(test.target, labels)
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(mat.T,square=True,annot=True,fmt='d',cbar=False,

xticklabels=train.target_names,yticklabels=train.target_names(
plt.xlabel('True Label')
plt.ylabel("Predicted Label")
plt.show()
```



من الواضح أنه حتى هذا المصنف البسيط جداً يمكنه فصل الحديث عن الفضاء عن حديث الكمبيوتر بنجاح، لكنه يختلط بين الحديث عن الدين والحديث عن المسيحية. ربما تكون هذه منطقة ارتباك متوقعة.

الشيء الرائع هنا هو أن لدينا الآن الأدوات لتحديد فئة أي سلسلة، باستخدام طريقة predict() لخط الأنابيب pipeline هذا. فيما يلي دالة الأداة السريعة التي ستعرض التنبؤ لسلسلة نصية واحدة:

```
def predict_category(s, train=train,model=model):
    pred = model.predict([s])
    print(train.target_names[pred[0]])
```

دعونا نجربها:

```
predict_category("sending a payload to the ISS")
```

```
sci.space
```

```
predict_category("discussing islam vs atheism")
```

```
soc.religion.christian
```

```
predict_category("determining the screen resolution")
```

```
comp.graphics
```

تذكر أن هذا ليس أكثر تعقيداً من نموذج احتمالي بسيط للتردد (المرجح) لكل كلمةفي السلسلة؛ ومع ذلك، فإن النتيجة مذهلة. حتى الخوارزمية الساذجة جداً، عند استخدامها بعناية وتدريبها على مجموعة كبيرة من البيانات عالية الأبعاد، يمكن أن تكون فعالة بشكل مدهش.

38) مشروع توقع فوز فريق IPL باستخدام التعلم الآلي IPL Team Win Prediction Project Using Machine Learning

مقدمة

يُعد التعلم الآلي وعلوم البيانات أحد أسرع المجالات التكنولوجية نموًا. ينتج عن هذا المجال تغييرات مذهلةفي المجال الطبي والإنتاج والروبوتات وما إلى ذلك. والسبب الرئيسي للتقدمفي هذا المجال هو زيادة القوة الحسابية وتوافر كميات كبيرة من البيانات.في علم البيانات، يتم تحليل هذه البيانات وجعلها مناسبة لإنشاء نماذج ومنتجات التعلم الآلي.

في مقال اليوم، سنناقش توقعات فوز فريق IPL. استنادًا إلى بعض إحصاءات المطابقة، نتوقع الفائزفي مباراة IPL. من خلال هذا المشروع، سوف تتعرف على تحليل البيانات الاستكشافية وتقنيات هندسة الميزات التي يجب تطبيقها لمعالجة البيانات.

فلننتقل إلى مجموعة البيانات.

مجموعة البيانات

مجموعة البيانات التي نستخدمها هنا هي مجموعة بيانات IPL، والتي تحتوي على التفاصيل المتعلقة بإحصائيات الفائز والمباراة. يحتوي على تفاصيل مثل الفرق التي تم لعبها، والفائز، ومكان المباراة، والفوز بعدد الويكيت والركض، وقرار القذف، سواء تم تطبيق DLS أم لا، وأسماء الحكام وما إلى ذلك. إجراء تحليل البيانات الاستكشافية وهندسة البيانات على هذه البيانات أمر بالغ الأهمية مهم.

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_match	venue	umpire1	umpire2
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj Singh	Rajiv Gandhi International Stadium, Uppal	AY Dandekar	NJ Llong
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	SPD Smith	Maharashtra Cricket Association Stadium	A Nand Kishore	S Ravi
2	3	IPL-2017	Rajkot	07-04-2017	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	CA Lynn	Saurashtra Cricket Association Stadium	Nitin Menon	CK Nandan
3	4	IPL-2017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	GJ Maxwell	Holkar Cricket Stadium	AK Chaudhary	C Shamshuddin
4	5	IPL-2017	Bangalore	08-04-2017	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	KM Jadhav	M Chinnaswamy Stadium	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
751	11347	IPL-2019	Mumbai	05-05-2019	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	0	Mumbai Indians	0	9	HH Pandya	Wankhede Stadium	Nanda Kishore	O Nandan
752	11412	IPL-2019	Chennai	07-05-2019	Chennai Super Kings	Mumbai Indians	Chennai Super Kings	bat	normal	0	Mumbai Indians	0	6	AS Yadav	M. A. Chidambaram Stadium	Nigel Llong	Nitin Menon
753	11413	IPL-2019	Visakhapatnam	08-05-2019	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	field	normal	0	Delhi Capitals	0	2	RR Pant	ACA-VDCA Stadium	NaN	NaN
754	11414	IPL-2019	Visakhapatnam	10-05-2019	Delhi Capitals	Chennai Super Kings	Chennai Super Kings	field	normal	0	Chennai Super Kings	0	6	F du Plessis	ACA-VDCA Stadium	Sundaram Ravi	Bruce Oxenford
755	11415	IPL-2019	Hyderabad	12-05-2019	Mumbai Indians	Chennai Super Kings	Mumbai Indians	bat	normal	0	Mumbai Indians	1	0	JJ Bumrah	Rajiv Gandhi Intl. Cricket Stadium	Nitin Menon	Ian Gould

التنفيذ

لذلك دعونا نبدأفي تنفيذ نموذج توقع فريق IPL.

كالعادة، فإن الخطوة الأولى هي استيراد جميع المكتبات المطلوبة.

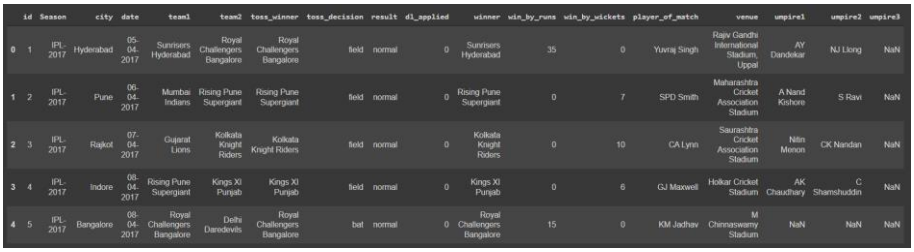
```
import pandas as pd
import numpy as np
import seaborn as sns
sns.set_style("whitegrid")
import matplotlib.pyplot as plt
import sklearn
```

فلنستورد مجموعة البيانات. لقد ناقشنا بالفعل مجموعة البيانات.

```
data = pd.read_csv("matches.csv")
```

جارٍ تحليل الصفوف الخمسة الأولى من مجموعة البيانات.

```
data.head()
```



	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_match	venue	umpire1	umpire2	umpire3
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj Singh	Rajiv Gandhi International Stadium, Uppal	AY Dandekar	NJ Llong	NaN
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	SPD Smith	Maharashtra Cricket Association Stadium	A Nand Kishore	S Ravi	NaN
2	3	IPL-2017	Rajkot	07-04-2017	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	CA Lynn	Saurashtra Cricket Association Stadium	Nitin Menon	CK Nandan	NaN
3	4	IPL-2017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	GJ Maxwell	Holkar Cricket Stadium	AK Chaudhary	C Shamshuddin	NaN
4	5	IPL-2017	Bangalore	08-04-2017	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	KM Jadhav	M Chinnaswamy Stadium	NaN	NaN	NaN

دعنا نحصل على ملخص موجز لمجموعة بيانات IPL.

```
data.describe()
```

	dl_applied	win_by_runs	win_by_wickets
count	756.000000	756.000000	756.000000
mean	0.025132	13.283069	3.350529
std	0.156630	23.471144	3.387963
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	4.000000
75%	0.000000	19.000000	6.000000
max	1.000000	146.000000	10.000000

التحقق من وجود أي قيم فارغةفي مجموعة البيانات.

```
data.isnull().sum()
```

```
id                  0 Season            0 city
7 date               0 team1             0 team2
0 toss_winner        0 toss_decision     0 result
0 dl_applied         0 winner            4 win_by_runs
0 win_by_wickets     0 player_of_match   4 venue
0 umpire1            2 umpire2           2 umpire3
637
```

كما ترى، فإن قيم umpire3 فارغةفي جميع الصفوف تقريبًا، لذلك نقوم بإسقاط العمود umpire3. وكذلك أسقطت بعض الصفوف التي تحتوي على القيم الخالية بعد إزالة عمود umpire3.

```
data = data.iloc[:,:-1]
data.dropna(inplace=True)
```

دعنا الآن نلقي نظرة على إجمالي الفرق المدرجةفي مجموعة البيانات هذه.

```
data["team1"].unique()
```

```
array(['Sunrisers Hyderabad', 'Mumbai Indians', 'Gujarat Lions',
       'Rising Pune Supergiant', 'Royal Challengers Bangalore',
       'Kolkata Knight Riders', 'Delhi Capitals', 'Kings XI Punjab',
       'Chennai Super Kings', 'Rajasthan Royals', 'Kochi Tuskers Kerala',
       'Pune Warriors', 'Rising Pune Supergiants'], dtype=object)
```

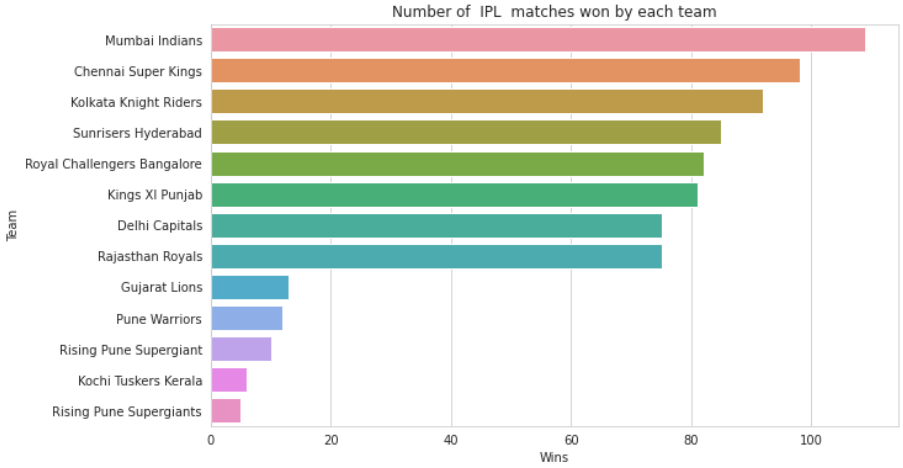
هنا يمكنك رؤية اسم Delhi Daredevils وDelhi Capitals؛ Delhi Daredevils هو الاسم القديم لعواصم دلهي. وبالمثل، فإن Decan Chargers هو الاسم القديم لـ Sunrisers Hyderabad. لذلك نحن بصدد تغيير الاسم القديم إلى الاسم الأحدث.

```
#for Delhi Capitals
data['team1']=data['team1'].str.replace('Delhi Daredevils','Delhi
Capitals')
data['team2']=data['team2'].str.replace('Delhi Daredevils','Delhi
Capitals')
data['winner']=data['winner'].str.replace('Delhi Daredevils','Delhi
Capitals')
#for sunrisers Hyderabad
data['team1']=data['team1'].str.replace('Deccan Chargers','Sunrisers
Hyderabad')
data['team2']=data['team2'].str.replace('Deccan Chargers','Sunrisers
Hyderabad')
data['winner']=data['winner'].str.replace('Deccan Chargers','Sunrisers
Hyderabad')
```

التمثيل البياني

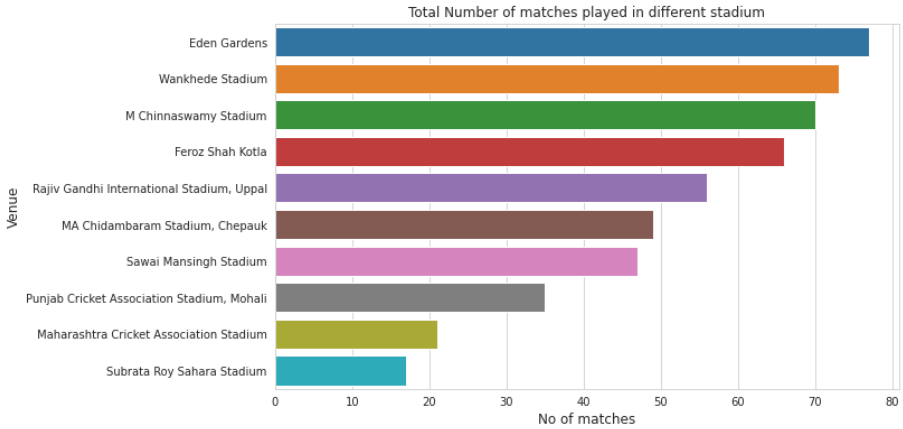
عدد مباريات IPL التي فاز بها كل فريق.

```
plt.figure(figsize = (10,6))
sns.countplot(y = 'winner',data = data,order=
data['winner'].value_counts().index)
plt.xlabel('Wins')
plt.ylabel('Team')
plt.title('Number of  IPL  matches won by each team')
```



إجمالي عدد المباريات التي تم لعبهافي ملعب مختلف .

```
plt.figure(figsize = (10,6))
sns.countplot(y = 'venue',data = data,order =
data['venue'].value_counts().iloc[:10].index)
plt.xlabel('No of matches',fontsize=12)
plt.ylabel('Venue',fontsize=12)
plt.title('Total Number of matches played in different stadium')
```



تم اتخاذ القرار من قبل الفريق الفائزفي القرعة.

```
plt.figure(figsize = (10,6))
sns.countplot(x = "toss_decision", data=data)
plt.xlabel('Toss Decision',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.title('Toss Decision')
```

دعنا الآن نتحقق من القيم الفريدة المقدمةفي كل ميزة.

```
x = ["city", "toss_decision", "result", "dl_applied"]
for i in x:
  print("------------")
  print(data[i].unique())
  print(data[i].value_counts())
```

```
------------
['Hyderabad' 'Pune' 'Rajkot' 'Indore' 'Mumbai' 'Kolkata' 'Bangalore'
 'Delhi' 'Chandigarh' 'Kanpur' 'Jaipur' 'Chennai' 'Cape Town'
 'Port Elizabeth' 'Durban' 'Centurion' 'East London' 'Johannesburg'
 'Kimberley' 'Bloemfontein' 'Ahmedabad' 'Cuttack' 'Nagpur' 'Dharamsala'
 'Kochi' 'Visakhapatnam' 'Raipur' 'Ranchi' 'Abu Dhabi' 'Sharjah' 'Mohali'
 'Bengaluru']
Mumbai            101
Kolkata            77
Delhi              73
Hyderabad          64
Bangalore          63
Chennai            57
Jaipur             47
Chandigarh         46
Pune               38
Durban             15
Bengaluru          13
```

```
Centurion          12
Ahmedabad          12
Visakhapatnam      12
Rajkot             10
Mohali             10
Indore              9
Dharamsala          9
Johannesburg        8
Cuttack             7
Ranchi              7
Port Elizabeth      7
Cape Town           7
Abu Dhabi           7
Sharjah             6
Raipur              6
Kochi               5
Kanpur              4
Nagpur              3
Kimberley           3
East London         3
Bloemfontein        2
Name: city, dtype: int64
------------
['field' 'bat']
field    455
bat      288
Name: toss_decision, dtype: int64
------------
['normal' 'tie']
normal    734
tie         9
Name: result, dtype: int64
------------
[0 1]
```

```
0    724

1     19

Name: dl_applied, dtype: int64
```

لا نحتاج إلى جميع الميزات أو الأعمدة من أجل إنشاء النموذج. سيؤدي ذلك إلى تقليل دقة النموذج، لذا فإننا نسقط بعض الميزات التي لا تؤثر على النتيجة.

```
data.drop(["id", "Season","city","date", "player_of_match", 'umpire1',
"venue", "umpire2"], axis=1, inplace=True)
```

تبدو بياناتنا هكذا.

	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets
0	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0
1	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7
2	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10
3	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6
5	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	normal	0	Sunrisers Hyderabad	0	9
...	...	...	...	...	...	...	...	...	...
750	Chennai Super Kings	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6
751	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	0	Mumbai Indians	0	9
752	Chennai Super Kings	Mumbai Indians	Chennai Super Kings	bat	normal	0	Mumbai Indians	0	6
754	Delhi Capitals	Chennai Super Kings	Chennai Super Kings	field	normal	0	Chennai Super Kings	0	6
755	Mumbai Indians	Chennai Super Kings	Mumbai Indians	bat	normal	0	Mumbai Indians	1	0

يمكننا تحويل بياناتنا إلى بيانات تابعة ومستقلة.

```
X = data.drop(["winner"], axis=1)
y = data["winner"]
```

توجد عدة قيم فئويةفي بيانات الإدخال، لذلك نقوم بتحويلها إلى قيم عددية باستخدام pandas، طريقة get_dummies.

```
X = pd.get_dummies(X, ["team1","team2", "toss_winner",
"toss_decision", "result"], drop_first = True)
```

بيانات الإخراج هي أيضًا قيمة فئوية، لذلك نقوم بتحويلها إلى عدد باستخدام LabelEncoder من sklearn.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

دعنا الآن نحول بياناتنا إلى مجموعة تدريب من أجل إنشاء النموذج ومجموعة الاختبار لتقييم النموذج الذي تم إنشاؤه.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, train_size =
0.8)
```

إنشاء النموذج وتقييمه

الخطوة التالية والأكثر أهميةفي خطوة إنشاء النموذج. لذلك نحن نستخدم التصنيف العشوائي للغابات والانحدار اللوجستي وتصنيف شجرة القرار لهذا الغرض.

```
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier(n_estimators=200,min_samples_split=3,
                               max_features = "auto")
```

تدريب نموذج مصنف الغابة العشوائي.

```
model.fit(x_train, y_train)
```

توقع النموذج بقيم x_test وحفظه كـ y_pred.

```
y_pred = model.predict(x_test)
```

باستخدام درجة الدقة accuracy score من sklearn، نقوم بتقييم النتيجة المتوقعة ودقة النموذج.

```
from sklearn.metrics import accuracy_score
ac = accuracy_score(y_pred, y_test)
```

```
#output - 0.92
```

الملخص

يوضح هذا المقال تطبيق نموذج التنبؤ IPL Win. لقد حصلت على نظرة ثاقبة حول كيفية تحليل بيانات أولية معينة وتحويلها إلى ميزات مفيدة عن طريق إزالة الميزات غير المرغوب فيها، أي إجراء تحليل البيانات الاستكشافية. لذلك دعونا نحدد النقاط الرئيسية من المقالة.

- **تحليل مجموعة بيانات IPL**: يتضمن هذا التحليل التحقق من القيم الخالية واستبدالها، ووصف أعمدة ميزات مجموعة البيانات، وتحليل كل ميزة.
- **رسم البيانات**: تم تمثيل رسومي لمجموعة البيانات من أجل فهم الفرق والمباريات وقرار القذف لمباريات IPL.
- **تقنيات المعالجة المسبقة على بيانات IPL**: نفذت العديد من تقنيات هندسة الميزات لجعل مجموعة البيانات مناسبة لصنع النموذج. يتم استخدام الترميز لتحويل الميزات الفئوية إلى ميزات وتقنيات رقمية لتجنب القيم الخالية.
- **إنشاء النموذج وتقييمه**: كان هذا هو الجزء الرئيسي من المشروع، واستخدمنا نموذج RandomForestClassifier.في وقت لاحق اختبرنا هذا النموذج مع مجموعة الاختبار للتقييم.

آمل أن تكون لديك فكرة بخصوص الخطوات المذكورة أعلاه. يرجى التأكد من أنك تمارس وتحاول فهم كل خطوة.

39) الكشف المبكر عن مرض باركنسون باستخدام التعلم الآلي
Parkinson disease onset detection Using Machine Learning

الهدف

الهدف الرئيسي من هذه المقالة هو فهم ما هو مرض باركنسون واكتشاف البداية المبكرة للمرض. سنستخدم هنا XGBoost وKNN Algorithm و Support Vector Machines (SVMs) وRandom Forest Algorithm ونستخدم مجموعة البيانات المتاحة على مجموعة بيانات UCL Parkinson ضمن عنوان URL (Index of /ml/machine-learning-databases/Parkinsons (uci.edu).

مرض باركنسون

مرض باركنسون هو اضطراب عصبيفي المخ. يؤدي إلى اهتزاز الجسم واليدين وتيبس الجسم. لا يوجد علاج أو علاج مناسب حتى الآنفي المرحلة المتقدمة. العلاج ممكن فقط عندما يتم إجراؤهفي وقت مبكر من المرض أوفي بداية ظهوره. لن يقلل ذلك من تكلفة المرض فحسب، بل سينقذ حياة أيضًا. يمكن لمعظم الطرق المتاحة اكتشاف مرض باركنسونفي مرحلة متقدمة؛ وهو ما يعني فقدان ما يقرب من.. 60٪ من الدوباميفي العقد القاعدية وهو مسؤول عن التحكمفي حركة الجسم بكمية صغيرة من الدوبامين. تم العثور على أكثر من 145000 شخص يعانون بمفردهمفي المملكة المتحدة والهند، ويعاني ما يقرب من مليون شخص من هذا المرض وينتشر بسرعةفي العالم بأسره.

يمكن أن يعاني الشخص المصاب بمرض باركنسون من أعراض أخرى تشمل:

1. الاكتئاب.
2. القلق.
3. النوم والقضايا المتعلقة بالذاكرة.
4. فقدان حاسة الشم مع مشاكل التوازن..

لا يزال سبب مرض باركنسون غير واضح، لكن الباحثين أجروا أبحاثًا تشير إلى أن عدة عوامل مسؤولة عن تحفيز المرض. ويشمل:

1. الجينات: تم العثور على جينات طفرة معينة عن طريق البحث وهي نادرة جدًا. غالبًا ما تزيد المتغيرات الجينية من خطر الإصابة بمرض باركنسون ولكن لها تأثير أقل على كل علامة جينية.

2. البيئة: بسبب بعض السموم الضارة أو المواد الكيميائية الموجودةفي البيئة يمكن أن تسبب المرض ولكن لها تأثير أقل.

على الرغم من أنه يتطورفي عمر 65 عامًا، إلا أنه يمكن العثور عليهفي سن مبكرة أقل من 50 عامًا. سنستخدم خوارزمية XGBoost وKNN وSVM وRandom Forest للتحقق من أفضل خوارزمية لاكتشاف بداية المرض.

ما هو XGBoost؟

XGBoost هي خوارزمية. لقد كان هذا الأمر يهيمن مؤخرًا على تعلم الأدوات التطبيقية. مجموعة قواعد XGBoost هي تطبيق لـ gradient boosted choice timber. تغير ذلك إلى تصميم السرعة والأداء العام.

الكود:

- استيراد مكتبات NumPy وPandas و Sklearn و XGBoost.

```
import numpy as np
import pandas as pd
import os, sys
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

- قراءة بيانات ملف مرض باركنسون.

```
features=df.loc[:,df.columns!='status'].values[:,1:]

labels=df.loc[:,'status'].values

print(labels[labels==1].shape[0], labels[labels==0].shape[0])


scaler=MinMaxScaler((-1,1))

x=scaler.fit_transform(features)

y=labels

x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.2,
random state=7)
model=XGBClassifier(eval_metric='mlogloss')

model.fit(x_train,y_train)

Output - XGBClassifier(base_score=0.5, booster='gbtree',
colsample_bylevel=1,
```

```
colsample_bynode=1, colsample_bytree=1, eval_metric='mlogloss',
gamma=0, gpu_id=-1, importance_type='gain',
interaction_constraints='', learning_rate=0.300000012,
max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
monotone_constraints='()', n_estimators=100, n_jobs=4,
num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1,
scale_pos_weight=1, subsample=1, tree_method='exact',
use_label_encoder=False, validate_parameters=1, verbosity=None)
```

```
y_pred=model.predict(x_test)

print(accuracy_score(y_test, y_pred)*100)
```

```
Output - 94.87179487179486
```

```
from sklearn.metrics import confusion_matrix

pd.DataFrame(

    confusion_matrix(y_test, y_pred),

    columns=['Predicted Healthy', 'Predicted Parkinsons'],

    index=['True Healthy', 'True Parkinsons']

)
```

	Predicted Healthy	Predicted Parkinsons
True Healthy	6	1
True Parkinsons	1	31

يظهر دقة 94٪ بواسطة خوارزمية XGBoost. الآن سنستخدم Random Forest.

تعتبر أشجار القرار Decision trees جهازًا استثنائيًا، ولكن يمكنهافي كثير من الأحيان الإفراط overifiting في احتواء مجموعة الحقائق التدريبية حتى يتم تقليمها pruned بشكل فعال، مما يعيق قدراتها التنبؤية.

ما هي SVM؟

خوارزمية أخرى لتحليل التصنيف والانحدار هي آلة متجه الدعم SVM.

إنها خوارزمية آلة خاضعة للإشراف مستخدمة. تصنيف الصور والتعرف المكتوب بخط اليد هو المكان الذي تستخدم فيه آلة ناقلات الدعم. يقوم بفرز البياناتفي فئة واحدة من فئتين ويعرض الإخراج بالهامش بينهما قدر الإمكان.

الكود:

```
#fitting the model in SVM
classifi2.fit(x_train,y_train)
print(accuracy_score(y_test, y2_pred)*100)
```

```
from sklearn.svm import SVC
classifi2 = SVC()
#predicting reults
Output-
87.17948717948718
```

```
y2_pred = classifi2.predict(x_test)
```

يُظهر نموذج الإخراج لـ SVMs دقة بنسبة 87 ٪ لمجموعة البيانات المحددة.

	Predicted Healthy	Predicted Parkinsons
True Healthy	2	5
True Parkinsons	0	32

ما هو KNN؟

تعد خوارزمية K-Nearest Neighbours (KNN) واحدة من أقوى الخوارزميات المستخدمةفي التعلم الآلي والتي تستخدم على نطاق واسع لكل من مهام الانحدار والتصنيف. من أجل التنبؤ بالفئة التي تقع فيها نقاط البيانات وفحصها، تقوم بفحص تسمية نقاط البيانات المختارة والمحاطة بالنقطة المستهدفة.

الكود:

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.decomposition import PCA

 pca = PCA(n_components = 2)

 x_train = pca.fit_transform(x_train)

 x_test = pca.transform(x_test)

 variance = pca.explained_variance_ratio_

 classifi = KNeighborsClassifier(n_neighbors = 8,p=2,metric
='minkowski')

 classifi.fit(x_train,y_train)

 y_pred = classifi.predict(x_test)

 from sklearn.metrics import confusion_matrix,accuracy_score

 #KNN model

 cm=confusion_matrix(y_test,y_pred)

 accuracy_score(y_test,y_pred)
```

```
Output - 0.8974358974358975
```

ما هي Random Forest؟

تعد الغابات العشوائية Random Forest إصدارًا مجمعًا للعديد من شجيرات الاختيار، حيث ستتخصص كل شجرةفي تركيزها على ميزة معينة مع الحفاظ على عرض المستوى الأعلى لجميع الإمكانات.

ستعمل كل شجرة داخل المنطقة المشجرة العشوائية على تقسيم المعلومات / فحصها العشوائي، والمشار إليه بتجميع التمهيد bootstrap aggregation، وتسمى العينات التي لم تعد مغطاة عينات " out-of-bag ". علاوة على ذلك، ستقوم كل شجرة بتعبئة مميزةفي كل انقسامفي فرع العقدة لتقليل نتائج خاصية مرتبطةفي الغالب بالاستجابة.

في حين أن الشجرة الفردية ربما تكون حساسة للقيم المتطرفة، فإن إصدار المجموعة لن يكون هو نفسه.

```
X = df.drop('status', axis=1)


X = X.drop('name', axis=1)

y = df['status']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=1)
```

```
from sklearn.ensemble import RandomForestClassifier

random_forest = RandomForestClassifier(n_estimators=30, max_depth=10,
random_state=1)
```

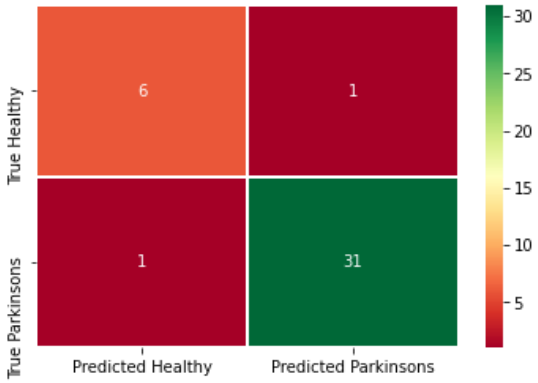
```
random_forest.fit(x_train, y_train)


from sklearn.metrics import accuracy_score

y_predict = random_forest.predict(x_test)

accuracy_score(y_test, y_predict)

Output - 0.9387755102040817

Random Forest shows accuracy 93% almost less then XGBoost Algorithm.

from sklearn.metrics import confusion_matrix


pd.DataFrame(

confusion_matrix(y_test, y_predict),
```

```
columns=['Predicted Healthy', 'Predicted Parkinsons'],
index=['True Healthy', 'True Parkinsons']
)
```

الخريطة الحرارية

الآن، دعنا نأخذ خريطة حرارية heatmap للبيانات المتوقعة بواسطة خوارزمية XGBoost.

```
import seaborn as sns

sns.heatmap(a, cmap ='RdYlGn', linewidths = 0.30, annot = True)
```



يبلغ عدد مرضى باركنسون المتوقع 31 عامًا على خريطة الحرارة.

الملخص

يؤثر مرض باركنسون على الجهاز العصبي المركزي للدماغ ولا يخضع للعلاج حتى الآن إلا إذا تم اكتشافه مبكرًا. يؤدي الاكتشاف المتأخر إلى عدم العلاج وخسارة الأرواح. وبالتالي فإن اكتشافه المبكر مهم. للكشف المبكر عن المرض، استخدمنا خوارزميات التعلم الآلي مثل XGBoost وRandom Forest. لقد فحصنا بيانات مرض باركنسون واكتشفنا أن XGBoost هي أفضل خوارزمية للتنبؤ بظهور المرض والتي ستتيح العلاج المبكر وإنقاذ الحياة.

40) تصنيف تسلسل الحمض النووي باستخدام التعلم الآلي Classifying DNA Sequences using machine learning

خلال هذا البرنامج التعليمي، سوف نستكشف عالم المعلوماتية الحيوية bioinformatics باستخدام نماذج ماركوف وخوارزميات K ـ أقرب الجيران (KNN) وآلات المتجهات الداعمة (SVM) والمصنفات الشائعة الأخرى لتصنيف تسلسلات E. Coli DNA القصيرة. سيستخدم هذا المشروع مجموعة بيانات من UCI Machine Learning Repository التي تحتوي على 106 تسلسل DNA، مع 57 نيوكليوتيد متسلسل ("أزواج قاعدية") لكل منها.

سوف تتعلم كيفية:

- استيراد البيانات من مستودع UCI.
- تحويل مدخلات النص إلى بيانات رقمية.
- بناء وتدريب خوارزميات التصنيف.
- قارن وقارن بين خوارزميات التصنيف.

الخطوة 1: استيراد مجموعة البيانات

ستقوم خلايا التعليمات البرمجية التالية باستيراد المكتبات الضرورية واستيراد مجموعة البيانات من مستودع UCI باعتباره Pandas DataFrame.

- للتأكد من تثبيت جميع المكتبات الصحيحة، قم باستيراد كل وحدة وطباعة رقم الإصدار

```
# To make sure all of the correct libraries are installed, import each
module and print the version number

import sys
import numpy
import sklearn
import pandas

print('Python: {}'.format(sys.version))
print('Numpy: {}'.format(numpy.__version__))
print('Sklearn: {}'.format(sklearn.__version__))
print('Pandas: {}'.format(pandas.__version__))
```

```
Python: 2.7.13 |Continuum Analytics, Inc.| (default, May 11 2017, 13:1
7:26) [MSC v.1500 64 bit (AMD64)]
Numpy: 1.14.0
Sklearn: 0.19.1
Pandas: 0.21.0
```

```
# Import, change module names
import numpy as np
import pandas as pd
```

```
# import the uci Molecular Biology (Promoter Gene Sequences) Data Set
url = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/molecular-biology/promoter-gene-sequences/promoters.data'
names = ['Class', 'id', 'Sequence']
data = pd.read_csv(url, names = names)
print(data.iloc[0])
```

```
Class                                                      +
id                                                       S10
Sequence    \t\ttactagcaatacgcttgcgttcggtggttaagtatgtataat...
Name: 0, dtype: object
```

الخطوة 2: المعالجة المسبقة لمجموعة البيانات

البيانات ليستفي شكل صالح للاستخدام؛ نتيجة لذلك، سنحتاج إلى معالجته قبل استخدامه لتدريب خوارزمياتنا.

- بناء مجموعة البيانات الخاصة بنا عن طريق إنشاء إطار بيانات Pandas مخصص.
- يُطلق على كل عمودفي إطار بيانات اسم سلسلة. لنبدأ بعمل سلسلة لكل عمود.

```
# Building our Dataset by creating a custom Pandas DataFrame
# Each column in a DataFrame is called a Series. Lets start by making
a series for each column.

classes = data.loc[:, 'Class']
print(classes[:5])
```

```
0    +
1    +
2    +
3    +
4    +
Name: Class, dtype: object
```

- إنشاء قائمة بتسلسل الحمض النووي.

```
# generate list of DNA sequences
sequences = list(data.loc[:, 'Sequence'])
dataset = {}

# loop through sequences and split into individual nucleotides
for i, seq in enumerate(sequences):

    # split into nucleotides, remove tab characters
    nucleotides = list(seq)
    nucleotides = [x for x in nucleotides if x != '\t']

    # append class assignment
    nucleotides.append(classes[i])

    # add to dataset
    dataset[i] = nucleotides

print(dataset[0])
```

```
['t', 'a', 'c', 't', 'a', 'g', 'c', 'a', 'a', 't', 'a', 'c', 'g', 'c',
't', 't', 'g', 'c', 'g', 't', 't', 'c', 'g', 'g', 't', 'g', 'g', 't',
't', 'a', 'a', 'g', 't', 'a', 't', 'g', 't', 'a', 't', 'a', 'a', 't',
```

```
'g', 'c', 'g', 'c', 'g', 'g', 'g', 'c', 't', 't', 'g', 't', 'c', 'g',
't', '+']
# turn dataset into pandas DataFrame
dframe = pd.DataFrame(dataset)
print(dframe)
```

```
0   1   2   3   4   5   6   7   8   9   ... 96  97  98  99  100 101 10
2  \
0    t   t   g   a   t   a   c   t   c   t ...   c   c   t   a   g   c
g
1    a   g   t   a   c   g   a   t   g   t ...   c   g   a   g   a   c
t
2    c   c   a   t   g   g   g   t   a   t ...   g   c   t   a   g   t
a
3    t   t   c   t   a   g   g   c   c   t ...   a   t   g   g   a   c
t
4    a   a   t   g   t   g   g   t   t   a ...   g   a   a   g   g   a
t
5    g   t   a   t   a   c   g   a   t   a ...   t   g   c   g   c   a
c
6    c   c   g   g   a   a   g   c   a   a ...   a   g   c   t   a   t
t
7    a   c   a   a   t   a   t   a   a   t ...   g   a   g   g   t   g
c
8    a   t   g   t   t   g   g   a   t   t ...   a   c   a   t   g   g
a
9    t   g   a   g   a   g   g   a   a   t ...   c   t   a   a   t   c
a
10   a   a   a   t   a   a   a   a   t   c ...   c   t   c   c   c   c
c
11   c   c   c   g   c   g   g   c   a   c ...   c   t   g   t   a   t
a
12   g   a   t   t   t   g   g   a   c   t ...   t   c   a   c   g   c
a
13   c   g   a   a   a   a   a   c   t   c ...   t   t   g   c   c   t
g
14   t   t   g   t   t   t   t   t   g   t ...   a   t   t   a   c   a
a
15   t   t   t   c   t   g   t   t   c   t ...   g   g   c   a   t   a
t
16   g   g   g   g   g   g   t   g   g   g ...   a   t   a   g   c   a
t
17   c   t   c   a   a   a   a   a   a   t ...   g   t   a   a   g   c
a
18   g   c   a   a   c   a   a   t   c   c ...   a   g   t   a   a   g
a
19   t   a   t   g   g   a   g   a   a   a ...   g   a   c   g   c   g
c
```

20	t	c	t	t	a	g	c	c	g	g	...	c	t	a	a	a	g	c
21	c	g	a	g	a	a	c	t	g	g	...	a	t	g	g	a	t	g
22	g	c	g	t	a	g	a	g	a	c	...	t	t	a	g	c	c	a
23	g	t	c	g	a	g	t	t	c	c	...	g	t	c	a	t	t	c
24	t	g	t	t	g	t	c	a	g	g	...	t	c	c	a	t	t	a
25	g	a	t	t	c	t	t	t	t	g	...	c	c	g	g	g	g	g
26	g	t	a	g	t	g	c	g	c	a	...	a	a	c	a	c	a	a
27	t	t	t	c	g	c	c	a	c	a	...	g	t	t	t	a	g	t
28	t	g	t	g	a	c	t	g	g	t	...	c	g	t	g	t	g	t
29	a	g	t	g	a	g	g	c	t	a	...	c	c	t	a	a	g	c
30	a	t	t	a	a	t	a	a	t	a	...	t	g	g	g	a	g	a
31	g	g	t	g	a	a	t	t	c	c	...	c	g	a	g	a	t	a
32	t	t	t	t	c	t	g	a	t	t	...	g	t	c	c	t	t	t
33	a	c	t	a	c	a	a	c	g	c	...	a	g	t	t	g	t	c
34	t	g	g	g	a	a	c	a	t	c	...	c	t	c	a	c	t	t
35	g	t	t	a	c	a	g	g	g	c	...	a	t	t	g	t	t	c
36	t	t	t	t	t	g	c	t	t	t	...	a	t	g	a	t	t	g
37	a	a	a	g	a	a	a	a	a	a	...	c	t	g	c	t	g	t
38	t	c	t	t	g	a	t	t	a	t	...	t	g	t	g	c	c	g
39	a	a	c	t	a	a	a	a	a	a	...	t	c	a	t	t	t	g
40	a	a	a	a	a	c	g	a	t	a	...	g	g	t	c	t	g	a
41	t	t	t	g	t	t	t	t	c	t	...	c	c	t	t	g	a	t
42	g	c	g	a	g	a	c	t	g	g	...	a	a	a	c	t	a	g

43	c	t	c	a	c	g	a	g	c	c	...	t	a	c	t	a	a
g																	
44	g	a	t	t	g	a	g	c	a	g	...	a	t	t	g	g	g
a																	
45	c	a	a	a	c	g	c	t	a	c	...	a	g	g	c	a	g
c																	
46	g	c	a	c	c	t	c	t	t	c	...	a	t	t	a	c	a
g																	
47	g	g	c	t	t	c	c	c	g	a	...	t	t	g	t	g	g
t																	
48	g	c	c	a	c	c	a	a	a	c	...	g	a	a	g	t	g
t																	
49	c	a	a	a	c	g	t	a	a	c	...	c	a	a	g	g	a
c																	
50	t	t	c	c	g	t	c	c	a	a	...	t	t	c	a	c	a
a																	
51	t	c	c	a	t	t	a	a	t	c	...	t	c	a	g	c	c
a																	
52	g	g	c	a	g	t	t	g	g	t	...	t	g	t	t	c	t
c																	
53	t	c	g	a	g	a	g	a	g	g	...	c	c	t	a	t	a
a																	
54	c	c	g	c	t	g	a	a	t	a	...	t	t	a	t	a	t
t																	
55	g	a	c	t	a	g	a	c	t	c	...	t	t	t	g	c	a
t																	
56	t	a	g	c	g	t	t	a	t	a	...	g	t	t	a	g	t
g																	
57	+	+	+	+	+	+	+	+	+	+	...	-	-	-	-	-	-
-																	

	103	104	105
0	c	c	t
1	g	t	a
2	c	c	a
3	g	g	c
4	a	t	a
5	c	c	t
6	t	c	t
7	a	t	a
8	c	c	a
9	g	a	t
10	a	a	a
11	t	t	a
12	g	g	a
13	a	g	t
14	g	c	a

```
15   a   c   a
16   t   t   g
17   g   c   g
18   c   t   a
19   c   a   g
20   t   a   g
21   g   a   c
22   a   c   t
23   g   g   c
24   t   g   t
25   g   g   a
26   c   t   a
27   t   c   t
28   t   t   g
29   c   t   g
30   c   g   c
31   g   a   a
32   t   g   c
33   t   g   t
34   a   g   c
35   c   g   a
36   t   t   t
37   g   t   t
38   g   t   a
39   a   t   g
40   t   t   c
41   t   t   c
42   g   g   a
43   t   c   a
44   c   t   t
45   a   g   c
46   c   a   a
47   c   a   a
48   a   a   t
49   a   g   c
50   g   g   a
51   g   a   a
52   c   g   g
53   t   g   a
54   t   a   a
55   c   a   c
56   c   c   t
57   -   -   -

[58 rows x 106 columns]
```

- تبديل إطار البيانات.

```
# transpose the DataFrame
df = dframe.transpose()
print(df.iloc[:5])
```

```
0  1  2  3  4  5  6  7  8  9  ... 48 49 50 51 52 53 54 55 56 57
0  t  a  c  t  a  g  c  a  a  t ...  g  c  t  t  g  t  c  g  t  +
1  t  g  c  t  a  t  c  c  t  g ...  c  a  t  c  g  c  c  a  a  +
2  g  t  a  c  t  a  g  a  g  a ...  c  a  c  c  c  g  g  c  g  +
3  a  a  t  t  g  t  g  a  t  g ...  a  a  c  a  a  a  c  t  c  +
4  t  c  g  a  t  a  a  t  t  a ...  c  c  g  t  g  g  t  a  g  +

[5 rows x 58 columns]
```

- من أجل الوضوح، دعنا نعيد تسمية عمود إطار البيانات الأخير بالفئة.

```
# for clarity, lets rename the last dataframe column to class
df.rename(columns = {57: 'Class'}, inplace = True)
print(df.iloc[:5])
```

```
0  1  2  3  4  5  6  7  8  9  ...  48 49 50 51 52 53 54 55 56 Class
0  t  a  c  t  a  g  c  a  a  t  ...  g  c  t  t  g  t  c  g  t     +
1  t  g  c  t  a  t  c  c  t  g  ...  c  a  t  c  g  c  c  a  a     +
2  g  t  a  c  t  a  g  a  g  a  ...  c  a  c  c  c  g  g  c  g     +
3  a  a  t  t  g  t  g  a  t  g  ...  a  a  c  a  a  a  c  t  c     +
4  t  c  g  a  t  a  a  t  t  a  ...  c  c  g  t  g  g  t  a  g     +
```

- لنبدأ في التعرف على مجموعة البيانات حتى نتمكن من اختيار الأنسب.

```
[5 rows x 58 columns]
# looks good! Let's start to familiarize ourselves with the dataset so
we can pick the most suitable
# algorithms for this data

df.describe()
```

	0	1	2	3	4	5	6	7	8	9	...	48	49	50	51	52	53	54	55	56	Class
count	106	106	106	106	106	106	106	106	106	106	...	106	106	106	106	106	106	106	106	106	106
unique	4	4	4	4	4	4	4	4	4	4	...	4	4	4	4	4	4	4	4	4	2
top	t	a	a	c	a	a	a	a	a	a	...	c	c	c	t	t	c	c	t	t	-
freq	38	34	30	30	36	42	38	34	33	36	...	36	42	31	33	35	32	29	29	34	53

4 rows × 58 columns

- الوصف لا يخبرنا بمعلومات كافية لأن السمات هي نص. يتيح تسجيل قيمة السجل لكل تسلسل.

```
# desribe does not tell us enough information since the attributes are
text. Lets record value counts for each sequence
series = []
for name in df.columns:
    series.append(df[name].value_counts())

info = pd.DataFrame(series)
details = info.transpose()
```

```
print(details)
```

```
      0     1     2     3     4     5     6     7     8     9  ...    48  \
+   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...   NaN
-   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...   NaN
a  26.0  34.0  30.0  22.0  36.0  42.0  38.0  34.0  33.0  36.0  ...  23.0
c  27.0  22.0  21.0  30.0  19.0  18.0  21.0  20.0  22.0  22.0  ...  36.0
g  15.0  24.0  28.0  28.0  29.0  22.0  17.0  20.0  19.0  20.0  ...  26.0
t  38.0  26.0  27.0  26.0  22.0  24.0  30.0  32.0  32.0  28.0  ...  21.0

     49    50    51    52    53    54    55    56  Class
+   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   53.0
-   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   53.0
a  24.0  28.0  27.0  25.0  22.0  26.0  24.0  27.0    NaN
c  42.0  31.0  32.0  21.0  32.0  29.0  29.0  17.0    NaN
g  18.0  24.0  14.0  25.0  22.0  28.0  24.0  28.0    NaN
t  22.0  23.0  33.0  35.0  30.0  23.0  29.0  34.0    NaN

[6 rows x 58 columns]
```

- لسوء الحظ، لا يمكننا تشغيل خوارزميات التعلم الآلي على البيانات بتنسيقات "String"نتيجة لذلك، نحن بحاجة إلى التبديل.

```
# Unfortunately, we can't run machine learning algorithms on the data
in 'String' formats. As a result, we need to switch
# it to numerical data. This can easily be accomplished using the
pd.get_dummies() function
numerical_df = pd.get_dummies(df)
numerical_df.iloc[:5]
```

	0_a	0_c	0_g	0_t	1_a	1_c	1_g	1_t	2_a	2_c	...	55_a	55_c	55_g	55_t	56_a	56_c	56_g	56_t	Class_+	Class_-
0	0	0	0	1	1	0	0	0	0	1	...	0	0	1	0	0	0	0	1	1	0
1	0	0	0	1	0	0	1	0	0	1	...	1	0	0	0	1	0	0	0	1	0
2	0	0	1	0	0	0	0	1	1	0	...	0	1	0	0	0	0	1	0	1	0
3	1	0	0	0	1	0	0	0	0	0	...	0	0	0	1	0	1	0	0	1	0
4	0	0	0	1	0	1	0	0	0	0	...	1	0	0	0	0	0	1	0	1	0

5 rows × 230 columns

- لا نحتاج إلى كلا عمودي الفئة. دعنا نترك أحدهما ثم نعيد تسمية الآخر ليصبح ببساطة "Class".

```
# We don't need both class columns.  Lets drop one then rename the
other to simply 'Class'.
df = numerical_df.drop(columns=['Class_-'])

df.rename(columns = {'Class_+': 'Class'}, inplace = True)
```

```
print(df.iloc[:5])
```

```
   0_a  0_c  0_g  0_t  1_a  1_c  1_g  1_t  2_a  2_c  ...    54_t  55_a  55_c  \
0    0    0    0    1    1    0    0    0    0    1  ...       0     0     0
1    0    0    0    1    0    0    1    0    0    1  ...       0     1     0
2    0    0    1    0    0    0    0    1    1    0  ...       0     0     1
3    1    0    0    0    1    0    0    0    0    0  ...       0     0     0
4    0    0    0    1    0    1    0    0    0    0  ...       1     1     0

   55_g  55_t  56_a  56_c  56_g  56_t  Class
0     1     0     0     0     0     1      1
1     0     0     1     0     0     0      1
2     0     0     0     0     1     0      1
3     0     1     0     1     0     0      1
4     0     0     0     0     1     0      1

[5 rows x 229 columns]
```

- استخدم model_selection للفصل بين مجموعات بيانات التدريب والاختبار.

```
# Use the model_selection module to separate training and testing
datasets
from sklearn import model_selection

# Create X and Y datasets for training
X = np.array(df.drop(['Class'], 1))
y = np.array(df['Class'])

# define seed for reproducibility
seed = 1

# split data into training and testing datasets
X_train, X_test, y_train, y_test = model_selection.train_test_split(X,
y, test_size=0.25, random_state=seed)
```

الخطوة 3: تدريب واختبار خوارزميات التصنيف

الآن بعد أن قمنا بمعالجة البيانات وقمنا ببناء مجموعات بيانات التدريب والاختبار الخاصة بنا، يمكننا البدءفي نشر خوارزميات تصنيف مختلفة. من السهل نسبيًا اختبار نماذج متعددة؛ نتيجة لذلك، سوف نقوم بمقارنة أداء عشرة خوارزميات مختلفة ومقارنتها.

- الآن بعد أن أصبح لدينا مجموعة البيانات الخاصة بنا، يمكننا البدءفي بناء الخوارزميات! سنحتاج إلى استيراد كل خوارزمية نخطط لاستخدامها.
- من sklearn. نحتاج أيضًا إلى استيراد بعض مقاييس الأداء، مثل مقياس الدقة وتقرير التصنيف.

```
# Now that we have our dataset, we can start building algorithms!
We'll need to import each algorithm we plan on using
# from sklearn.  We also need to import some performance metrics, such
as accuracy_score and classification_report.

from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score

# define scoring method
scoring = 'accuracy'

# Define models to train
names = ["Nearest Neighbors", "Gaussian Process",
         "Decision Tree", "Random Forest", "Neural Net", "AdaBoost",
         "Naive Bayes", "SVM Linear", "SVM RBF", "SVM Sigmoid"]

classifiers = [
    KNeighborsClassifier(n_neighbors = 3),
    GaussianProcessClassifier(1.0 * RBF(1.0)),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10,
max_features=1),
    MLPClassifier(alpha=1),
    AdaBoostClassifier(),
    GaussianNB(),
    SVC(kernel = 'linear'),
    SVC(kernel = 'rbf'),
    SVC(kernel = 'sigmoid')
]

models = zip(names, classifiers)

# evaluate each model in turn
results = []
names = []

for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state = seed)
    cv_results = model_selection.cross_val_score(model, X_train,
y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
Nearest Neighbors: 0.823214 (0.113908)
Gaussian Process: 0.873214 (0.056158)
Decision Tree: 0.750000 (0.185405)
Random Forest: 0.580357 (0.106021)
C:\Programdata\anaconda2\lib\site-packages\sklearn\neural_network\mult
ilayer_perceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Ma
ximum iterations (200) reached and the optimization hasn't converged y
et.
  % self.max_iter, ConvergenceWarning)
Neural Net: 0.887500 (0.087500)
AdaBoost: 0.912500 (0.112500)
Naive Bayes: 0.837500 (0.137500)
SVM Linear: 0.850000 (0.108972)
SVM RBF: 0.737500 (0.117925)
SVM Sigmoid: 0.569643 (0.159209)
```

- تذكر أن الأداء على بيانات التدريب ليس بهذه الأهمية. نريد أن نعرف مدى جودة خوارزمياتنا
- يمكن التعميم على البيانات الجديدة. لاختبار ذلك، دعنا نتنبأ بمجموعة بيانات التحقق.

```
# Remember, performance on the training data is not that important. We
want to know how well our algorithms
# can generalize to new data.  To test this, let's make predictions on
the validation dataset.

for name, model in models:
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(name)
    print(accuracy_score(y_test, predictions))
    print(classification_report(y_test, predictions))

# Accuracy - ratio of correctly predicted observation to the total
observations.
# Precision - (false positives) ratio of correctly predicted positive
observations to the total predicted positive observations
# Recall (Sensitivity) - (false negatives) ratio of correctly
predicted positive observations to the all observations in actual
class - yes.
# F1 score - F1 Score is the weighted average of Precision and Recall.
Therefore, this score takes both false positives and false
```

```
Nearest Neighbors
0.7777777777777778
             precision    recall  f1-score   support

          0       1.00      0.65      0.79        17
          1       0.62      1.00      0.77        10

avg / total       0.86      0.78      0.78        27

Gaussian Process
0.8888888888888888
             precision    recall  f1-score   support

          0       1.00      0.82      0.90        17
          1       0.77      1.00      0.87        10

avg / total       0.91      0.89      0.89        27

Decision Tree
0.7777777777777778
             precision    recall  f1-score   support

          0       1.00      0.65      0.79        17
          1       0.62      1.00      0.77        10

avg / total       0.86      0.78      0.78        27

Random Forest
0.5925925925925926
             precision    recall  f1-score   support

          0       0.88      0.41      0.56        17
          1       0.47      0.90      0.62        10

avg / total       0.73      0.59      0.58        27
```

```
Neural Net
0.9259259259259259
             precision    recall  f1-score   support

          0       1.00      0.88      0.94        17
          1       0.83      1.00      0.91        10

avg / total       0.94      0.93      0.93        27

AdaBoost
0.8518518518518519
             precision    recall  f1-score   support

          0       1.00      0.76      0.87        17
          1       0.71      1.00      0.83        10

avg / total       0.89      0.85      0.85        27

Naive Bayes
0.9259259259259259
             precision    recall  f1-score   support

          0       1.00      0.88      0.94        17
          1       0.83      1.00      0.91        10

avg / total       0.94      0.93      0.93        27

SVM Linear
0.9629629629629629
             precision    recall  f1-score   support

          0       1.00      0.94      0.97        17
          1       0.91      1.00      0.95        10

avg / total       0.97      0.96      0.96        27

 SVM RBF
 0.7777777777777778
              precision    recall  f1-score   support

           0       1.00      0.65      0.79        17
           1       0.62      1.00      0.77        10

 avg / total       0.86      0.78      0.78        27

 SVM Sigmoid
 0.4444444444444444
              precision    recall  f1-score   support

           0       1.00      0.12      0.21        17
           1       0.40      1.00      0.57        10

 avg / total       0.78      0.44      0.34        27
```

41) فحص اضطراب طيف التوحد في مرحلة الطفولة باستخدام التعلم الآلي Childhood Autistic Spectrum Disorder Screening using Machine Learning

يمكن أن يؤدي التشخيص المبكر لاضطرابات النمو العصبي إلى تحسين العلاج وتقليل تكاليف الرعاية الصحية المرتبطة به بشكل كبير.في هذا المشروع، سوف نستخدم التعلم الخاضع للإشراف لتشخيص اضطراب طيف التوحد Autistic Spectrum Disorder (ASD) بناءً على السمات السلوكية والخصائص الفردية. وبشكل أكثر تحديداً، سنقوم ببناء ونشر شبكة عصبية باستخدام Keras API.

سيستخدم هذا المشروع مجموعة بيانات مقدمة من UCI Machine Learning Repository التي تحتوي على بيانات فحص لـ 292 مريضًا. يمكن العثور على مجموعة البيانات من هنا.

دعنا نتعمقفي! أولاً، سنقوم باستيراد بعض المكتبات التي سنستخدمهافي هذا المشروع.

```
import sys
import pandas as pd
import sklearn
import keras

print 'Python: {}'.format(sys.version)
print 'Pandas: {}'.format(pd.__version__)
print 'Sklearn: {}'.format(sklearn.__version__)
print 'Keras: {}'.format(keras.__version__)
```

```
Using Theano backend.
WARNING (theano.tensor.blas): Using NumPy C-API based implementation f
or BLAS functions.
Python: 2.7.13 |Continuum Analytics, Inc.| (default, May 11 2017, 13:1
7:26) [MSC v.1500 64 bit (AMD64)]
Pandas: 0.21.0
Sklearn: 0.19.1
Keras: 2.1.4
```

1. استيراد مجموعة البيانات

سنحصل على البيانات من UCI Machine Learning Repository؛ ومع ذلك، نظرًا لأن البيانات غير موجودةفي ملف csv أو txt، فسنضطر إلى تنزيل ملف zip المضغوط ثم استخراج البيانات يدويًا. بمجرد الانتهاء من ذلك، سنقرأ المعلومات الواردة من ملف نصي باستخدام Pandas.

- استيراد مجموعة البيانات.

```
# import the dataset
file = 'C:/users/brend/tutorial/autism-data.txt'

# read the csv
data = pd.read_table(file, sep = ',', index_col = None)
```

- اطبع شكل DataFrame، حتى نتمكن من رؤية عدد الأمثلة التي لدينا.

```
# print the shape of the DataFrame, so we can see how many examples we
have
print 'Shape of DataFrame: {}'.format(data.shape)
print data.loc[0]
```

```
Shape of DataFrame: (292, 21)
A1_Score                          1
A2_Score                          1
A3_Score                          0
A4_Score                          0
A5_Score                          1
A6_Score                          1
A7_Score                          0
A8_Score                          1
A9_Score                          0
A10_Score                         0
age                               6
gender                            m
ethnicity                    Others
jundice                          no
family_history_of_PDD            no
contry_of_res                Jordan
used_app_before                  no
result                            5
age_desc                '4-11 years'
relation                       Parent
class                            NO
Name: 0, dtype: object
```

- اطبع عدة مرضىفي نفس الوقت.

```
# print out multiple patients at the same time
data.loc[:10]
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	...	gender	ethnicity	jundice	family_history_of_PDD	contry_of_res
0	1	1	0	0	1	1	0	1	0	0	...	m	Others	no	no	Jordan
1	1	1	0	0	1	1	0	1	0	0	...	m	'Middle Eastern '	no	no	Jordan
2	1	1	0	0	0	1	1	1	0	0	...	m	?	no	no	Jordan
3	0	1	0	0	1	1	0	0	0	1	...	f	?	yes	no	Jordan
4	1	1	1	1	1	1	1	1	1	1	...	m	Others	yes	no	'United States'
5	0	0	1	0	1	1	0	1	0	1	...	m	?	no	yes	Egypt
6	1	0	1	1	1	1	0	1	0	1	...	m	White-European	no	no	'United Kingdom'
7	1	1	1	1	1	1	1	1	0	0	...	f	'Middle Eastern '	no	no	Bahrain
8	1	1	1	1	1	1	1	0	0	0	...	f	'Middle Eastern '	no	no	Bahrain
9	0	0	1	1	1	0	1	1	0	0	...	f	?	no	yes	Austria
10	1	0	0	0	1	1	1	1	1	1	...	m	White-European	yes	no	'United Kingdom'

11 rows × 21 columns

- اطبع وصفًا لإطار البيانات.

```
# print out a description of the dataframe
data.describe()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	result
count	292.000000	292.000000	292.000000	292.000000	292.000000	292.000000	292.000000	292.000000	292.000000	292.000000	292.000000
mean	0.633562	0.534247	0.743151	0.551370	0.743151	0.712329	0.606164	0.496575	0.493151	0.726027	6.239726
std	0.482658	0.499682	0.437646	0.498208	0.437646	0.453454	0.489438	0.500847	0.500811	0.446761	2.284882
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5.000000
50%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	1.000000	6.000000
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	8.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	10.000000

2. المعالجة المسبقة للبيانات

ستتطلب مجموعة البيانات هذه خطوات معالجة مسبقة متعددة. أولاً، لدينا أعمدةفي DataFrame (السمات) التي لا نريد استخدامها عند تدريب شبكتنا العصبية. سنقوم بإسقاط هذه الأعمدة أولاً. ثانيًا، يتم الإبلاغ عن الكثير من بياناتنا باستخدام السلاسل النصية strings؛ نتيجة لذلك، سنقوم بتحويل بياناتنا إلى تسميات فئوية categorical labels. أثناء المعالجة المسبقة، سنقسم مجموعة البيانات أيضًا إلى مجموعتي بيانات X وY، حيث يحتوي X على جميع السمات التي نريد استخدامها للتنبؤ وY به تسميات الفئات.

- إسقاط الاعمدة غير المرغوب فيها.

```
# drop unwanted columns
data = data.drop(['result', 'age_desc'], axis=1)
```

```
data.loc[:10]
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity	jundice	family_history_of_PDD	contry_of_res
0	1	1	0	0	1	1	0	1	0	0	6	m	Others	no	no	Jordan
1	1	1	0	0	1	1	0	1	0	0	6	m	'Middle Eastern '	no	no	Jordan
2	1	1	0	0	0	1	1	1	0	0	6	m	?	no	no	Jordan
3	0	1	0	0	1	1	0	0	0	1	5	f	?	yes	no	Jordan
4	1	1	1	1	1	1	1	1	1	1	5	m	Others	yes	no	'United States'
5	0	0	1	0	1	1	0	1	0	1	4	m	?	no	yes	Egypt
6	1	0	1	1	1	1	0	1	0	1	5	m	White-European	no	no	'United Kingdom'
7	1	1	1	1	1	1	1	1	0	0	5	f	'Middle Eastern '	no	no	Bahrain
8	1	1	1	1	1	1	1	0	0	0	11	f	'Middle Eastern '	no	no	Bahrain
9	0	0	1	1	1	0	1	1	0	0	11	f	?	no	yes	Austria
10	1	0	0	0	1	1	1	1	1	1	10	m	White-European	yes	no	'United Kingdom'

- إنشاء مجموعات بيانات X و Y للتدريب.

```
# create X and Y datasets for training
x = data.drop(['class'], 1)
y = data['class']
```

```
x.loc[:10]
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity	jundice	family_history_of_PDD	contry_of_res
0	1	1	0	0	1	1	0	1	0	0	6	m	Others	no	no	Jordar
1	1	1	0	0	1	1	0	1	0	0	6	m	'Middle Eastern '	no	no	Jordar
2	1	1	0	0	0	1	1	1	0	0	6	m	?	no	no	Jordar
3	0	1	0	0	1	1	0	0	0	1	5	f	?	yes	no	Jordar
4	1	1	1	1	1	1	1	1	1	1	5	m	Others	yes	no	'United States
5	0	0	1	0	1	1	0	1	0	1	4	m	?	no	yes	Egypt
6	1	0	1	1	1	1	0	1	0	1	5	m	White-European	no	no	'United Kingdom
7	1	1	1	1	1	1	1	1	0	0	5	f	'Middle Eastern '	no	no	Bahrair
8	1	1	1	1	1	1	1	0	0	0	11	f	'Middle Eastern '	no	no	Bahrair
9	0	0	1	1	1	0	1	1	0	0	11	f	?	no	yes	Austria
10	1	0	0	0	1	1	1	1	1	1	10	m	White-European	yes	no	'United Kingdom

- تحويل البيانات إلى قيم فئوية ـ متجهات ذات ترميز واحد ساخن one-hot- encoded.

```
# convert the data to categorical values - one-hot-encoded vectors
X = pd.get_dummies(x)
```

- طباعة تسميات الأعمدة الفئوية الجديدة.

```
# print the new categorical column labels
X.columns.values
```

```
array(['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score',
       'A6_Score', 'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score',
       'age_10', 'age_11', 'age_4', 'age_5', 'age_6', 'age_7', 'age_8',
       'age_9', 'age_?', 'gender_f', 'gender_m',
       "ethnicity_'Middle Eastern '", "ethnicity_'South Asian'",
       'ethnicity_?', 'ethnicity_Asian', 'ethnicity_Black',
       'ethnicity_Hispanic', 'ethnicity_Latino', 'ethnicity_Others',
       'ethnicity_Pasifika', 'ethnicity_Turkish',
       'ethnicity_White-European', 'jundice_no', 'jundice_yes',
       'family_history_of_PDD_no', 'family_history_of_PDD_yes',
       "contry_of_res_'Costa Rica'", "contry_of_res_'Isle of Man'",
       "contry_of_res_'New Zealand'", "contry_of_res_'Saudi Arabia'",
       "contry_of_res_'South Africa'", "contry_of_res_'South Korea'",
       "contry_of_res_'U.S. Outlying Islands'",
       "contry_of_res_'United Arab Emirates'",
       "contry_of_res_'United Kingdom'", "contry_of_res_'United States'",
       'contry_of_res_Afghanistan', 'contry_of_res_Argentina',
       'contry_of_res_Armenia', 'contry_of_res_Australia',
       'contry_of_res_Austria', 'contry_of_res_Bahrain',
       'contry_of_res_Bangladesh', 'contry_of_res_Bhutan',
       'contry_of_res_Brazil', 'contry_of_res_Bulgaria',
       'contry_of_res_Canada', 'contry_of_res_China',
       'contry_of_res_Egypt', 'contry_of_res_Europe',
       'contry_of_res_Georgia', 'contry_of_res_Germany',
       'contry_of_res_Ghana', 'contry_of_res_India', 'contry_of_res_Iraq'
       'contry_of_res_Ireland', 'contry_of_res_Italy',
       'contry_of_res_Japan', 'contry_of_res_Jordan',
       'contry_of_res_Kuwait', 'contry_of_res_Latvia',
       'contry_of_res_Lebanon', 'contry_of_res_Libya',
       'contry_of_res_Malaysia', 'contry_of_res_Malta',
       'contry_of_res_Mexico', 'contry_of_res_Nepal',
       'contry_of_res_Netherlands', 'contry_of_res_Nigeria',
       'contry_of_res_Oman', 'contry_of_res_Pakistan',
       'contry_of_res_Philippines', 'contry_of_res_Qatar',
       'contry_of_res_Romania', 'contry_of_res_Russia',
       'contry_of_res_Sweden', 'contry_of_res_Syria',
       'contry_of_res_Turkey', 'used_app_before_no',
       'used_app_before_yes', "relation_'Health care professional'",
       'relation_?', 'relation_Parent', 'relation_Relative',
       'relation_Self', 'relation_self'], dtype=object)
```

- اطبع مثالاً للمريض من البيانات الفئوية.

```
# print an example patient from the categorical data
X.loc[1]
```

```
A1_Score                                1
A2_Score                                1
A3_Score                                0
A4_Score                                0
A5_Score                                1
A6_Score                                1
A7_Score                                0
A8_Score                                1
A9_Score                                0
A10_Score                               0
age_10                                  0
age_11                                  0
age_4                                   0
age_5                                   0
age_6                                   1
age_7                                   0
age_8                                   0
age_9                                   0
age_?                                   0
gender_f                                0
gender_m                                1
ethnicity_'Middle Eastern '             1
ethnicity_'South Asian'                 0
ethnicity_?                             0
ethnicity_Asian                         0
ethnicity_Black                         0
ethnicity_Hispanic                      0
ethnicity_Latino                        0
ethnicity_Others                        0
ethnicity_Pasifika                      0
                                       ..
contry_of_res_Italy                     0
contry_of_res_Japan                     0
contry_of_res_Jordan                    1
contry_of_res_Kuwait                    0
contry_of_res_Latvia                    0
contry_of_res_Lebanon                   0
contry_of_res_Libya                     0
contry_of_res_Malaysia                  0
contry_of_res_Malta                     0
contry_of_res_Mexico                    0
contry_of_res_Nepal                     0
contry_of_res_Netherlands               0
contry_of_res_Nigeria                   0
contry_of_res_Oman                      0
contry_of_res_Pakistan                  0
contry_of_res_Philippines               0
contry_of_res_Qatar                     0
contry_of_res_Romania                   0
contry_of_res_Russia                    0
contry_of_res_Sweden                    0
contry_of_res_Syria                     0
contry_of_res_Turkey                    0
used_app_before_no                      1
used_app_before_yes                     0
relation_'Health care professional'     0
relation_?                              0
relation_Parent                         1
relation_Relative                       0
```

```
relation_Self                           0
relation_self                           0
Name: 1, Length: 96, dtype: int64
```

- تحويل بيانات الفئة إلى قيم فئوية ـ متجهات ذات ترميز واحد ساخن.

```
# convert the class data to categorical values - one-hot-encoded
vectors
Y = pd.get_dummies(y)
```

```
Y.iloc[:10]
```

	NO	YES
0	1	0
1	1	0
2	1	0
3	1	0
4	0	1
5	1	0
6	0	1
7	0	1
8	0	1
9	1	0

3. تقسيم مجموعة البيانات إلى مجموعات بيانات تدريب واختبار

قبل أن نبدأفي تدريب شبكتنا العصبية، نحتاج إلى تقسيم مجموعة البيانات إلى مجموعات بيانات تدريب واختبار. سيسمح لنا ذلك باختبار شبكتنا بعد أن ننتهي من التدريب لتحديد مدى نجاحها في التعميم على البيانات الجديدة. هذه الخطوة سهلة للغاية عند استخدام الدالة train_test_split() التي يوفرها scikit-Learn!

```
from sklearn import model_selection
# split the X and Y data into training and testing datasets
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X,
Y, test_size = 0.2)
```

```
print X_train.shape
print X_test.shape
print Y_train.shape
print Y_test.shape
```

```
(233, 96)
(59, 96)
(233, 2)
(59, 2)
```

4. بناء الشبكة - كيراس

في هذا المشروع، سنستخدم Keras لبناء شبكتنا وتدريبها. سيكون هذا النموذج بسيطًا نسبيًا وسيستخدم فقط طبقات كثيفة (تُعرف أيضًا باسم متصلة بالكامل). هذه هي طبقة الشبكة العصبية الأكثر شيوعًا. ستحتوي الشبكة على طبقة مخفية واحدة، وتستخدم مُحسِّن آدم Adam optimizer، وخسارة انتروبي متقاطعة categorical crossentropy loss. لن نقلق بشأن تحسين المعلمات مثل معدل التعلم أو عدد الخلايا العصبيةفي كل طبقة أو دوال التنشيطفي هذا

المشروع؛ ومع ذلك، إذا كان لديك الوقت، فإن ضبط هذه المعلمات يدويًا ومراقبة النتائج يعد طريقة رائعة للتعرف على وظيفتها!

- بناء شبكة عصبية باستخدام Keras.

```
# build a neural network using Keras
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam

# define a function to build the keras model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim=96, kernel_initializer='normal',
activation='relu'))
    model.add(Dense(4, kernel_initializer='normal',
activation='relu'))
    model.add(Dense(2, activation='sigmoid'))

    # compile model
    adam = Adam(lr=0.001)
    model.compile(loss='categorical_crossentropy', optimizer=adam,
metrics=['accuracy'])
    return model

model = create_model()

print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 8)                 776
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 36
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 10
=================================================================
Total params: 822
Trainable params: 822
Non-trainable params: 0
_________________________________________________________________
None
```

5. تدريب الشبكة

حان وقت المرح الآن! تدريب نموذج Keras بسيط مثل استدعاء model.fit().

- تدريب النموذج مع بيانات التدريب.

```
# fit the model to the training data
model.fit(X_train, Y_train, epochs=50, batch_size=10, verbose = 1)
Epoch 1/50
233/233 [==============================] - 0s 288us/step - loss: 0.692
7 - acc: 0.5794
Epoch 2/50
233/233 [==============================] - 0s 245us/step - loss: 0.691
0 - acc: 0.7210
Epoch 3/50
```

```
233/233 [==============================] - 0s 258us/step - loss: 0.686
8 - acc: 0.7639
Epoch 4/50
233/233 [==============================] - 0s 236us/step - loss: 0.677
9 - acc: 0.7082
Epoch 5/50
233/233 [==============================] - 0s 236us/step - loss: 0.661
9 - acc: 0.8541
Epoch 6/50
233/233 [==============================] - 0s 305us/step - loss: 0.634
0 - acc: 0.8283
Epoch 7/50
233/233 [==============================] - 0s 227us/step - loss: 0.596
3 - acc: 0.8541
Epoch 8/50
233/233 [==============================] - 0s 305us/step - loss: 0.544
6 - acc: 0.9399
Epoch 9/50
233/233 [==============================] - 0s 240us/step - loss: 0.488
4 - acc: 0.8884
Epoch 10/50
233/233 [==============================] - 0s 227us/step - loss: 0.422
0 - acc: 0.9227
Epoch 11/50
233/233 [==============================] - 0s 322us/step - loss: 0.360
3 - acc: 0.9313
Epoch 12/50
233/233 [==============================] - 0s 245us/step - loss: 0.293
5 - acc: 0.9614
Epoch 13/50
233/233 [==============================] - 0s 296us/step - loss: 0.252
8 - acc: 0.9657
Epoch 14/50
233/233 [==============================] - 0s 330us/step - loss: 0.208
7 - acc: 0.9657
Epoch 15/50
233/233 [==============================] - 0s 305us/step - loss: 0.178
8 - acc: 0.9871
Epoch 16/50
233/233 [==============================] - 0s 313us/step - loss: 0.160
5 - acc: 0.9700
Epoch 17/50
233/233 [==============================] - 0s 309us/step - loss: 0.138
9 - acc: 0.9828
Epoch 18/50
233/233 [==============================] - 0s 335us/step - loss: 0.125
8 - acc: 0.9785
Epoch 19/50
233/233 [==============================] - 0s 343us/step - loss: 0.110
8 - acc: 0.9871
Epoch 20/50
233/233 [==============================] - 0s 399us/step - loss: 0.100
4 - acc: 0.9871
Epoch 21/50
233/233 [==============================] - 0s 416us/step - loss: 0.091
0 - acc: 0.9871
Epoch 22/50
233/233 [==============================] - 0s 343us/step - loss: 0.082
0 - acc: 0.9871
Epoch 23/50
233/233 [==============================] - 0s 361us/step - loss: 0.075
2 - acc: 0.9914
Epoch 24/50
```

```
233/233 [==============================] - 0s 356us/step - loss: 0.071
4 - acc: 0.9957
Epoch 25/50
233/233 [==============================] - 0s 309us/step - loss: 0.063
4 - acc: 0.9957
Epoch 26/50
233/233 [==============================] - 0s 339us/step - loss: 0.058
5 - acc: 0.9957
Epoch 27/50
233/233 [==============================] - 0s 335us/step - loss: 0.057
1 - acc: 1.0000
Epoch 28/50
233/233 [==============================] - 0s 429us/step - loss: 0.052
6 - acc: 0.9957
Epoch 29/50
233/233 [==============================] - 0s 335us/step - loss: 0.047
4 - acc: 1.0000
Epoch 30/50
233/233 [==============================] - 0s 322us/step - loss: 0.046
3 - acc: 0.9957
Epoch 31/50
233/233 [==============================] - 0s 296us/step - loss: 0.043
1 - acc: 1.0000
Epoch 32/50
233/233 [==============================] - 0s 348us/step - loss: 0.038
1 - acc: 1.0000
Epoch 33/50
233/233 [==============================] - 0s 322us/step - loss: 0.035
7 - acc: 1.0000
Epoch 34/50
233/233 [==============================] - 0s 292us/step - loss: 0.033
1 - acc: 1.0000
Epoch 35/50
233/233 [==============================] - 0s 305us/step - loss: 0.031
6 - acc: 1.0000
Epoch 36/50
233/233 [==============================] - 0s 335us/step - loss: 0.029
4 - acc: 1.0000
Epoch 37/50
233/233 [==============================] - 0s 322us/step - loss: 0.028
2 - acc: 1.0000
Epoch 38/50
233/233 [==============================] - 0s 236us/step - loss: 0.028
1 - acc: 1.0000
Epoch 39/50
233/233 [==============================] - 0s 339us/step - loss: 0.025
3 - acc: 1.0000
Epoch 40/50
233/233 [==============================] - 0s 223us/step - loss: 0.025
2 - acc: 1.0000
Epoch 41/50
233/233 [==============================] - 0s 326us/step - loss: 0.022
6 - acc: 1.0000
Epoch 42/50
233/233 [==============================] - 0s 326us/step - loss: 0.021
3 - acc: 1.0000
Epoch 43/50
233/233 [==============================] - 0s 219us/step - loss: 0.020
3 - acc: 1.0000
Epoch 44/50
233/233 [==============================] - 0s 215us/step - loss: 0.019
3 - acc: 1.0000
Epoch 45/50
```

```
233/233 [==============================] - 0s 318us/step - loss: 0.019
0 - acc: 1.0000
Epoch 46/50
233/233 [==============================] - 0s 232us/step - loss: 0.017
6 - acc: 1.0000
Epoch 47/50
233/233 [==============================] - 0s 215us/step - loss: 0.016
3 - acc: 1.0000
Epoch 48/50
233/233 [==============================] - 0s 202us/step - loss: 0.016
1 - acc: 1.0000
Epoch 49/50
233/233 [==============================] - 0s 240us/step - loss: 0.015
4 - acc: 1.0000
Epoch 50/50
233/233 [==============================] - 0s 223us/step - loss: 0.015
0 - acc: 1.0000
```

6. مقاييس الاختبار والأداء

الآن وقد تم تدريب نموذجنا، نحتاج إلى اختبار أدائه على مجموعة بيانات الاختبار. النموذج لم ير هذه المعلومات من قبل؛ ونتيجة لذلك، تتيح لنا مجموعة بيانات الاختبار تحديد ما إذا كان النموذج سيكون قادرًا على التعميم على المعلومات التي لم يتم استخدامها أثناء مرحلة التدريب أم لا. سنستخدم بعض المقاييس التي يوفرها موقع scikit-Learn لهذا الغرض!

- إنشاء تقرير التصنيف باستخدام التنبؤات للنموذج الفئوي

```
# generate classification report using predictions for categorical
model
from sklearn.metrics import classification_report, accuracy_score

predictions = model.predict_classes(X_test)
predictions
```

```
array([1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0,
1,
       1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1,
0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0], dtype=int64)
```

```
print('Results for Categorical Model')
print(accuracy_score(Y_test[['YES']], predictions))
print(classification_report(Y_test[['YES']], predictions))
```

```
Results for Categorical Model
0.9661016949152542
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        36
           1       0.96      0.96      0.96        23

avg / total       0.97      0.97      0.97        59
```

42) التنبؤ بهطول الأمطار باستخدام التعلم الآلي Rainfall Prediction using Machine Learning

يعد التنبؤ بهطول الأمطار إحدى المهام الصعبة وغير المؤكدة التي لها تأثير كبير على المجتمع البشري. يمكن للتنبؤ الدقيق وفي الوقت المناسب أن يساعد بشكل استباقيفي تقليل الخسائر البشرية والمالية. تقدم هذه الدراسة مجموعة من التجارب التي تتضمن استخدام تقنيات التعلم الآلي الشائعة لإنشاء نماذج يمكنها التنبؤ بما إذا كانت ستمطر غداً أم لا بناءً على بيانات الطقس لذلك اليومفي المدن الكبرىفي أستراليا.

لطالما أحببت معرفة المعلمات التي يأخذها خبراء الأرصاد الجويةفي الاعتبار قبل وضع توقعات الطقس، لذلك وجدت مجموعة البيانات مثيرة للاهتمام. ومع ذلك، من وجهة نظر الخبير، فإن مجموعة البيانات هذه واضحة إلى حد ما.في نهاية هذا المقال سوف تتعلم:

- كيف يتم إجراء التوازن لمجموعة بيانات غير متوازنة.
- كيف يتم ترميز التسميات للمتغيرات الفئوية.
- كيف يتم استخدام ضمني معقد sophisticated imputation مثل MICE.
- كيف يمكن الكشف عن القيم المتطرفة outliers واستبعادها من البيانات.
- كيف يتم استخدام طريقة التصفية filter method وطريقة الغلاف wrapper methods لاختيار الميزة.
- كيفية مقارنة السرعة والأداء لنماذج شهيرة مختلفة.
- أي مقياس يمكن أن يكون الأفضل للحكم على الأداء على مجموعة بيانات غير متوازنة: الدقة ودرجة F1.

لنبدأ مهمة التنبؤ بهطول الأمطار عن طريق استيراد البيانات، يمكنك تنزيل مجموعة البيانات التي أستخدمهافي هذه المهمة من هنا:

```
import pandas as pd
from google.colab import files
uploaded = files.upload()
full_data = pd.read_csv('weatherAUS.csv')
full_data.head()
```

استكشاف البيانات

سوف نتحقق أولاً من عدد الصفوف والأعمدة. بعد ذلك، سنتحقق من حجم مجموعة البيانات لتحديد ما إذا كانت بحاجة إلى ضغط الحجم.

```
full_data.shape
```

(142193, 24)

```
full_data.info()
```

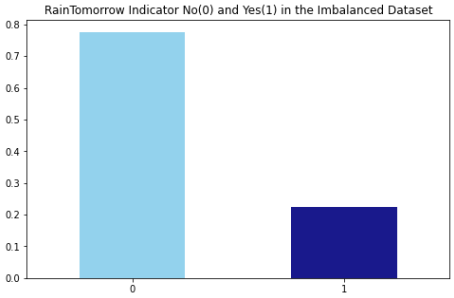
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 142193 entries, 0 to 142192
Data columns (total 24 columns):
 0   Date           142193 non-null  object
 1   Location       142193 non-null  object
 2   MinTemp        141556 non-null  float64
 3   MaxTemp        141871 non-null  float64
 4   Rainfall       140787 non-null  float64
 5   Evaporation    81350 non-null   float64
 6   Sunshine       74377 non-null   float64
 7   WindGustDir    132863 non-null  object
 8   WindGustSpeed  132923 non-null  float64
 9   WindDir9am     132180 non-null  object
 10  WindDir3pm     138415 non-null  object
 11  WindSpeed9am   140845 non-null  float64
 12  WindSpeed3pm   139563 non-null  float64
 13  Humidity9am    140419 non-null  float64
 14  Humidity3pm    138583 non-null  float64
 15  Pressure9am    128179 non-null  float64
 16  Pressure3pm    128212 non-null  float64
 17  Cloud9am       88536 non-null   float64
 18  Cloud3pm       85099 non-null   float64
 19  Temp9am        141289 non-null  float64
 20  Temp3pm        139467 non-null  float64
 21  RainToday      140787 non-null  object
 22  RISK_MM        142193 non-null  float64
 23  RainTomorrow   142193 non-null  object
dtypes: float64(17), object(7)
memory usage: 26.0+ MB
```

"RainToday" و "RainTomorrow" هما كائنان (نعم / لا). سوف أقوم بتحويلها إلى ثنائي (0/1) لراحتنا.

```
full_data['RainToday'].replace({'No': 0, 'Yes': 1},inplace = True)
full_data['RainTomorrow'].replace({'No': 0, 'Yes': 1},inplace = True)
```

بعد ذلك، سوف نتحقق مما إذا كانت مجموعة البيانات غير متوازنة أو متوازنة. إذا كانت مجموعة البيانات غير متوازنة، فنحن بحاجة إما إلى تقليص حجم downsample الأغلبية majority أو الإفراط في أخذ عينة oversample من الأقلية minority لموازنتها.

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (8,5))
full_data.RainTomorrow.value_counts(normalize = True).plot(kind='bar',
color= ['skyblue','navy'], alpha = 0.9, rot=0)
plt.title('RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced
Dataset')
plt.show()
```

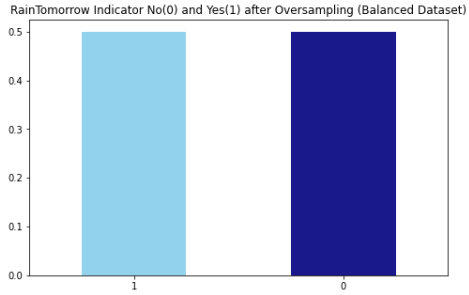


يمكننا أن نلاحظ أن وجود "0" و "1" يكاد يكون في نسبة 78:22. إذن هناك خلل طبقي وعلينا أن نتعامل معه. لمحاربة عدم توازن الفئة class imbalance، سنستخدم هنا الإفراط في أخذ عينة

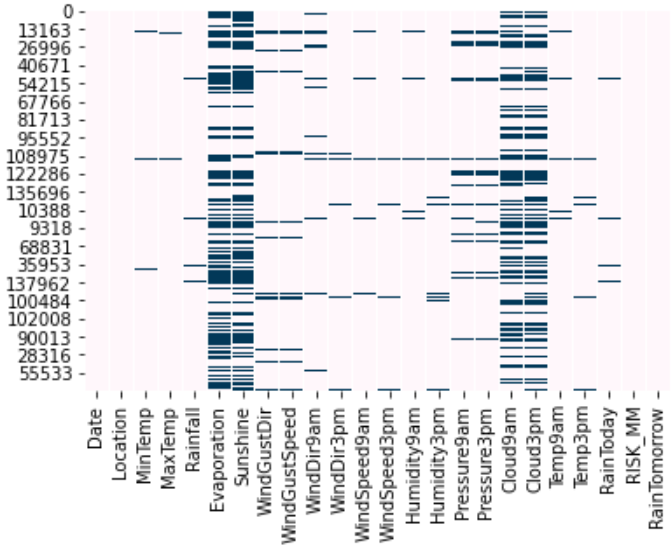
من طبقة الأقلية. نظرًا لأن حجم مجموعة البيانات صغير جدًا، فإن أخذ العينات الفرعية لفئة الأغلبية لن يكون له معنى كبير هنا.

معالجة عدم توازن الفئة للتنبؤ بهطول الأمطار

```
from sklearn.utils import resample

no = full_data[full_data.RainTomorrow == 0]
yes = full_data[full_data.RainTomorrow == 1]
yes_oversampled = resample(yes, replace=True, n_samples=len(no),
random_state=123)
oversampled = pd.concat([no, yes_oversampled])

fig = plt.figure(figsize = (8,5))
oversampled.RainTomorrow.value_counts(normalize =
True).plot(kind='bar', color= ['skyblue','navy'], alpha = 0.9, rot=0)
plt.title('RainTomorrow Indicator No(0) and Yes(1) after Oversampling
(Balanced Dataset)')
plt.show()
```



الآن، سوف أتحقق من نموذج البيانات المفقود missing dataفي مجموعة البيانات:



من الواضح أن "Evaporation" و "Sunshine" و "Cloud9am" و "Cloud3pm" هي ميزات ذات نسبة عالية مفقودة. لذلك سوف نتحقق من تفاصيل البيانات المفقودة لهذه الميزات الأربع.

```
total = oversampled.isnull().sum().sort_values(ascending=False)
percent =
(oversampled.isnull().sum()/oversampled.isnull().count()).sort_values(
ascending=False)
missing = pd.concat([total, percent], axis=1, keys=['Total',
'Percent'])
missing.head(4)
```

	Total	Percent
Sunshine	104831	0.475140
Evaporation	95411	0.432444
Cloud3pm	85614	0.388040
Cloud9am	81339	0.368664

نلاحظ أن الميزات الأربع تحتوي على بيانات مفقودة بنسبة أقل من 50في المائة. لذا بدلاً من رفضها تمامًا، سنأخذهافي الاعتبارفي نموذجنا مع التضمين imputation المناسب.

التضمين والتحول

سنحسب الأعمدة الفئوية مع الوضع، ثم سنستخدم مشفر التسمية label encoder لتحويلها إلى أرقام رقمية. بمجرد تحويل جميع الأعمدةفي إطار البيانات الكامل إلى أعمدة رقمية، سنحسب القيم المفقودة باستخدام حزمة المحاكاة المتعددة Multiple Imputation by Chained Equations (MICE).

ثم سنكتشف القيم المتطرفة باستخدام النطاق الربعي interquartile range ونزيلها للحصول على مجموعة بيانات العمل النهائية. أخيرًا، سوف نتحقق من الارتباط بين المتغيرات المختلفة، وإذا وجدنا زوجًا من المتغيرات شديدة الارتباط، فسوف نتجاهل أحدهما مع الاحتفاظ بالآخر.

```
oversampled.select_dtypes(include=['object']).columns
```

```
Index(['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm'], dtype='object')
```

```
# Impute categorical var with Mode
oversampled['Date'] =
oversampled['Date'].fillna(oversampled['Date'].mode()[0])
oversampled['Location'] =
oversampled['Location'].fillna(oversampled['Location'].mode()[0])
oversampled['WindGustDir'] =
oversampled['WindGustDir'].fillna(oversampled['WindGustDir'].mode()[0]
)
oversampled['WindDir9am'] =
oversampled['WindDir9am'].fillna(oversampled['WindDir9am'].mode()[0])
oversampled['WindDir3pm'] =
oversampled['WindDir3pm'].fillna(oversampled['WindDir3pm'].mode()[0])
```

```
# Convert categorical features to continuous features with Label
Encoding
from sklearn.preprocessing import LabelEncoder
lencoders = {}
for col in oversampled.select dtypes(include=['object']).columns:
    lencoders[col] = LabelEncoder()
    oversampled[col] = lencoders[col].fit_transform(oversampled[col])
```

```
import warnings
warnings.filterwarnings("ignore")
# Multiple Imputation by Chained Equations
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
MiceImputed = oversampled.copy(deep=True)
mice_imputer = IterativeImputer()
MiceImputed.iloc[:, :] = mice_imputer.fit_transform(oversampled)
```

وبالتالي، فإن إطار البيانات ليس له قيمة "NaN". سنكتشف الآن القيم المتطرفة والقضاء عليها من مجموعة البيانات الفاصلة بين الشرائح الربعية inter-quartile interval-based data set.

```
# Detecting outliers with IQR
Q1 = MiceImputed.quantile(0.25)
Q3 = MiceImputed.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

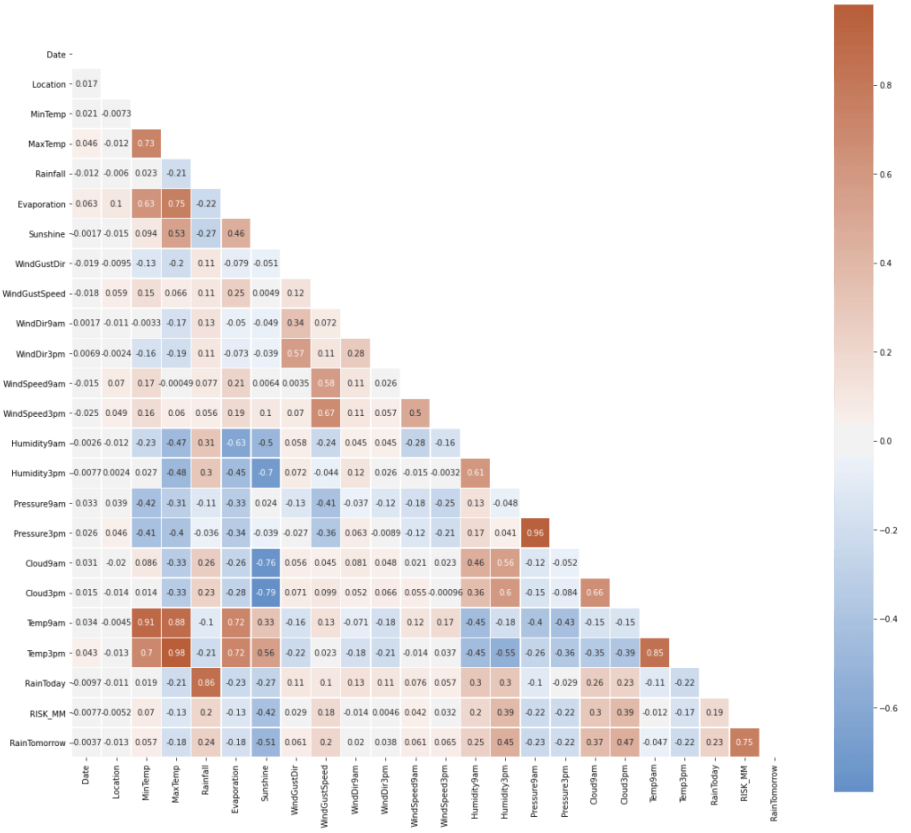
```
Date            1535.000000
Location          25.000000
MinTemp            9.300000
MaxTemp           10.200000
Rainfall           2.400000
Evaporation        4.119679
Sunshine           5.947404
WindGustDir        9.000000
WindGustSpeed     19.000000
WindDir9am         8.000000
WindDir3pm         8.000000
WindSpeed9am      13.000000
WindSpeed3pm      11.000000
Humidity9am       26.000000
Humidity3pm       30.000000
Pressure9am        8.800000
Pressure3pm        8.800000
Cloud9am           4.000000
Cloud3pm           3.681346
Temp9am            9.300000
Temp3pm            9.800000
RainToday          1.000000
RISK_MM            5.200000
RainTomorrow       1.000000
dtype: float64
```

```
# Removing outliers from the dataset
MiceImputed = MiceImputed[~((MiceImputed < (Q1 - 1.5 * IQR))
|(MiceImputed > (Q3 + 1.5 * IQR))).any(axis=1)]
MiceImputed.shape
```

```
(156852, 24)
```

نلاحظ أن مجموعة البيانات الأصلية كانت بالشكل (87927, 24). بعد تشغيل مقتطف الشفرة لإزالة القيم المتطرفة، أصبحت مجموعة البيانات الآن بالشكل (86065, 24). نتيجة لذلك، أصبحت مجموعة البيانات الآن خالية من 1862 قيمة متطرفة. سنقوم الآن بفحص العلاقة الخطية المتعددة، أي ما إذا كان الحرف مرتبطة بقوة بأخرى.

```
# Correlation Heatmap
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
corr = MiceImputed.corr()
mask = np.triu(np.ones_like(corr, dtype=np.bool))
f, ax = plt.subplots(figsize=(20, 20))
cmap = sns.diverging_palette(250, 25, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=None,
center=0,square=True, annot=True, linewidths=.5, cbar_kws={"shrink":
.9})
```

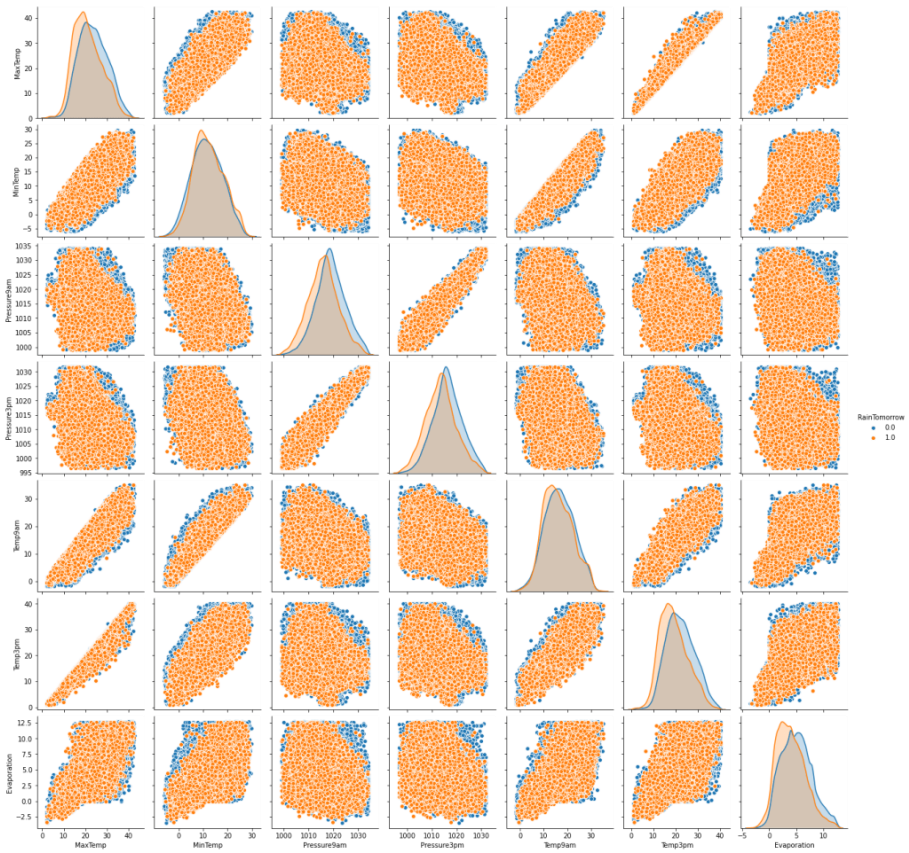


ترتبط أزواج الميزات التالية ارتباطًا وثيقًا ببعضها البعض:

- MaxTemp و MinTemp.
- Pressure9h وpressure3h .
- Temp9am و Temp3pm.
- Evaporation و MaxTemp.
- MaxTemp و Temp3pm لكن قيمة الارتباط لا تساوي بأي حال من الأحوال "1". لذلك نحن لا نزيل أي وظيفة

ومع ذلك، يمكننا التعمق في العلاقة الزوجية بين هذه الخصائص شديدة الارتباط من خلال فحص مخطط الزوج التالي. تُظهر كل قطعة من المخططات المزدوجة مجموعات مميزة بوضوح شديد من مجموعات RainTomorrow's "نعم" و "لا". هناك حد أدنى من التداخل بينهما.

```
sns.pairplot( data=MiceImputed,
vars=('MaxTemp','MinTemp','Pressure9am','Pressure3pm', 'Temp9am',
'Temp3pm', 'Evaporation'), hue='RainTomorrow' )
```



اختيار الميزة للتنبؤ بهطول الأمطار

سأستخدم كل من طريقة التصفية filter وطريقة wrapper لاختيار الميزة لتدريب نموذج التنبؤ بهطول الأمطار.

تحديد الميزات عن طريق طريقة التصفية (قيمة chi-square): قبل القيام بذلك، يجب أولاً تسوية بياناتنا. نستخدم MinMaxScaler بدلاً من StandardScaler لتجنب القيم السالبة.

```
# Standardizing data
from sklearn import preprocessing
r_scaler = preprocessing.MinMaxScaler()
r_scaler.fit(MiceImputed)
modified_data = pd.DataFrame(r_scaler.transform(MiceImputed),
index=MiceImputed.index, columns=MiceImputed.columns)
```

```
# Feature Importance using Filter Method (Chi-Square)
from sklearn.feature_selection import SelectKBest, chi2
X = modified_data.loc[:,modified_data.columns!='RainTomorrow']
y = modified_data[['RainTomorrow']]
selector = SelectKBest(chi2, k=10)
selector.fit(X, y)
X_new = selector.transform(X)
print(X.columns[selector.get_support(indices=True)])
```

```
Index(['Sunshine', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm',
       'Cloud9am', 'Cloud3pm', 'Temp3pm', 'RainToday', 'RISK_MM'],
      dtype='object')
```

يمكننا أن نلاحظ أن "Sunshine" ،"Humidity9am" ،"Humidity3am"، "Pressure9am"، "Pressure3pm" لها أهمية أعلى مقارنة بالميزات الأخرى.

اختيار المعالم بطريقة الالتفاف wrapping method (الغابة العشوائية random forest):

```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier as rf

X = MiceImputed.drop('RainTomorrow', axis=1)
y = MiceImputed['RainTomorrow']
selector = SelectFromModel(rf(n_estimators=100, random_state=0))
selector.fit(X, y)
support = selector.get_support()
features = X.loc[:,support].columns.tolist()
print(features)
print(rf(n_estimators=100,
random_state=0).fit(X,y).feature_importances_)
```

```
['Sunshine', 'Cloud3pm', 'RISK_MM']
[0.00205993 0.00215407 0.00259089 0.00367568 0.0102656  0.00252838
 0.05894157 0.00143001 0.00797518 0.00177178 0.00167654 0.0014278
 0.00187743 0.00760691 0.03091966 0.00830365 0.01193018 0.02113544
 0.04962418 0.00270103 0.00513723 0.00352198 0.76074491]
```

تدريب نموذج التنبؤ بهطول الأمطار بنماذج مختلفة

سنقسم مجموعة البيانات إلى مجموعات تدريب (75٪) واختبار (25٪) على التوالي لتدريب نموذج التنبؤ بهطول الأمطار. للحصول على أفضل النتائج، سنقوم بتوحيد بيانات X_train وX_test الخاصة بنا:

```
features = MiceImputed[['Location', 'MinTemp', 'MaxTemp', 'Rainfall',
'Evaporation', 'Sunshine', 'WindGustDir',
                        'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
                        'Humidity3pm', 'Pressure9am', 'Pressure3pm',
'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm',
                        'RainToday']]
target = MiceImputed['RainTomorrow']

# Split into test and train
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.25, random_state=12345)

# Normalize Features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

```
def plot_roc_cur(fper, tper):
    plt.plot(fper, tper, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
```

```
import time
from sklearn.metrics import accuracy_score, roc_auc_score,
cohen_kappa_score, plot_confusion_matrix, roc_curve,
classification_report
def run_model(model, X_train, y_train, X_test, y_test, verbose=True):
    t0=time.time()
    if verbose == False:
        model.fit(X_train,y_train, verbose=0)
    else:
        model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_pred)
    coh_kap = cohen_kappa_score(y_test, y_pred)
    time_taken = time.time()-t0
    print("Accuracy = {}".format(accuracy))
    print("ROC Area under Curve = {}".format(roc_auc))
    print("Cohen's Kappa = {}".format(coh_kap))
    print("Time taken = {}".format(time_taken))
    print(classification_report(y_test,y_pred,digits=5))

    probs = model.predict_proba(X_test)
    probs = probs[:, 1]
```

```
    fper, tper, thresholds = roc_curve(y_test, probs)
    plot_roc_cur(fper, tper)

    plot_confusion_matrix(model, X_test, y_test,cmap=plt.cm.Blues,
normalize = 'all')

    return model, accuracy, roc_auc, coh_kap, time_taken
```

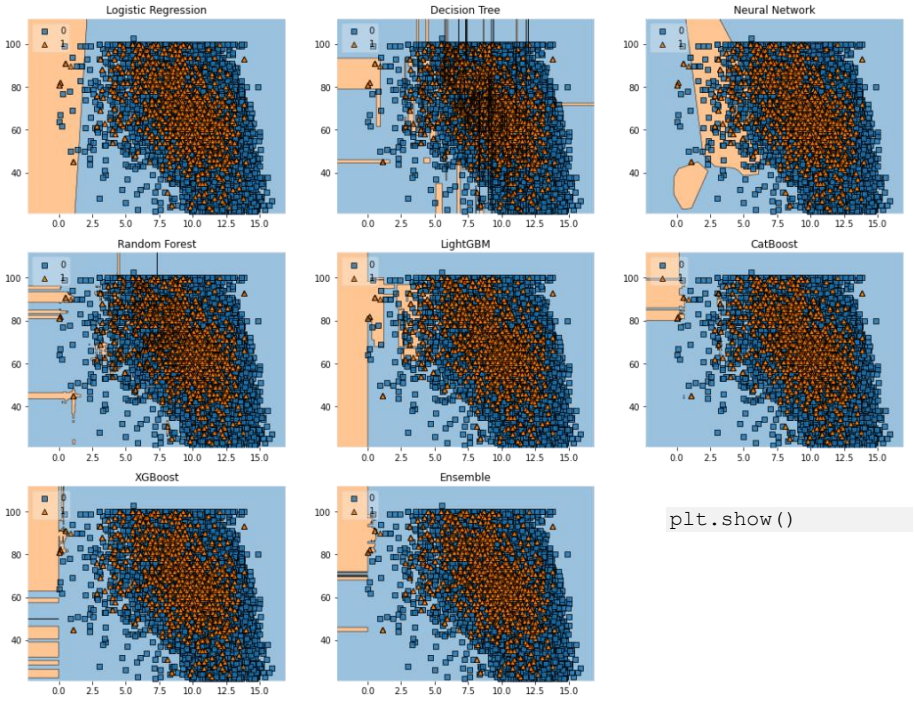
```
# Logistic Regression
from sklearn.linear_model import LogisticRegression

params_lr = {'penalty': 'l1', 'solver':'liblinear'}

model_lr = LogisticRegression(**params_lr)
model_lr, accuracy_lr, roc_auc_lr, coh_kap_lr, tt_lr =
run_model(model_lr, X_train, y_train, X_test, y_test)

# Decision Tree
from sklearn.tree import DecisionTreeClassifier

params_dt = {'max_depth': 16,
             'max_features': "sqrt"}

model_dt = DecisionTreeClassifier(**params_dt)
model_dt, accuracy_dt, roc_auc_dt, coh_kap_dt, tt_dt =
run_model(model_dt, X_train, y_train, X_test, y_test)

# Neural Network
from sklearn.neural_network import MLPClassifier

params_nn = {'hidden_layer_sizes': (30,30,30),
             'activation': 'logistic',
             'solver': 'lbfgs',
             'max_iter': 500}

model_nn = MLPClassifier(**params_nn)
model_nn, accuracy_nn, roc_auc_nn, coh_kap_nn, tt_nn =
run_model(model_nn, X_train, y_train, X_test, y_test)

# Random Forest
from sklearn.ensemble import RandomForestClassifier

params_rf = {'max_depth': 16,
             'min_samples_leaf': 1,
             'min_samples_split': 2,
             'n_estimators': 100,
             'random_state': 12345}

model_rf = RandomForestClassifier(**params_rf)
model_rf, accuracy_rf, roc_auc_rf, coh_kap_rf, tt_rf =
run_model(model_rf, X_train, y_train, X_test, y_test)

# Light GBM
import lightgbm as lgb
params_lgb ={'colsample_bytree': 0.95,
         'max_depth': 16,
         'min_split_gain': 0.1,
         'n_estimators': 200,
         'num_leaves': 50,
         'reg_alpha': 1.2,
         'reg_lambda': 1.2,
```

```
          'subsample': 0.95,
          'subsample_freq': 20}

model_lgb = lgb.LGBMClassifier(**params_lgb)
model lgb, accuracy lgb, roc auc lgb, coh kap lgb, tt lgb =
run_model(model_lgb, X_train, y_train, X_test, y_test)

# Catboost
!pip install catboost
import catboost as cb
params_cb ={'iterations': 50,
            'max_depth': 16}

model_cb = cb.CatBoostClassifier(**params_cb)
model_cb, accuracy_cb, roc_auc_cb, coh_kap_cb, tt_cb =
run_model(model_cb, X_train, y_train, X_test, y_test, verbose=False)

# XGBoost
import xgboost as xgb
params_xgb ={'n_estimators': 500,
            'max_depth': 16}

model_xgb = xgb.XGBClassifier(**params_xgb)
model_xgb, accuracy_xgb, roc_auc_xgb, coh_kap_xgb, tt_xgb =
run_model(model_xgb, X_train, y_train, X_test, y_test)
```

رسم منطقة القرار لكل النماذج

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import itertools
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
import lightgbm as lgb
import catboost as cb
import xgboost as xgb
from mlxtend.classifier import EnsembleVoteClassifier
from mlxtend.plotting import plot_decision_regions

value = 1.80
width = 0.90

clf1 = LogisticRegression(random_state=12345)
clf2 = DecisionTreeClassifier(random_state=12345)
clf3 = MLPClassifier(random_state=12345, verbose = 0)
clf4 = RandomForestClassifier(random_state=12345)
clf5 = lgb.LGBMClassifier(random_state=12345, verbose = 0)
clf6 = cb.CatBoostClassifier(random_state=12345, verbose = 0)
clf7 = xgb.XGBClassifier(random_state=12345)
eclf = EnsembleVoteClassifier(clfs=[clf4, clf5, clf6, clf7],
weights=[1, 1, 1, 1], voting='soft')

X_list = MiceImputed[["Sunshine", "Humidity9am", "Cloud3pm"]] #took
only really important features
X = np.asarray(X_list, dtype=np.float32)
y_list = MiceImputed["RainTomorrow"]
y = np.asarray(y_list, dtype=np.int32)

# Plotting Decision Regions
gs = gridspec.GridSpec(3,3)
fig = plt.figure(figsize=(18, 14))
```

```
labels = ['Logistic Regression',
          'Decision Tree',
          'Neural Network',
          'Random Forest',
          'LightGBM',
          'CatBoost',
          'XGBoost',
          'Ensemble']

for clf, lab, grd in zip([clf1, clf2, clf3, clf4, clf5, clf6, clf7,
eclf],
                         labels,
                         itertools.product([0, 1, 2],
                         repeat=2)):
    clf.fit(X, y)
    ax = plt.subplot(gs[grd[0], grd[1]])
    fig = plot_decision_regions(X=X, y=y, clf=clf,
                                filler_feature_values={2: value},
                                filler_feature_ranges={2: width},
                                legend=2)
    plt.title(lab)
```




```
plt.show()
```

يمكننا ملاحظة الاختلاففي حدود الفئة للنماذج المختلفة، بمافي ذلك المجموعة النموذجية (تم تنفيذ المخطط مع الأخذفي الاعتبار بيانات التدريب فقط). يتميز CatBoost بحد إقليمي متميز مقارنة بجميع الموديلات الأخرى. ومع ذلك، فإن نماذج XGBoost وRandom Forest

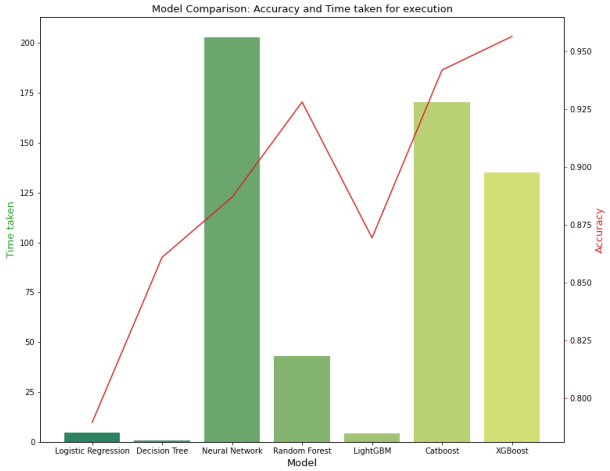
تحتوي أيضًا على عدد أقل بكثير من نقاط البيانات المصنفة بشكل خاطئ مقارنة بالنماذج الأخرى.

مقارنة نموذج التنبؤ بهطول الأمطار

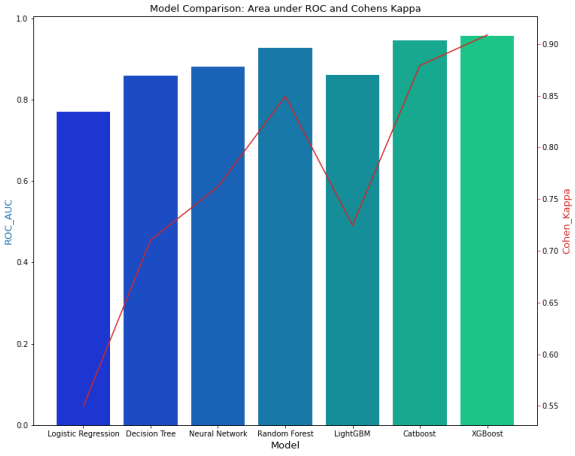
نحتاج الآن إلى تحديد النموذج الأفضل أداءً بناءً على Precision Score و ROC_AUC و Cohen's Kappa و Total Run Time. نقطة واحدة يجب ذكرها هنا هي: كان بإمكاننا اعتبار F1-Score مقياسًا أفضل للحكم على أداء النموذج بدلاً من الدقة، لكننا قمنا بالفعل بتحويل مجموعة البيانات غير المتوازنة إلى مجموعة متوازنة، لذا ضع في اعتبارك الدقة كمقياس لتحديد أفضل نموذج له ما يبرره في هذه الحالة.

لاتخاذ قرار أفضل، اخترنا "Cohen's Kappa" والذي يعد في الواقع خيارًا مثاليًا كمقياس لاتخاذ قرار بشأن أفضل نموذج في حالة مجموعات البيانات غير المتوازنة. دعنا نتحقق من النموذج الذي يعمل جيدًا على أي جبهة:

```
accuracy_scores = [accuracy_lr, accuracy_dt, accuracy_nn, accuracy_rf,
accuracy lgb, accuracy cb, accuracy xgb]
roc_auc_scores = [roc_auc_lr, roc_auc_dt, roc_auc_nn, roc_auc_rf,
roc_auc_lgb, roc_auc_cb, roc_auc_xgb]
coh_kap_scores = [coh_kap_lr, coh_kap_dt, coh_kap_nn, coh_kap_rf,
coh_kap_lgb, coh_kap_cb, coh_kap_xgb]
tt = [tt_lr, tt_dt, tt_nn, tt_rf, tt_lgb, tt_cb, tt_xgb]

model_data = {'Model': ['Logistic Regression','Decision Tree','Neural
Network','Random Forest','LightGBM','Catboost','XGBoost'],
              'Accuracy': accuracy_scores,
              'ROC_AUC': roc_auc_scores,
              'Cohen_Kappa': coh_kap_scores,
              'Time taken': tt}
data = pd.DataFrame(model_data)

fig, ax1 = plt.subplots(figsize=(12,10))
ax1.set_title('Model Comparison: Accuracy and Time taken for
execution', fontsize=13)
color = 'tab:green'
ax1.set_xlabel('Model', fontsize=13)
ax1.set_ylabel('Time taken', fontsize=13, color=color)
ax2 = sns.barplot(x='Model', y='Time taken', data = data,
palette='summer')
ax1.tick_params(axis='y')
ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Accuracy', fontsize=13, color=color)
ax2 = sns.lineplot(x='Model', y='Accuracy', data = data, sort=False,
color=color)
ax2.tick_params(axis='y', color=color)
```




```
fig, ax3 = plt.subplots(figsize=(12,10))
ax3.set_title('Model Comparison: Area under ROC and Cohens Kappa',
fontsize=13)
color = 'tab:blue'
ax3.set_xlabel('Model', fontsize=13)
ax3.set_ylabel('ROC_AUC', fontsize=13, color=color)
ax4 = sns.barplot(x='Model', y='ROC_AUC', data = data,
palette='winter')
ax3.tick_params(axis='y')
ax4 = ax3.twinx()
color = 'tab:red'
ax4.set_ylabel('Cohen_Kappa', fontsize=13, color=color)
ax4 = sns.lineplot(x='Model', y='Cohen_Kappa', data = data,
sort=False, color=color)
ax4.tick_params(axis='y', color=color)
plt.show()
```



يمكننا ملاحظة أن أداء XGBoost و CatBoost و Random Forest كان أفضل مقارنة بالموديلات الأخرى. ومع ذلك، إذا كانت السرعة أمرًا مهمًا يجب مراعاته، فيمكننا الالتزام بـ Random Forest بدلاً من XGBoost أو CatBoost.

43) التنبؤ بكفاءة استهلاك الوقود باستخدام التعلم الآلي Predict Fuel Efficiency using Machine Learning

في هذه الأنواع من مشاكل التعلم الآلي للتنبؤ بكفاءة الوقود، نهدف إلى التنبؤ بمخرجات القيمة المستمرة، مثل السعر أو الاحتمالية.في هذه المقالة، سأطلعك على كيفية التنبؤ بكفاءة استهلاك الوقود من خلال التعلم الآلي.

التنبؤ بكفاءة الوقود

سأستخدم هنا إحدى مجموعات البيانات الشهيرة بين ممارسي التعلم الآلي، مجموعة بيانات Auto MPG لإنشاء نموذج للتنبؤ بكفاءة استهلاك الوقود للمركباتفي أواخر السبعينيات وأوائل الثمانينيات. للقيام بذلك، سنزود النموذج بوصف للعديد من السيارات من هذه الفترة. يتضمن هذا الوصف سمات مثل الأسطوانات cylinders والإزاحة displacement والقدرة الحصانية horsepower والوزن weight.

دعنا نستورد المكتبات الضرورية للبدء بهذه المهمة:

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

الآن، الشيء التالي الذي يجب فعله هو تنزيل مجموعة البيانات. يمكنك بسهولة تنزيل مجموعة البيانات من هنا. الآن، دعنا نستورد البيانات باستخدام حزمة pandas:

```
column_names =
['MPG','Cylinders','Displacement','Horsepower','Weight',
                'Acceleration', 'Model Year', 'Origin']
dataset = pd.read_csv("auto.csv", names=column_names,
                      na_values = "?", comment='\t',
                      sep=" ", skipinitialspace=True)
```

عمود "origin"في مجموعة البيانات فئوي، لذا للمضي قدمًا نحتاج إلى استخدام بعض ترميز واحد ـ ساخن:

```
origin = dataset.pop('Origin')
dataset['USA'] = (origin == 1)*1.0
dataset['Europe'] = (origin == 2)*1.0
dataset['Japan'] = (origin == 3)*1.0
```

الآن، دعنا نقسم البيانات إلى مجموعات تدريب واختبار:

```
train_dataset = dataset.sample(frac=0.8,random_state=0)
test_dataset = dataset.drop(train_dataset.index)
```

قبل التدريب والاختبار للتنبؤ بكفاءة استهلاك الوقود من خلال التعلم الآلي، دعنا نرسم البيانات باستخدام طريقة المخطط الزوجي من seaborn:

```
sns.pairplot(train_dataset[["MPG", "Cylinders", "Displacement",
"Weight"]], diag_kind="kde")
```

الآن، سأفصل القيم المستهدفة عن الميزات الموجودةفي مجموعة البيانات. هذه التسمية هي تلك الميزة التي سأستخدمها لتدريب النموذج على التنبؤ بكفاءة الوقود:

```
train_labels = train_dataset.pop('MPG')
test_labels = test_dataset.pop('MPG')
```

تسوية البيانات

يوصى بتوحيد standardize الميزات التي تستخدم مقاييس ونطاقات مختلفة. على الرغم من أن النموذج يمكن أن يتقارب دون توحيد الميزات، إلا أن هذا يجعل التعلم أكثر صعوبة ويجعل النموذج الناتج يعتمد على اختيار الوحدات المستخدمةفي الإدخال. نحتاج إلى القيام بذلك لعرض مجموعة بيانات الاختبارفي نفس التوزيع الذي تم تدريب النموذج عليه:

```
def norm(x):
  return (x - train_stats['mean']) / train_stats['std']
normed_train_data = norm(train_dataset)
normed_test_data = norm(test_dataset)
```

ملاحظة: يجب تطبيق الإحصائيات المستخدمة لتسوية المدخلات هنا (المتوسط والانحراف والمعياري) على جميع البيانات الأخرى المقدمة إلى النموذج، باستخدام ترميز واحد-ساخن الذي قمنا به سابقًا. يتضمن ذلك مجموعة الاختبار بالإضافة إلى البيانات الحية عند استخدام النموذجفي الإنتاج.

بناء النموذج

دعونا نبني نموذجنا. هنا، سأستخدم API التسلسلي مع طبقتين مخفيتين وطبقة إخراج واحدة ستعيد قيمة واحدة. يتم تغليف خطوات إنشاء النموذجفي دالة، build_model، نظرًا لأننا سننشئ نموذجًا ثانيًا لاحقًا:

```
def build_model():
  model = keras.Sequential([
    layers.Dense(64, activation=tf.nn.relu,
input_shape=[len(train_dataset.keys())]),
    layers.Dense(64, activation=tf.nn.relu),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.001)

  model.compile(loss='mean_squared_error',
                optimizer=optimizer,
                metrics=['mean_absolute_error', 'mean_squared_error'])
  return model
model = build_model()
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                640
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 4,865
Trainable params: 4,865
Non-trainable params: 0
```

الآن، قبل تدريب النموذج للتنبؤ بكفاءة الوقود، دعنا نضع هذا النموذجفي العينات العشر الأولى:

```
example_batch = normed_train_data[:10]
example_result = model.predict(example_batch)
example_result
```

```
array([[-0.12670723],
       [-0.03443428],
       [ 0.3062502 ],
       [ 0.3065169 ],
       [ 0.36841604],
       [ 0.02191051],
       [ 0.3674207 ],
       [ 0.10561748],
       [ 0.00638346],
       [-0.00226454]], dtype=float32)
```

تدريب نموذج للتنبؤ بكفاءة الوقود

الآن، دعنا ندرب النموذج على التنبؤ بكفاءة الوقود:

```
class PrintDot(keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs):
    if epoch % 100 == 0: print('')
    print('.', end='')

EPOCHS = 1000

history = model.fit(
  normed_train_data, train_labels,
  epochs=EPOCHS, validation_split = 0.2, verbose=0,
  callbacks=[PrintDot()])
```

الآن، دعونا نرسم تدريب النموذج:

```
def plot_history(history):
  hist = pd.DataFrame(history.history)
  hist['epoch'] = history.epoch

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('Mean Abs Error [MPG]')
  plt.plot(hist['epoch'], hist['mean_absolute_error'],
           label='Train Error')
  plt.plot(hist['epoch'], hist['val_mean_absolute_error'],
```

```
            label = 'Val Error')
  plt.ylim([0,5])
  plt.legend()

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('Mean Square Error [$MPG^2$]')
  plt.plot(hist['epoch'], hist['mean_squared_error'],
            label='Train Error')
  plt.plot(hist['epoch'], hist['val_mean_squared_error'],
            label = 'Val Error')
  plt.ylim([0,20])
  plt.legend()
  plt.show()
plot_history(history)
```

يمثل هذا الرسم البياني أدناه تحسنًا طفيفًا أو حتى تدهورًافي خطأ التحقق بعد حوالي 100 فترة (حقبة). الآن، دعنا نحدِّث طريقة model.fit لإيقاف التدريب عندما لا تتحسن نتيجة التحقق. سنستخدم EarlyStopping من callback الذي يختبر حالة التدريب لكل فترة. إذا مر عدد محدد من الفترات دون إظهار تحسن، فقم بإيقاف التدريب تلقائيًا:

```
model = build_model()

# The patience parameter is the amount of epochs to check for
improvement
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss',
patience=10)

history = model.fit(normed_train_data, train_labels, epochs=EPOCHS,
                    validation_split = 0.2, verbose=0,
callbacks=[early_stop, PrintDot()])

plot_history(history)
```

يوضح الرسم البياني أنهفي مجموعة التحقق من الصحة، يكون متوسط الخطأ عادةً حوالي +/- 2 ميلافي الغالون (MPG). هل هذا جيد؟ سنترك هذا القرار لك.

دعونا نرى كيف يتم تعميم النموذج باستخدام مجموعة الاختبار، والتي لم نستخدمها عند تدريب النموذج. يوضح هذا إلى أي مدى يُتوقع من هذا النموذج أن يتنبأ عندما نستخدمهفي العالم الحقيقي:

```
loss, mae, mse = model.evaluate(normed_test_data, test_labels,
verbose=0)
print("Testing set Mean Abs Error: {:5.2f} MPG".format(mae))
```

```
Testing set Mean Abs Error:  1.97 MPG
```

الآن، دعنا نضع تنبؤات على النموذج للتنبؤ بكفاءة الوقود:

```
test_predictions = model.predict(normed_test_data).flatten()

plt.scatter(test_labels, test_predictions)
plt.xlabel('True Values [MPG]')
plt.ylabel('Predictions [MPG]')
plt.axis('equal')
```

```
plt.axis('square')
plt.xlim([0,plt.xlim()[1]])
plt.ylim([0,plt.ylim()[1]])
_ = plt.plot([-100, 100], [-100, 100])
```

44) التنبؤ بالهجرة باستخدام تعلم الآلة Predict Migration using Machine Learning

في هذه المقالة، سوف أخوضكفي مهمة حقيقية من مهام التعلم الآلي للتنبؤ بهجرة البشر بين البلدان. الهجرة البشرية هي نوع من التنقل البشري، حيث تنطوي الرحلة على تحرك الشخص لتغيير موطنه.

من المهم التنبؤ بالهجرة البشرية بأكبر قدر ممكن من الدقةفي تطبيقات تخطيط المدن، والتجارة الدولية، وانتشار الأمراض المعدية، وتخطيط الحفظ، وصنع السياسات العامة.

التنبؤ بالهجرة باستخدام التعلم الآلي

سأبدأ هذه المهمة للتنبؤ بالهجرة عن طريق استيراد جميع المكتبات الضرورية:

```
import pandas as pd
from sklearn.cross_validation import train_test_split
from sklearn import svm
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn.naive_bayes import GaussianNB
```

مجموعة البيانات التي أستخدمهافي هذه المهمة للتنبؤ بإمكانية تنزيل الترحيل بسهولة من هنا. دعونا نرى كيف تبدو البيانات: أود أن ألفت انتباهك إلى عمود "Measure" و "Country" و "CitizenShip" . إذا أردنا الحصول على نتيجة تنبؤ، فنحن بحاجة إلى تحويل كل قيم السلسلة هذه إلى عدد صحيح:

```
data = pd.read_csv('migration_nz.csv')
data.head(10)
```

	Measure	Country	Citizenship	Year	Value
0	Arrivals	Oceania	New Zealand Citizen	1979	11817.0
1	Arrivals	Oceania	Australian Citizen	1979	4436.0
2	Arrivals	Oceania	Total All Citizenships	1979	19965.0
3	Arrivals	Antarctica	New Zealand Citizen	1979	10.0
4	Arrivals	Antarctica	Australian Citizen	1979	0.0
5	Arrivals	Antarctica	Total All Citizenships	1979	13.0
6	Arrivals	American Samoa	New Zealand Citizen	1979	17.0
7	Arrivals	American Samoa	Australian Citizen	1979	4.0
8	Arrivals	American Samoa	Total All Citizenships	1979	30.0
9	Arrivals	Australia	New Zealand Citizen	1979	8224.0

لكن أولاً، دعنا نرى القيم الفريدة التي لدينافي عمود "Measure":

```
data['Measure'].unique()
```

```
array(['Arrivals', 'Departures', 'Net'], dtype=object)
```

نحتاج الآن إلى إعطاء قيمة عدد صحيح فريد لكل قيمة سلسلة فريدة:في حالة عدم وجود العديد من القيم، من الممكن استخدام دالة "replace":

```
data['Measure'].replace("Arrivals",0,inplace=True)
data['Measure'].replace("Departures",1,inplace=True)
data['Measure'].replace("Net",2,inplace=True)
```

دعنا الآن نتحقق مما إذا تم تعيين كل شيء بشكل صحيح:

```
data['Measure'].unique()
```

```
array([0, 1, 2])
```

في هذه الحالة، لدينا حوالي 250 دولة فريدة:

```
data['Country'].unique()
```

```
array(['Oceania', 'Antarctica', 'American Samoa', 'Australia',
       'Cocos Islands', 'Cook Islands', 'Christmas Island', 'Fiji',
       'Micronesia', 'Guam', 'Kiribati', 'Marshall Islands',
       'Northern Mariana Islands', 'New Caledonia', 'Norfolk Island',
       'Nauru', 'Niue', 'New Zealand', 'French Polynesia',
       'Papua New Guinea', 'Pitcairn Island', 'Palau', 'Solomon Islands',
       'French Southern Territories', 'Tokelau', 'Tonga', 'Tuvalu',
       'Vanuatu', 'Wallis and Futuna', 'Samoa', 'Asia', 'Afghanistan',
       'Armenia', 'Azerbaijan', 'Bangladesh', 'Brunei Darussalam',
       'Bhutan', 'China', 'Georgia', 'Hong Kong', 'Indonesia', 'India',
       'Japan', 'Kyrgyzstan', 'Cambodia', 'North Korea', 'South Korea',
       'Kazakhstan', 'Laos', 'Sri Lanka', 'Myanmar', 'Mongolia', 'Macau',
       'Maldives', 'Malaysia', 'Nepal', 'Philippines', 'Pakistan',
       'Singapore', 'Thailand', 'Tajikistan', 'Timor-Leste',
       'Turkmenistan', 'Taiwan', 'Uzbekistan', 'Vietnam', 'Europe',
       'Andorra', 'Albania', 'Austria', 'Bosnia and Herzegovina',
       'Belgium', 'Bulgaria', 'Belarus', 'Switzerland', 'Czechoslovakia',
       'Cyprus', 'Czechia', 'East Germany', 'Germany', 'Denmark',
       'Estonia', 'Spain', 'Finland', 'Faeroe Islands', 'France', 'UK',
       'Gibraltar', 'Greenland', 'Greece', 'Croatia', 'Hungary', 'Ireland',
       'Iceland', 'Italy', 'Kosovo', 'Liechtenstein', 'Lithuania',
       'Luxembourg', 'Latvia', 'Monaco', 'Moldova', 'Montenegro',
       'Macedonia', 'Malta', 'Netherlands', 'Norway', 'Poland', 'Portugal',
       'Romania', 'Serbia', 'Russia', 'Sweden', 'Slovenia', 'Slovakia',
       'San Marino', 'USSR', 'Ukraine', 'Vatican City',
```

نحتاج الآن إلى تعيين قيمة عدد صحيح فريد لكل قيمة سلسلة فريدة:

```
data['CountryID'] = pd.factorize(data.Country)[0]
data['CitID'] = pd.factorize(data.Citizenship)[0]
```

الآن، دعنا نرى ما إذا كان كل شيء على ما يرام:

```
data['CountryID'].unique()
```

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
       247, 248, 249, 250, 251, 252])
```

مشكلة أخرى هي أن لدينا بعض القيم المفقودة، دعنا نرى كم وأين توجد بالضبط:

```
data.isnull().sum()
```

```
Measure         0
Country         0
Citizenship     0
Year            0
Value          72
CountryID       0
CitID           0
dtype: int64
```

الآن، سأقوم ببساطة بملء هذه القيم المفقودة بالقيم المتوسطة:

```
data["Value"].fillna(data["Value"].median(),inplace=True)
```

الآن، دعنا نرى ما إذا كان كل شيء على ما يرام حتى الآن:

```
data.isnull().sum()
```

```
Measure        0
Country        0
Citizenship    0
Year           0
Value          0
CountryID      0
CitID          0
dtype: int64
```

تقسيم البيانات إلى مجموعات تدريب واختبار

الآن، سأقسم البيانات إلى 70في المائة من التدريب و30في المائةفي مجموعة الاختبار:

```
data.drop('Country', axis=1, inplace=True)
data.drop('Citizenship', axis=1, inplace=True)
from sklearn.cross_validation import train_test_split
X= data[['CountryID','Measure','Year','CitID']].as_matrix()
Y= data['Value'].as matrix()
X_train, X_test, y_train, y_test = train_test_split(
  X, Y, test_size=0.3, random_state=9)
```

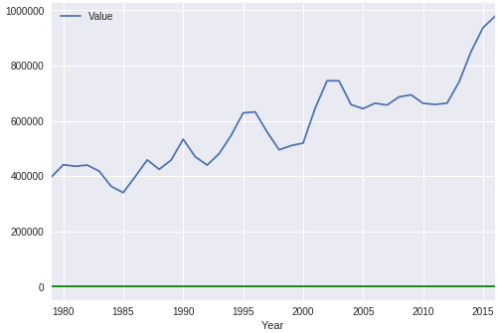
التنبؤ بالهجرة

الآن، دعنا نتنبؤ بالهجرة باستخدام خوارزمية التعلم الآلي الخاصة بنا ونعرض النتائج:

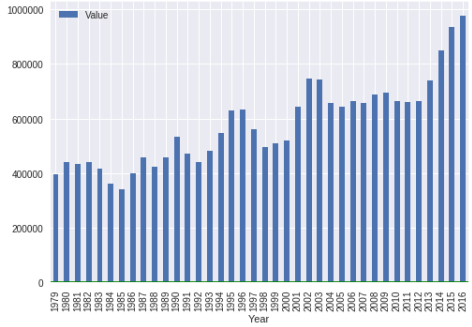
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=70,max_features =
3,max_depth=5,n_jobs=-1)
rf.fit(X_train ,y_train)
rf.score(X_test, y_test)
X = data[['CountryID','Measure','Year','CitID']]
```

```
0.73654599831394985
```

```
Y = data['Value']
X_train, X_test, y_train, y_test = train_test_split(
  X, Y, test_size=0.3, random_state=9)
grouped = data.groupby(['Year']).aggregate({'Value' : 'sum'})


#Growth of migration to New-Zeland by year
grouped.plot(kind='line');plt.axhline(0, color='g')
sns.plt.show()
```

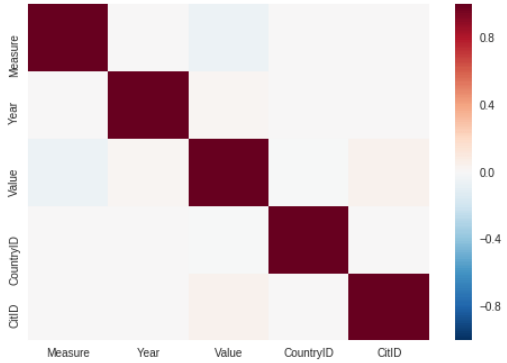



```
grouped.plot(kind='bar');plt.axhline(0, color='g')
sns.plt.show()
```




```
import seaborn as sns
corr = data.corr()
```

```
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
sns.plt.show()
```



آمل أن تكون قد أحببت هذا المقال لمهمة بسيطةفي العالم الحقيقي تستند إلى كيفية التنبؤ بهجرة البشر بين البلدان.

45) كشف الشذوذ باستخدام التعلم الآلي Anomaly Detection using Machine Learning

اكتشاف الشذوذ Anomaly detection هو عملية تحديد العناصر أو الأحداث غير المتوقعة في مجموعات البيانات، والتي تختلف عن القاعدة norm. وغالبًا ما يتم تطبيق اكتشاف الشذوذ على البيانات غير المصنفة والتي تُعرف باسم اكتشاف الشذوذ غير الخاضع للإشراف. كشف الشذوذ له افتراضان أساسيان:

- تحدث حالات الشذوذ نادرًا جدًافي البيانات.
- تختلف ميزاتها عن الحالات العادية بشكل كبير.

كشف الشذوذ وحيد المتغير

قبل أن نصل إلى اكتشاف الشذوذ متعدد المتغيرات، أعتقد أنه من الضروري العمل من خلال مثال بسيط لطريقة اكتشاف الشذوذ أحادي المتغير حيث نكتشف القيم المتطرفة من توزيع القيم في مساحة ميزة واحدة.

نحن نستخدم مجموعة بيانات Super Store Sales التي يمكن تنزيلها من هنا، وسنجد أنماطًا في المبيعات والأرباح بشكل منفصل لا تتوافق مع السلوك المتوقع. أي اكتشاف القيم المتطرفة لمتغير واحدفي كل مرة.

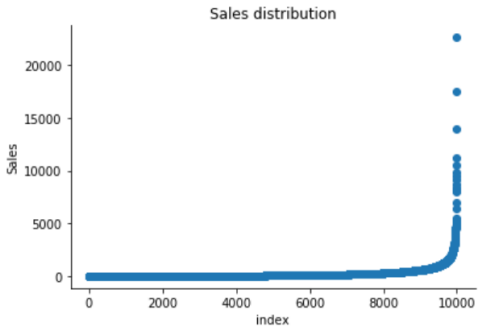
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib
from sklearn.ensemble import IsolationForest
```

توزيع المبيعات

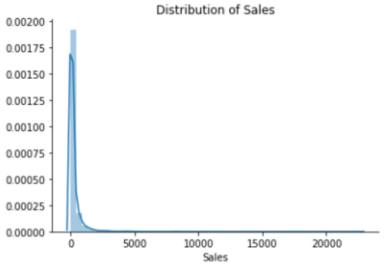
```
df = pd.read_excel("Superstore.xls")
df['Sales'].describe()
```

```
count     9994.000000
mean       229.858001
std        623.245101
min          0.444000
25%         17.280000
50%         54.490000
75%        209.940000
max      22638.480000
Name: Sales, dtype: float64
```

```
plt.scatter(range(df.shape[0]), np.sort(df['Sales'].values))
plt.xlabel('index')
plt.ylabel('Sales')
plt.title("Sales distribution")
sns.despine()
```




```
sns.distplot(df['Sales'])
plt.title("Distribution of Sales")
sns.despine()
```




```
print("Skewness: %f" % df['Sales'].skew())
print("Kurtosis: %f" % df['Sales'].kurt())
```

```
Skewness: 12.972752
Kurtosis: 305.311753
```

توزيع مبيعات Superstore بعيد كل البعد عن التوزيع الطبيعي، وله ذيل رفيع طويل إيجابي، وتتركز كتلة التوزيع على يسار الشكل. وتوزيع مبيعات الذيل يتجاوز بكثير ذيول التوزيع الطبيعي.

هناك منطقة واحدة يكون فيها احتمال ظهور البيانات منخفضًا وهي على الجانب الأيمن من التوزيع.

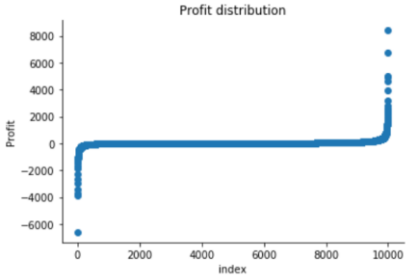
توزيع الربح

```
df['Profit'].describe()
```

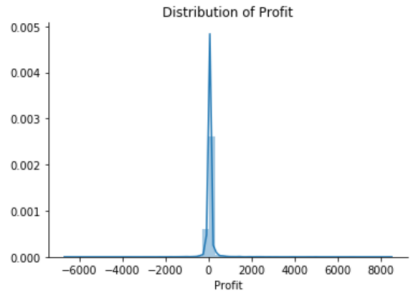
```
count    9994.000000
mean       28.656896
std       234.260108
min     -6599.978000
25%         1.728750
50%         8.666500
75%        29.364000
max      8399.976000
Name: Profit, dtype: float64
```

```
plt.scatter(range(df.shape[0]), np.sort(df['Profit'].values))
plt.xlabel('index')
plt.ylabel('Profit')
```

```
plt.title("Profit distribution")
sns.despine()
```




```
sns.distplot(df['Profit'])
plt.title("Distribution of Profit")
sns.despine()
```




```
print("Skewness: %f" % df['Profit'].skew())
print("Kurtosis: %f" % df['Profit'].kurt())
```

```
Skewness: 7.561432
Kurtosis: 397.188515
```

توزيع أرباح Superstore له ذيل إيجابي وذيل سلبي. ومع ذلك، فإن الذيل الموجب أطول من الذيل السلبي. لذا فإن التوزيع منحرف بشكل إيجابي، والبيانات ثقيلة الذيل أو وفرة من القيم المتطرفة.

هناك منطقتان تقل احتمالية ظهور البيانات فيهما: واحدة على الجانب الأيمن من التوزيع، وأخرى على اليسار.

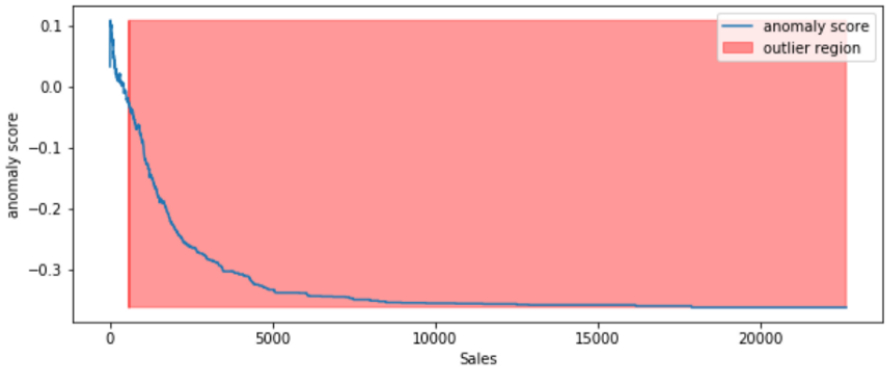
كشف الشذوذ أحادي المتغير في المبيعات

تعد Isolation Forest خوارزمية لاكتشاف القيم المتطرفة التي تُرجع درجة الانحراف لكل عينة باستخدام خوارزمية IsolationForest التي تستند إلى حقيقة أن الحالات الشاذة هي نقاط بيانات قليلة ومختلفة. غابة العزلة هي نموذج قائم على الأشجار.في هذه الأشجار، يتم إنشاء الأقسام عن طريق التحديد العشوائي للمعلم أولاً ثم تحديد قيمة الانقسام العشوائي بين الحد الأدنى والحد الأقصى لقيمة المعلم المحدد.

توضح العملية التالية كيف يتصرف IsolationForestفي حالة مبيعات Susperstore، وتم تنفيذ الخوارزميةفي Sklearn وتم استعارة الرمز إلى حد كبير من هذا البرنامج التعليمي.

- تدريب IsolationForest باستخدام بيانات المبيعات.
- قم بتخزين المبيعاتفي مجموعة NumPy لاستخدامهافي نماذجنا لاحقًا.
- تم حساب درجة الشذوذ لكل ملاحظة. يتم حساب درجة الشذوذ لعينة الإدخال على أنها متوسط درجة الشذوذ للأشجارفي الغابة.
- صنف كل ملاحظة على أنها شاذة أو غير شاذة.
- يسلط التصور الضوء على المناطق التي تقع فيها القيم المتطرفة.

```
isolation_forest = IsolationForest(n_estimators=100)
isolation_forest.fit(df['Sales'].values.reshape(-1, 1))
xx = np.linspace(df['Sales'].min(), df['Sales'].max(),
len(df)).reshape(-1,1)
anomaly_score = isolation_forest.decision_function(xx)
outlier = isolation_forest.predict(xx)
plt.figure(figsize=(10,4))
plt.plot(xx, anomaly_score, label='anomaly score')
plt.fill_between(xx.T[0], np.min(anomaly_score),
np.max(anomaly_score),
                 where=outlier==-1, color='r',
                 alpha=.4, label='outlier region')
plt.legend()
plt.ylabel('anomaly score')
plt.xlabel('Sales')
plt.show();
```



وفقًا للنتائج والتصور أعلاه، يبدو أن المبيعات التي تتجاوز 1000 ستُعتبر بالتأكيد خارجة عن المألوف.

تحقق بصريًا من شذوذ واحد

```
df.iloc[10]
```

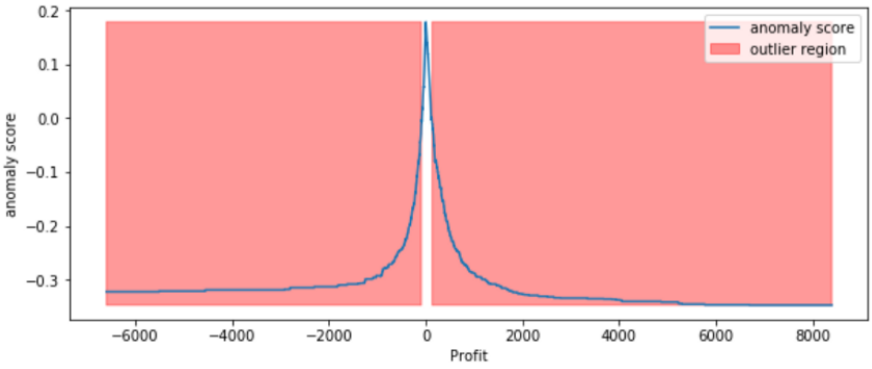
```
Row ID                                          11
Order ID                                CA-2014-115812
Order Date                        2014-06-09 00:00:00
Ship Date                         2014-06-14 00:00:00
Ship Mode                            Standard Class
Customer ID                                BH-11710
Customer Name                       Brosina Hoffman
Segment                                    Consumer
Country                               United States
City                                    Los Angeles
State                                    California
Postal Code                               90032
Region                                         West
Product ID                        FUR-TA-10001539
Category                                 Furniture
Sub-Category                               Tables
Product Name     Chromcraft Rectangular Conference Tables
Sales                                      1706.18
Quantity                                         9
Discount                                       0.2
Profit                                     85.3092
Name: 10, dtype: object
```

يبدو هذا الشراء طبيعيًا بالنسبة لي، وأتوقع أنه كان قدرًا أكبر من المبيعات مقارنة بالطلبات الأخرى في البيانات.

كشف الشذوذ أحادي المتغير على الربح

- تدريب IsolationForest باستخدام متغير الربح.
- قم بتخزين الأرباح في مصفوفة NumPy لاستخدامها في نماذجنا لاحقًا.
- تم حساب درجة الشذوذ لكل ملاحظة. يتم حساب درجة الشذوذ لعينة الإدخال على أنها متوسط درجة الشذوذ للأشجار في الغابة.
- صنف كل ملاحظة على أنها شاذة أو غير شاذة.
- يسلط التصور الضوء على المناطق التي تقع فيها القيم المتطرفة.

```
isolation_forest = IsolationForest(n_estimators=100)
isolation_forest.fit(df['Profit'].values.reshape(-1, 1))
xx = np.linspace(df['Profit'].min(), df['Profit'].max(),
len(df)).reshape(-1,1)
anomaly_score = isolation_forest.decision_function(xx)
outlier = isolation_forest.predict(xx)
plt.figure(figsize=(10,4))
plt.plot(xx, anomaly_score, label='anomaly score')
plt.fill_between(xx.T[0], np.min(anomaly_score),
np.max(anomaly_score),
                 where=outlier==-1, color='r',
                 alpha=.4, label='outlier region')
plt.legend()
plt.ylabel('anomaly score')
plt.xlabel('Profit')
plt.show();
```



تحقق بصريًا من بعض الحالات الشاذة

وفقًا للنتائج والتصور أعلاه، يبدو أن الربح الذي يقل عن 100ـ أو يتجاوز 100 سيتم اعتباره قيمة خارجية، دعنا نفحص بصريًا مثالًا واحدًا يحدده نموذجنا ونرى ما إذا كان له معنى.

```
df.iloc[3]
```

```
Row ID                                          4
Order ID                              US-2015-108966
Order Date                      2015-10-11 00:00:00
Ship Date                       2015-10-18 00:00:00
Ship Mode                          Standard Class
Customer ID                                SO-20335
Customer Name                       Sean O'Donnell
Segment                                  Consumer
Country                             United States
City                                Fort Lauderdale
State                                       Florida
Postal Code                                   33311
Region                                        South
Product ID                        FUR-TA-10000577
Category                                Furniture
Sub-Category                               Tables
Product Name    Bretford CR4500 Series Slim Rectangular Table
Sales                                       957.577
Quantity                                          5
Discount                                       0.45
Profit                                      -383.031
outlier                                           0
Name: 3, dtype: object
```

أي ربح سلبي سيكون أمرًا شاذًا ويجب إجراء مزيد من التحقيق، وهذا غني عن البيان.

```
df.iloc[1]
```

```
Row ID                                          2
Order ID                              CA-2016-152156
Order Date                      2016-11-08 00:00:00
Ship Date                       2016-11-11 00:00:00
Ship Mode                            Second Class
Customer ID                                CG-12520
Customer Name                         Claire Gute
Segment                                  Consumer
Country                             United States
City                                    Henderson
State                                      Kentucky
Postal Code                                   42420
Region                                        South
Product ID                        FUR-CH-10000454
Category                                Furniture
Sub-Category                               Chairs
Product Name    Hon Deluxe Fabric Upholstered Stacking Chairs,...
Sales                                       731.94
Quantity                                          3
Discount                                          0
Profit                                      219.582
Name: 1, dtype: object
```

قرر نموذجنا أن هذا الأمر الذي يحقق ربحًا كبيرًا هو أمر شاذ. ومع ذلك، عندما نتحرى عن هذا الطلب، يمكن أن يكون مجرد منتج به هامش مرتفع نسبيًا.

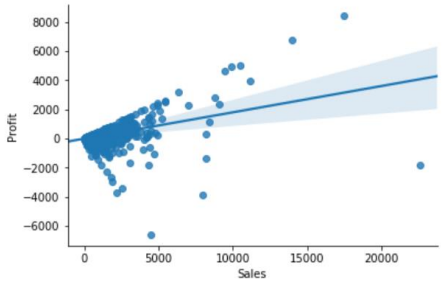
يُظهر التمثيلان المرئيان أعلاه درجات الانحراف ويبرزان المناطق التي توجد فيها القيم المتطرفة. كما هو متوقع، تعكس درجة الانحراف شكل التوزيع الأساسي وتتوافق المناطق النائية مع مناطق الاحتمالية المنخفضة.

ومع ذلك، فإن التحليل أحادي المتغير يمكن أن يقودنا فقط حتى الآن. قد ندرك أن بعض هذه الحالات الشاذة التي حددتها نماذجنا ليست هي الحالات الشاذة التي توقعناها. عندما تكون بياناتنا متعددة الأبعاد على عكس المتغير أحادي المتغير، تصبح طرق اكتشاف الشذوذ أكثر كثافة من الناحية الحسابية وأكثر تعقيدًا من الناحية الرياضية.

المبيعات والربح

عندما نعمل في مجال الأعمال التجارية، نتوقع أن يكون هناك ارتباط إيجابي بين المبيعات والأرباح. إذا لم تكن بعض نقاط بيانات المبيعات ونقاط بيانات الربح مرتبطة بشكل إيجابي، فسيتم اعتبارها قيمًا متطرفة وتحتاج إلى مزيد من التحقيق.

```
sns.regplot(x="Sales", y="Profit", data=df)
sns.despine();
```



من مخطط الارتباط أعلاه، يمكننا أن نرى أن بعض نقاط البيانات هي قيم متطرفة واضحة مثل القيم المنخفضة للغاية والقيم المرتفعة للغاية.

عامل خارجي محلي قائم على الكتلة (CBLOF)

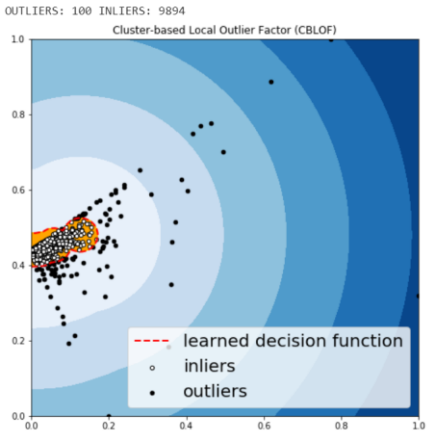
تحسب CBLOF النتيجة الشاذة بناءً على عامل خارجي محلي قائم على الكتلة. يتم حساب درجة الشذوذ من خلال مسافة كل حالة إلى مركز المجموعة الخاص بها مضروبةفي الحالات التي تنتمي إلى مجموعتها. تتضمن مكتبة PyOD تطبيق CBLOF.

تم استعارة الكود التالي من البرنامج التعليمي PyOD مع هذه المقالة.

- زيادة المبيعات والربح إلى ما بين صفر وواحد.
- قم بتعيين كسر القيم المتطرفة بشكل تعسفي على 1٪ بناءً على التجربة وأفضل تخمين.
- قم بملاءمة البيانات مع نموذج CBLOF وتوقع النتائج.
- استخدم قيمة العتبة للنظرفي أن نقطة البيانات داخلية أو خارجية.
- استخدم دالة القرار لحساب درجة الانحراف لكل نقطة.

```
outliers_fraction = 0.01
xx , yy = np.meshgrid(np.linspace(0, 1, 100), np.linspace(0, 1, 100))
clf = CBLOF(contamination=outliers_fraction,check_estimator=False,
random_state=0)
clf.fit(X)
scores_pred = clf.decision_function(X) * -1
y_pred = clf.predict(X)
n_inliers = len(y_pred) - np.count_nonzero(y_pred)
n_outliers = np.count_nonzero(y_pred == 1)

plt.figure(figsize=(8, 8))

df1 = df
df1['outlier'] = y_pred.tolist()

# sales - inlier feature 1,  profit - inlier feature 2
inliers_sales = np.array(df1['Sales'][df1['outlier'] == 0]).reshape(-
1,1)
inliers_profit = np.array(df1['Profit'][df1['outlier'] ==
0]).reshape(-1,1)

# sales - outlier feature 1, profit - outlier feature 2
outliers_sales = df1['Sales'][df1['outlier'] == 1].values.reshape(-
1,1)
outliers_profit = df1['Profit'][df1['outlier'] == 1].values.reshape(-
1,1)

print('OUTLIERS:',n_outliers,'INLIERS:',n_inliers)
threshold = percentile(scores_pred, 100 * outliers_fraction)
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) * -1
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold,
7),cmap=plt.cm.Blues_r)
a = plt.contour(xx, yy, Z, levels=[threshold],linewidths=2,
colors='red')
plt.contourf(xx, yy, Z, levels=[threshold, Z.max()],colors='orange')
b = plt.scatter(inliers_sales, inliers_profit, c='white',s=20,
edgecolor='k')

c = plt.scatter(outliers_sales, outliers_profit, c='black',s=20,
edgecolor='k')

plt.axis('tight')
plt.legend([a.collections[0], b,c], ['learned decision function',
'inliers','outliers'],

prop=matplotlib.font_manager.FontProperties(size=20),loc='lower
right')
plt.xlim((0, 1))
plt.ylim((0, 1))
plt.title('Cluster-based Local Outlier Factor (CBLOF)')
```

```
plt.show();
```

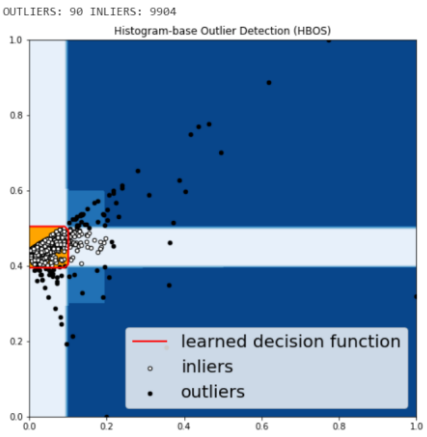


الكشف الخارجي القائم على الهستوكرام (HBOS)

يفترض HBOS استقلالية الميزة ويحسب درجة الانحرافات من خلال إنشاء الهستوكرام.في اكتشاف الشذوذ متعدد المتغيرات، يمكن حساب الرسم البياني لكل ميزة فردية وتسجيلها بشكل فردي ودمجهافي النهاية. عند استخدام مكتبة PyOD، فإن الكود مشابه جدًا لـ CBLOF.

```
outliers_fraction = 0.01
xx , yy = np.meshgrid(np.linspace(0, 1, 100), np.linspace(0, 1, 100))
clf = HBOS(contamination=outliers_fraction)
clf.fit(X)
scores_pred = clf.decision_function(X) * -1
y_pred = clf.predict(X)
n_inliers = len(y_pred) - np.count_nonzero(y_pred)
n_outliers = np.count_nonzero(y_pred == 1)
plt.figure(figsize=(8, 8))
df1 = df
df1['outlier'] = y_pred.tolist()

# sales - inlier feature 1,  profit - inlier feature 2
inliers_sales = np.array(df1['Sales'][df1['outlier'] == 0]).reshape(-
1,1)
inliers_profit = np.array(df1['Profit'][df1['outlier'] ==
0]).reshape(-1,1)

# sales - outlier feature 1, profit - outlier feature 2
outliers_sales = df1['Sales'][df1['outlier'] == 1].values.reshape(-
1,1)
outliers_profit = df1['Profit'][df1['outlier'] == 1].values.reshape(-
1,1)

print('OUTLIERS:',n_outliers,'INLIERS:',n_inliers)
threshold = percentile(scores_pred, 100 * outliers_fraction)=
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) * -1
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold,
7),cmap=plt.cm.Blues_r)
a = plt.contour(xx, yy, Z, levels=[threshold],linewidths=2,
colors='red')
```

```
plt.contourf(xx, yy, Z, levels=[threshold, Z.max()],colors='orange')
b = plt.scatter(inliers_sales, inliers_profit, c='white',s=20,
edgecolor='k')

c = plt.scatter(outliers_sales, outliers_profit, c='black',s=20,
edgecolor='k')

plt.axis('tight')
plt.legend([a.collections[0], b,c], ['learned decision function',
'inliers','outliers'],

prop=matplotlib.font_manager.FontProperties(size=20),loc='lower
right')
plt.xlim((0, 1))
plt.ylim((0, 1))
plt.title('Histogram-base Outlier Detection (HBOS)')
plt.show();
```



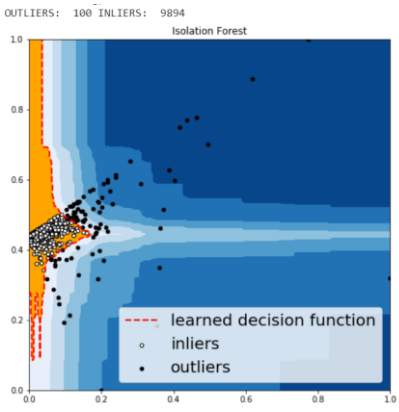
Isolation Forest

تشبه Isolation Forest من حيث المبدأ Random Forest وهي مبنية على أساس أشجار القرار. تقوم Isolation Forest بعزل الملاحظات عن طريق تحديد معلم عشوائيًا ثم تحديد قيمة تقسيم عشوائيًا بين الحد الأقصى والحد الأدنى لقيم ذلك المعلم المحدد.

وحدة PyOD Isolation Forest عبارة عن wrapper لـ Scikit-Learn Isolation Forest مع المزيد من الوظائف.

```
outliers_fraction = 0.01
xx , yy = np.meshgrid(np.linspace(0, 1, 100), np.linspace(0, 1, 100))
clf = IForest(contamination=outliers_fraction,random_state=0)
clf.fit(X)
scores_pred = clf.decision_function(X) * -1

y_pred = clf.predict(X)
n_inliers = len(y_pred) - np.count_nonzero(y_pred)
n_outliers = np.count_nonzero(y_pred == 1)
plt.figure(figsize=(8, 8))

df1 = df
```

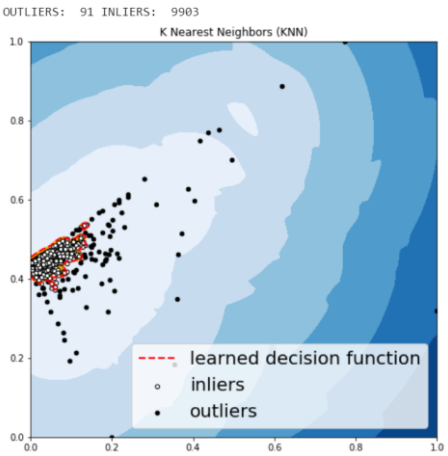
```
df1['outlier'] = y_pred.tolist()

# sales - inlier feature 1,  profit - inlier feature 2
inliers_sales = np.array(df1['Sales'][df1['outlier'] == 0]).reshape(-
1,1)
inliers_profit = np.array(df1['Profit'][df1['outlier'] ==
0]).reshape(-1,1)

# sales - outlier feature 1, profit - outlier feature 2
outliers_sales = df1['Sales'][df1['outlier'] == 1].values.reshape(-
1,1)
outliers_profit = df1['Profit'][df1['outlier'] == 1].values.reshape(-
1,1)

print('OUTLIERS: ',n_outliers,'INLIERS: ',n_inliers)

threshold = percentile(scores_pred, 100 * outliers_fraction)
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) * -1
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold,
7),cmap=plt.cm.Blues_r)
a = plt.contour(xx, yy, Z, levels=[threshold],linewidths=2,
colors='red')
plt.contourf(xx, yy, Z, levels=[threshold, Z.max()],colors='orange')
b = plt.scatter(inliers_sales, inliers_profit, c='white',s=20,
edgecolor='k')

c = plt.scatter(outliers_sales, outliers_profit, c='black',s=20,
edgecolor='k')

plt.axis('tight')
plt.legend([a.collections[0], b,c], ['learned decision function',
'inliers','outliers'],

prop=matplotlib.font_manager.FontProperties(size=20),loc='lower
right')
plt.xlim((0, 1))
plt.ylim((0, 1))
plt.title('Isolation Forest')
plt.show();
```



K - أقرب الجيران (KNN)

KNN هي واحدة من أبسط الطرقفي اكتشاف الشذوذ. بالنسبة لنقطة البيانات، يمكن اعتبار المسافة التي تفصلها عن أقرب جار لها k هي الدرجة الخارجية.

```
outliers_fraction = 0.01
xx , yy = np.meshgrid(np.linspace(0, 1, 100), np.linspace(0, 1, 100))
clf = KNN(contamination=outliers_fraction)
clf.fit(X)
scores pred = clf.decision function(X) * -1
y_pred = clf.predict(X)
n_inliers = len(y_pred) - np.count_nonzero(y_pred)
n_outliers = np.count_nonzero(y_pred == 1)
plt.figure(figsize=(8, 8))

df1 = df
df1['outlier'] = y_pred.tolist()

# sales - inlier feature 1,  profit - inlier feature 2
inliers_sales = np.array(df1['Sales'][df1['outlier'] == 0]).reshape(-
1,1)
inliers_profit = np.array(df1['Profit'][df1['outlier'] ==
0]).reshape(-1,1)

# sales - outlier feature 1, profit - outlier feature 2
outliers_sales = df1['Sales'][df1['outlier'] == 1].values.reshape(-
1,1)
outliers_profit = df1['Profit'][df1['outlier'] == 1].values.reshape(-
1,1)

print('OUTLIERS: ',n_outliers,'INLIERS: ',n_inliers)

threshold = percentile(scores_pred, 100 * outliers_fraction)

Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) * -1
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold,
7),cmap=plt.cm.Blues_r)
a = plt.contour(xx, yy, Z, levels=[threshold],linewidths=2,
colors='red')
plt.contourf(xx, yy, Z, levels=[threshold, Z.max()],colors='orange')
b = plt.scatter(inliers_sales, inliers_profit, c='white',s=20,
edgecolor='k')
c = plt.scatter(outliers_sales, outliers_profit, c='black',s=20,
edgecolor='k')
plt.axis('tight')
plt.legend([a.collections[0], b,c], ['learned decision function',
'inliers','outliers'],

prop=matplotlib.font_manager.FontProperties(size=20),loc='lower
right')
plt.xlim((0, 1))
plt.ylim((0, 1))
plt.title('K Nearest Neighbors (KNN)')
plt.show();
```



لم تكن الحالات الشاذة التي تنبأت بها الخوارزميات الأربعة المذكورة أعلاه مختلفة تمامًا.

تحقق بصريًا من بعض الحالات الشاذة

قد نرغب في التحقيق في كل من القيم المتطرفة التي يحددها نموذجنا، على سبيل المثال، دعنا نلقي نظرة على تفاصيل بعض القيم المتطرفة التي حددتها KNN، ونحاول فهم ما يجعلها شذوذًا.

```
df.iloc[1995]
```

```
Row ID                                          1996
Order ID                                US-2017-147221
Order Date                     2017-12-02 00:00:00
Ship Date                      2017-12-04 00:00:00
Ship Mode                              Second Class
Customer ID                                JS-16030
Customer Name                           Joy Smith
Segment                                    Consumer
Country                              United States
City                                       Houston
State                                        Texas
Postal Code                                  77036
Region                                     Central
Product ID                          OFF-AP-10002534
Category                            Office Supplies
Sub-Category                             Appliances
Product Name     3.6 Cubic Foot Counter Height Office Refrigerator
Sales                                       294.62
Quantity                                         5
Discount                                       0.8
Profit                                      -766.012
Name: 1995, dtype: object
```

بالنسبة لهذا الطلب المحدد، اشترى عميل 5 منتجات بسعر إجمالي 294.62 وربح أقل من -766، بخصم 80٪. يبدو وكأنه تصريح. يجب أن نكون على دراية بالخسارة لكل منتج نبيعه.

```
df.iloc[9649]
```

```
Row ID                                          9650
Order ID                                CA-2016-107104
Order Date                        2016-11-26 00:00:00
Ship Date                         2016-11-30 00:00:00
Ship Mode                            Standard Class
Customer ID                                MS-17365
Customer Name                     Maribeth Schnelling
Segment                                    Consumer
Country                               United States
City                                    Los Angeles
State                                    California
Postal Code                               90045
Region                                         West
Product ID                       FUR-BO-10002213
Category                                 Furniture
Sub-Category                             Bookcases
Product Name     DMI Eclipse Executive Suite Bookcases
Sales                                      3406.66
Quantity                                         8
Discount                                      0.15
Profit                                     160.314
Name: 9649, dtype: object
```

بالنسبة لعملية الشراء هذه، يبدو لي أن الربح عند حوالي 4.7٪ صغير جدًا وأن النموذج قرر أن هذا الأمر يعد أمرًا شاذًا.

```
df.iloc[9270]
```

```
Row ID                                          9271
Order ID                                US-2017-102183
Order Date                        2017-08-21 00:00:00
Ship Date                         2017-08-28 00:00:00
Ship Mode                            Standard Class
Customer ID                                PK-19075
Customer Name                           Pete Kriz
Segment                                    Consumer
Country                               United States
City                                  New York City
State                                      New York
Postal Code                               10035
Region                                         East
Product ID                       OFF-BI-10001359
Category                              Office Supplies
Sub-Category                                Binders
Product Name     GBC DocuBind TL300 Electric Binding System
Sales                                      4305.55
Quantity                                         6
Discount                                       0.2
Profit                                     1453.12
Name: 9270, dtype: object
```

بالنسبة للطلب أعلاه، اشترى عميل 6 منتجات بسعر 4305في السعر الإجمالي، وبعد خصم 20٪، ما زلنا نحصل على أكثر من 33٪ من الربح. نود أن يكون لدينا المزيد من هذه الحالات الشاذة.

يمكن العثور على نوتبوك Jupyter للتحليل أعلاه على Github.

46) تصنيف الأخبار باستخدام التعلم الآلي News Classification using Machine Learning

يجب أن تكون قد شاهدت الأخبار مقسمة إلى فئات عندما تذهب إلى موقع إخباري. بعض الفئات الشائعة التي ستشاهدهافي أي موقع إخباري تقريبًا هي التكنولوجيا والترفيه والرياضة. إذا كنت تريد معرفة كيفية تصنيف فئات الأخبار باستخدام التعلم الآلي، فهذه المقالة مناسبة لك.في هذه المقالة، سوف أطلعك على مهمة تصنيف الأخبار باستخدام التعلم الآلي باستخدام Python.

تصنيف الأخبار

يصنف كل موقع إخباري المقال الإخباري قبل نشره بحيث يمكن للزائرينفي كل مرة يزورون موقع الويب الخاص بهم النقر بسهولة على نوع الأخبار التي تهمهم. على سبيل المثال، أحب قراءة آخر تحديثات التكنولوجيا، لذلكفي كل مرة أزور فيها موقعًا إخباريًا، أنقر على قسم التكنولوجيا. ولكن قد ترغب أو لا ترغبفي القراءة عن التكنولوجيا، فقد تكون مهتمًا بالسياسة أو الأعمال أو الترفيه أو ربما الرياضة.

حاليًا، يتم تصنيف المقالات الإخبارية يدويًا بواسطة مديري المحتوىفي مواقع الويب الإخبارية. ولكن لتوفير الوقت، يمكنهم أيضًا تنفيذ نموذج التعلم الآلي على مواقع الويب الخاصة بهم والذي يقرأ عنوان الأخبار أو محتوى الأخبار ويصنف فئة الأخبار.في القسم أدناه، سأطلعك على كيفية تدريب نموذج التعلم الآلي لمهمة تصنيف الأخبار باستخدام لغة برمجة Python.

تصنيف الأخبار باستخدام بايثون

بالنسبة لمهمة تصنيف الأخبار باستخدام التعلم الآلي، فقد جمعت مجموعة بيانات من Kaggle، والتي تحتوي على مقالات إخبارية بمافي ذلك عناوينها وفئاتها. الفئات المشمولةفي مجموعة البيانات هذه هي:

1. رياضة Sports.
2. اعمال Business.
3. سياسة Politics.
4. تقنية Tech.
5. وسائل الترفيه Entertainment.

لذلك دعونا نستورد مكتبات Python الضرورية ومجموعة البيانات التي نحتاجها لهذه المهمة:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

```
from sklearn.naive_bayes import MultinomialNB

data =
pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-
data/master/bbc-news-data.csv", sep='\t')
print(data.head())
```

```
   category  ...                                            content
0  business  ...   Quarterly profits at US media giant TimeWarne...
1  business  ...   The dollar has hit its highest level against ...
2  business  ...   The owners of embattled Russian oil giant Yuk...
3  business  ...   British Airways has blamed high fuel prices f...
4  business  ...   Shares in UK drinks and food firm Allied Dome...

[5 rows x 4 columns]
```

الآن، دعنا نلقي نظرة سريعة على ما إذا كانت مجموعة البيانات هذه تحتوي على أي قيم فارغة أم لا:

```
data.isnull().sum()
```

```
category    0
filename    0
title       0
content     0
dtype: int64
```

توجد التسميات التي نحتاج إلى تصنيفها من مجموعة البيانات هذهفي عمود الفئة بهذه البيانات، دعنا نلقي نظرة على توزيع جميع فئات الأخبار:

```
data["category"].value_counts()
```

```
sport           511
business        510
politics        417
tech            401
entertainment   386
Name: category, dtype: int64
```

نموذج تصنيف الأخبار

دعنا الآن نجهز البيانات لمهمة تدريب نموذج تصنيف الأخبار:

```
data = data[["title", "category"]]

x = np.array(data["title"])
y = np.array(data["category"])

cv = CountVectorizer()
X = cv.fit_transform(x)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)
```

سأستخدم الآن خوارزمية Multinomial Naive Bayes لتدريب نموذج تصنيف الأخبار:

```
model = MultinomialNB()
model.fit(X_train,y_train)
```

أخيرًا، دعنا نختبر كيفية عمل هذا النموذج على أحد العناوين الرئيسيةفي أخبار اليوم:

```
user = input("Enter a Text: ")
data = cv.transform([user]).toarray()
output = model.predict(data)
print(output
```

```
Enter a Text: Latest Apple iPhone SE 3 concept renders show a compact smartphone in the style of the
iPhone 4
['tech']
```

هذه هي الطريقة التي يمكنك بها تدريب نموذج تصنيف الأخبار باستخدام التعلم الآلي باستخدام Python.

الملخص

هذه هي الطريقة التي يمكننا بها استخدام التعلم الآلي لتصنيف فئات الأخبار. يقوم كل موقع إخباري بتصنيف المقال الإخباري قبل نشره بحيث يمكن للزائرينفي كل مرة يزورون موقع الويب الخاص بهم النقر بسهولة على نوع الأخبار التي تهمهم.

47) أمان الشبكة باستخدام التعلم الآلي Network Security using Machine Learning

الطريقة الأكثر احتمالاً للمهاجمين للوصول إلى البنية التحتية الخاصة بك هي عبر الشبكة. أمان الشبكة Network security هو الممارسة العامة لحماية شبكات الكمبيوتر والأجهزة التي يمكن الوصول إليها من الشبكة ضد النوايا الخبيثة وإساءة الاستخدام والحرمان.في هذه المقالة، سوف أطلعك على تقنيات تحليل أمان الشبكة باستخدام التعلم الآلي.

يُعد استكشاف الأنماط أحد نقاط القوة الرئيسية للتعلم الآلي، وهناك العديد من الأنماط المتأصلة التي يجب اكتشافهافي بيانات حركة مرور الشبكة. للوهلة الأولى، قد تظهر بيانات التقاط حزم الشبكة متقطعة وعشوائية، ولكن معظم تدفقات الاتصال تتبع بروتوكول شبكة صارم.

يُعد التقاط بيانات الشبكة الحية هو الطريقة الأساسية لتسجيل نشاط الشبكة للتحليل عبر الإنترنت أو دون الاتصال بالإنترنت. مثل كاميرا CCTV عند تقاطع حركة المرور، يقوم محللو الحزم باعتراض حركة مرور الشبكة وتسجيلها. دعنا الآن ننشئ مصنِّفًا لهجمات الشبكة من البداية باستخدام التعلم الآلي.

بناء نموذج تنبؤي لتصنيف هجمات أمن الشبكات

مجموعة البيانات التي سنستخدمها هي مجموعة بيانات NSLKDD، والتي تعد تحسينًا على اكتشاف اختراق الشبكة التقليدي. تستخدم مجموعة البيانات هذه على نطاق واسع بواسطة متخصصي علوم بيانات الأمان لتصنيف مشاكل أمان الشبكة. يمكنك تنزيل مجموعة البيانات من هنا. لنبدأ هذه المهمة عن طريق استيراد بعض المكتبات الضرورية:

```
import os
from collections import defaultdict
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

استكشاف البيانات:

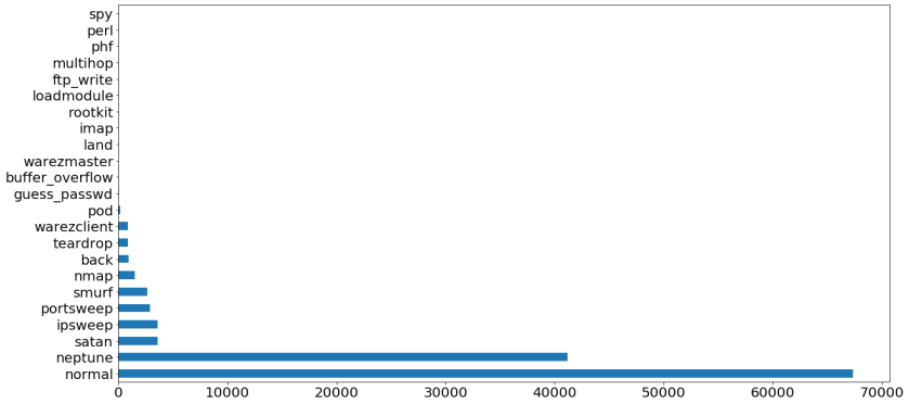
دعنا نلقي نظرة على البيانات الأولية للحصول على بعض الأفكار حول البيانات. دعونا نلقي نظرة على تصنيف الفئات أولاً:

```
dataset_root = 'datasets/nsl-kdd'
train_file = os.path.join(dataset_root, 'KDDTrain+.txt')
test_file = os.path.join(dataset_root, 'KDDTest+.txt')
header_names = ['duration', 'protocol_type', 'service', 'flag',
                'src_bytes', 'dst_bytes', 'land', 'wrong_fragment',
                'urgent', 'hot', 'num_failed_logins', 'logged_in',
                'num_compromised', 'root_shell', 'su_attempted',
                'num_root', 'num_file_creations', 'num_shells',
                'num_access_files', 'num_outbound_cmds',
                'is_host_login', 'is_guest_login', 'count',
                'srv_count', 'serror_rate', 'srv_serror_rate',
```

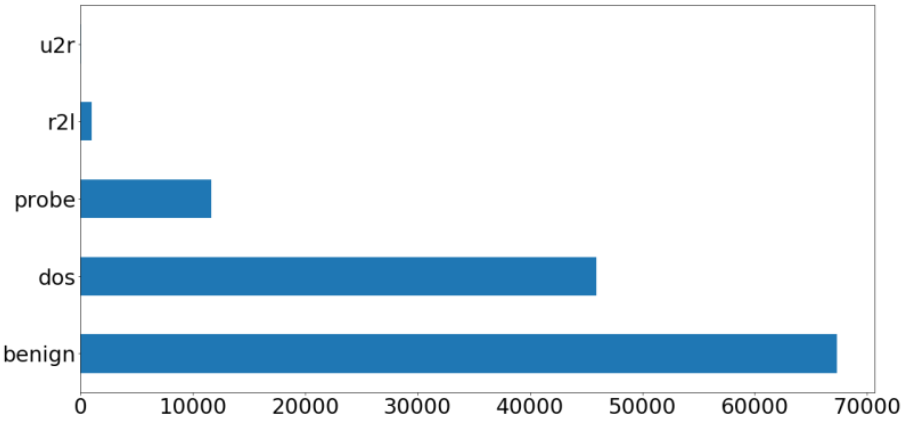
```
                'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
                'diff_srv_rate', 'srv_diff_host_rate',
                'dst_host_count', 'dst_host_srv_count',
                'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
                'dst host same src port rate',
'dst_host_srv_diff_host_rate',
                'dst_host_serror_rate', 'dst_host_srv_serror_rate',
                'dst_host_rerror_rate', 'dst_host_srv_rerror_rate',
                'attack_type', 'success_pred']
col_names = np.array(header_names)

nominal_idx = [1, 2, 3]
binary_idx = [6, 11, 13, 14, 20, 21]
numeric_idx =
list(set(range(41)).difference(nominal_idx).difference(binary_idx))

nominal_cols = col_names[nominal_idx].tolist()
binary_cols = col_names[binary_idx].tolist()
numeric_cols = col_names[numeric_idx].tolist()
category = defaultdict(list)
category['benign'].append('normal')

with open('datasets/training_attack_types.txt', 'r') as f:
    for line in f.readlines():
        attack, cat = line.strip().split(' ')
        category[cat].append(attack)

attack_mapping = dict((v,k) for k in category for v in category[k])
```

الآن، هذه هي البيانات التي سنستخدمها:

```
train_df = pd.read_csv(train_file, names=header_names)
train_df['attack_category'] = train_df['attack_type'] \
                                .map(lambda x: attack_mapping[x])
train_df.drop(['success_pred'], axis=1, inplace=True)

test_df = pd.read_csv(test_file, names=header_names)
test_df['attack_category'] = test_df['attack_type'] \
                                .map(lambda x: attack_mapping[x])
test_df.drop(['success_pred'], axis=1, inplace=True)
train_attack_types = train_df['attack_type'].value_counts()
train attack cats = train df['attack category'].value counts()
test_attack_types = test_df['attack_type'].value_counts()
test_attack_cats = test_df['attack_category'].value_counts()
train_attack_types.plot(kind='barh', figsize=(20,10), fontsize=20)
```



```
train_attack_cats.plot(kind='barh', figsize=(20,10), fontsize=30)
```



تحضير البيانات

تعد مجموعة بيانات NSL-KDD مجموعة بيانات مفيدة للتعليم والتجريب باستخدام التنقيب عن البيانات وتصنيف التعلم الآلي لأنها تحقق توازنًا بين البساطة والتطور.

لنبدأ بتقسيم الاختبار والتدريب DataFrames إلى بيانات وتسميات:

```
train_Y = train_df['attack_category']
train_x_raw = train_df.drop(['attack_category','attack_type'], axis=1)
test_Y = test_df['attack_category']
test_x_raw = test_df.drop(['attack_category','attack_type'], axis=1)
```

في الحالات النموذجية، سيكون لدينا معرفة كاملة بجميع المتغيرات الفئوية إما لأننا حددناها أو لأن مجموعة البيانات قدمت هذه المعلومات.في حالة تحليل أمان الشبكة هذه، لا يتم تزويد مجموعة البيانات بقائمة من القيم المحتملة لكل متغير فئوي، لذلك يمكننا المعالجة المسبقة على النحو التالي:

```
combined_df_raw = pd.concat([train_x_raw, test_x_raw])
combined_df = pd.get_dummies(combined_df_raw, columns=nominal_cols,
drop_first=True)

train_x = combined_df[:len(train_x_raw)]
test_x = combined_df[len(train_x_raw):]

# Store dummy variable feature names
dummy_variables = list(set(train_x)-set(combined_df_raw))
```

الآن دعنا نطبق الخوارزمية القياسية على هذه البيانات لتوسيع نطاق مجموعة البيانات:

```
# Experimenting with StandardScaler on the single 'duration' feature
from sklearn.preprocessing import StandardScaler

durations = train_x['duration'].values.reshape(-1, 1)
standard_scaler = StandardScaler().fit(durations)
scaled_durations = standard_scaler.transform(durations)
pd.Series(scaled_durations.flatten()).describe()
```

```
count    1.259730e+05
mean     2.549477e-17
std      1.000004e+00
min     -1.102492e-01
25%     -1.102492e-01
50%     -1.102492e-01
75%     -1.102492e-01
max      1.636428e+01
dtype: float64
```

يمكنك اختيار استخدام MinMaxScaler على StandardScaler إذا كنت تريد أن تحافظ عملية التحجيم على الانحرافات المعيارية الصغيرة للسلسلة الأصلية، أو إذا كنت تريد الاحتفاظ بإدخالات صفريةفي البيانات المتفرقة. إليك كيفية تحويل MinMaxScaler لدالة duration function:

```
from sklearn.preprocessing import MinMaxScaler

min_max_scaler = MinMaxScaler().fit(durations)
min_max_scaled_durations = min_max_scaler.transform(durations)
pd.Series(min_max_scaled_durations.flatten()).describe()
```

```
count    125973.000000
mean          0.006692
std           0.060700
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           1.000000
dtype: float64
```

يمكن أن تؤدي القيم المتطرفةفي بياناتك إلى تحريف خطير وسلبي لنتائج القياس القياسي والتسوية. إذا كانت البيانات تحتوي على قيم متطرفة، فسيكون sklearn.preprocessing.RobustScaler أكثر ملاءمة لمشكلة أمان الشبكة هذه. يستخدم RobustScaler تقديرات قوية مثل النطاقات المتوسطة والربيعية، لذلك لن يتأثر كثيرًا بالقيم المتطرفة:

```
from sklearn.preprocessing import RobustScaler

min_max_scaler = RobustScaler().fit(durations)
robust_scaled_durations = min_max_scaler.transform(durations)
pd.Series(robust_scaled_durations.flatten()).describe()
```

```
count    125973.00000
mean        287.14465
std        2604.51531
min           0.00000
25%           0.00000
50%           0.00000
75%           0.00000
max       42908.00000
dtype: float64
```

نكمل مرحلة المعالجة المسبقة للبيانات من خلال توحيد بيانات التدريب والاختبار:

```
standard_scaler = StandardScaler().fit(train_x[numeric_cols])

train_x[numeric_cols] = \
    standard_scaler.transform(train_x[numeric_cols])

test_x[numeric_cols] = \
    standard_scaler.transform(test_x[numeric_cols])
```

تصنيف لتحليل أمن الشبكة

من خلال تطبيق معلمات أفضل تخمين افتراضية أو أولية على الخوارزمية، يمكننا الحصول بسرعة على نتائج التصنيف الأولية لأمان الشبكة. على الرغم من أن هذه النتائج قد لا تكون قريبة من دقة هدفنا، فإنها عادة ما تعطينا مؤشرًا تقريبيًا على إمكانات الخوارزمية.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, zero_one_loss
classifier = DecisionTreeClassifier(random_state=0)
classifier.fit(train_x, train_Y)
pred_y = classifier.predict(test_x)
results = confusion_matrix(test_Y, pred_y)
error = zero_one_loss(test_Y, pred_y)
```

```
[[9365   56  289    1    0]
 [1541 5998   97    0    0]
 [ 675  220 1528    0    0]
 [2278    1   14  277    4]
 [ 179    0    5    5   11]]
0.238245209368
```

مع وجود بضعة أسطر فقط من التعليمات البرمجية وعدم وجود تعديل على الإطلاق، فإن دقة التصنيف التي تبلغ 76.2٪ (1 ـ معدل الخطأ)في مشكلة تصنيف من خمس فئات ليست سيئة للغاية. بهذه الطريقة يمكننا استخدام تحليل أمان الشبكة لتحسين أمان شبكاتنا.

48) الكشف عن الرسائل القصيرة غير المرغوب فيها باستخدام التعلم الآلي SMS Spam Detection using Machine Learning

تستند هذه المقالة إلى تصنيف اكتشاف الرسائل القصيرة غير المرغوب فيها باستخدام التعلم الآلي. سأستخدم تنفيذ multinomial Naive Bayes.

هذا المصنف الخاص مناسب للتصنيف بميزات متقطعة (كمافي حالتنا، عدد الكلمات لتصنيف النص). يأخذ عددًا صحيحًا من الكلمات كمدخلات.

من ناحية أخرى، يعد Gaussian Naive Bayes أكثر ملاءمة للبيانات المستمرة لأنه يفترض أن بيانات الإدخال لها توزيع غاوسي (عادي).

كشف الرسائل القصيرة غير المرغوب فيها

لنبدأ باستيراد المكتبات:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import nltk
```

تنزيل وقراءة مجموعة البيانات من هنا.

```
import pandas
df_sms = pd.read_csv('spam.csv',encoding='latin-1')
df_sms.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

إسقاط الأعمدة غير المرغوب فيها لم يذكر اسمه: Unnamed: 2 ، 3 ، 4:

```
df_sms = df_sms.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"],
axis=1)
df_sms = df_sms.rename(columns={"v1":"label", "v2":"sms"})
df_sms.head()
```

	label	sms
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

التحقق من الحد الأقصى لطول الرسائل القصيرة

```
print(len(df_sms))
```

عدد الملاحظاتفي كل تسمية البريد العشوائي spam وham.

```
df_sms.label.value_counts()
```

```
ham     4825
spam     747
Name: label, dtype: int64
```

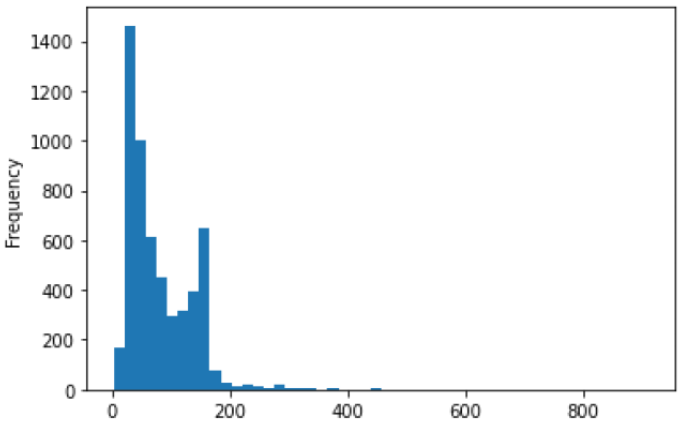
```
df_sms.describe()
```

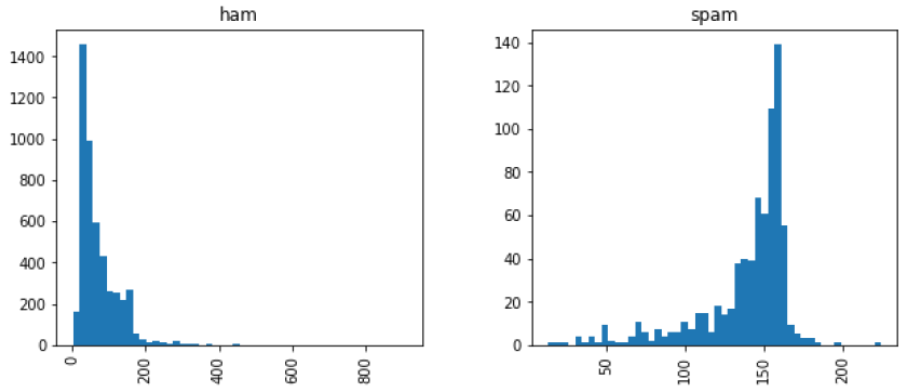
```
        label   sms
count   5572    5572
unique     2    5169
top      ham    Sorry, I'll call later
freq    4825    30
```

```
df_sms['length'] = df_sms['sms'].apply(len)
df_sms.head()
```

	label	sms	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
df_sms['length'].plot(bins=50, kind='hist')
```






```
df_sms.hist(column='length', by='label', bins=50,figsize=(10,4))
df_sms.loc[:,'label'] = df_sms.label.map({'ham':0, 'spam':1})
print(df_sms.shape)
df_sms.head()
```

```
(5572, 3)
   label    sms                                                length
0  0        Go until jurong point, crazy.. Available only ...  111
1  0        Ok lar... Joking wif u oni...                      29
2  1        Free entry in 2 a wkly comp to win FA Cup fina...  155
3  0        U dun say so early hor... U c already then say...  49
4  0        Nah I don't think he goes to usf, he lives aro...  61
```

نهج حقيبة الكلمات

ما لدينا هنافي مجموعة البيانات الخاصة بنا هو مجموعة كبيرة من البيانات النصية (5572 صفًا من البيانات). تعتمد معظم خوارزميات التعلم الآلي على البيانات الرقمية لتغذيتها كمدخلات، وعادة ما تكون رسائل البريد الإلكتروني / الرسائل القصيرة نصية ثقيلة.

نحتاج إلى طريقة لتمثيل البيانات النصية لخوارزمية التعلم الآلي ويساعدنا نموذج حقيبة الكلمات bag-of-words على تحقيق هذه المهمة. إنها طريقة لاستخراج الميزات من النص لاستخدامهافي خوارزميات التعلم الآلي.

في هذا النهج، نستخدم الكلمات المميزة لكل ملاحظة ونكتشف مدى تكرار كل رمز مميز token.

باستخدام عملية سنمر بها الآن، يمكننا تحويل مجموعة من المستندات إلى مصفوفة، حيث يكون كل مستند صفًا وكل كلمة (رمز مميز) هي العمود، والقيم المقابلة (الصف، العمود) هي تكرار حدوث كل كلمة أو رمز مميزفي ذلك المستند.

فمثلا:

لنفترض أن لدينا 4 مستندات على النحو التالي:

['Hello, how are you!', 'Win money, win from home.', 'Call me now', 'Hello, Call you tomorrow?']

هدفنا هنا هو تحويل هذه المجموعة من النصوص إلى مصفوفة توزيع التردد، على النحو التالي:

	are	call	from	hello	home	how	me	money	now	tomorrow	win	you
0	1	0	0	1	0	1	0	0	0	0	0	1
1	0	0	1	0	1	0	0	1	0	0	2	0
2	0	1	0	0	0	0	1	0	1	0	0	0
3	0	1	0	1	0	0	0	0	0	1	0	1

هنا كما نرى، يتم ترقيم المستنداتفي الصفوف، وكل كلمة هي اسم عمود، والقيمة المقابلة هي تكرار تلك الكلمةفي المستند.

لنفصل هذا ونرى كيف يمكننا إجراء هذا التحويل باستخدام مجموعة صغيرة من المستندات.

للتعامل مع هذا، سنستخدم طريقة sklearns count vectorizer التي تقوم بما يلي:

- يقوم بترميز السلسلة (يفصل السلسلة إلى كلمات فردية) ويعطي معرفًا صحيحًا لكل رمز مميز.
- تحسب حدوث كل من هذه الرموز المميزة.

تطبيق منهج حقيبة الكلمات

الخطوة 1: تحويل جميع السلاسل إلى شكلها الصغير.

```
documents = ['Hello, how are you,'!
'              Win money, win from home,'.
'              Call me now,'.
'              Hello, Call hello you tomorrow['?

lower_case_documents[] =
lower case documents = [d.lower() for d in documents]
print(lower_case_documents)
```

```
['hello, how are you!', 'win money, win from home.', 'call me now.', 'hello, call hello you tomorrow?']
```

الخطوة 2: إزالة جميع علامات الترقيم.

```
sans_punctuation_documents[] =
import string

for i in lower_case_documents:
```

```
    sans punctuation documents.append(i.translate(str.maketrans("","",
string.punctuation)))

sans_punctuation_documents
```

```
['hello how are you',
 'win money win from home',
 'call me now',
 'hello call hello you tomorrow']
```

الخطوة 3: الترميز Tokenization

```
preprocessed_documents = [[w for w in d.split()] for d in
sans_punctuation_documents]
preprocessed_documents
```

```
[['hello', 'how', 'are', 'you'],
 ['win', 'money', 'win', 'from', 'home'],
 ['call', 'me', 'now'],
 ['hello', 'call', 'hello', 'you', 'tomorrow']]
```

الخطوة 4: عد التكرارات.

```
frequency_list[] =
import pprint
from collections import Counter

frequency_list = [Counter(d) for d in preprocessed_documents]
pprint.pprint(frequency_list)
```

```
[Counter({'hello': 1, 'how': 1, 'are': 1, 'you': 1}),
 Counter({'win': 2, 'money': 1, 'from': 1, 'home': 1}),
 Counter({'call': 1, 'me': 1, 'now': 1}),
 Counter({'hello': 2, 'call': 1, 'you': 1, 'tomorrow': 1})]
```

تنفيذ حقيبة الكلمات في scikit-Learn

سننظر هنافي إنشاء مصفوفة تكرار frequency matrix على مجموعة مستندات أصغر للتأكد من فهمنا لكيفية حدوث إنشاء مصفوفة مصطلح المستند. لقد أنشأنا نموذجًا لمجموعة المستندات "documents".

documents = [‘Hello, how are you!’, ‘Win money, win from home.’, ‘Call me now.’, ‘Hello, Call hello you tomorrow?’]

```
from sklearn.feature_extraction.text import CountVectorizer
count_vector = CountVectorizer()
```

المعالجة المسبقة للبيانات باستخدام CountVectorizer()

في الخطوة أعلاه، قمنا بتنفيذ نسخة من طريقة CountVectorizer() من البداية والتي استلزم تنظيف بياناتنا أولاً.

تضمن هذا التنظيف تحويل جميع بياناتنا إلى أحرف صغيرة وإزالة جميع علامات الترقيم.

يحتوي CountVectorizer() على معلمات معينة تهتم بهذه الخطوات بالنسبة لنا. هم انهم:

lowercase = True

تحتوي المعلمة الصغيرة على قيمة افتراضية لـ True والتي تحول كل نصنا إلى شكله بأحرف صغيرة.

token_pattern = (?u)\b\w\w+\b

تحتوي المعلمة token_pattern على قيمة تعبير عادي افتراضية هي (؟ u) \ b \ w \ w + \ b والتي تتجاهل جميع علامات الترقيم وتعاملها كمحددات، بينما تقبل سلاسل أبجدية رقمية بطول أكبر من أو تساوي 2، كرموز فردية أو كلمات.

stop_words

ستزيل المعلمة stop_words، إذا تم ضبطها على اللغة الإنجليزية، جميع الكلمات من مجموعة المستندات التي تطابق قائمة كلمات التوقف الإنجليزية المحددةفي scikit-Learn.

بالنظر إلى حجم مجموعة البيانات الخاصة بنا وحقيقة أننا نتعامل مع رسائل SMS وليس مع مصادر نصية أكبر مثل البريد الإلكتروني، فلن نقوم بتعيين قيمة المعلمة هذه.

```
count_vector.fit(documents)
count_vector.get_feature_names()
```

```
['are',
 'call',
 'from',
 'hello',
 'home',
 'how',
 'me',
 'money',
 'now',
 'tomorrow',
 'win',
 'you']
```

```
doc_array = count_vector.transform(documents).toarray()
doc_array
```

```
array([[1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1],
       [0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 2, 0],
       [0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0],
       [0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 1]])
```

```
frequency_matrix = pd.DataFrame(doc_array, columns =
count_vector.get_feature_names())
frequency_matrix
```

	are	call	from	hello	home	how	me	money	now	tomorrow	win	you
0	1	0	0	1	0	1	0	0	0	0	0	1
1	0	0	1	0	1	0	0	1	0	0	2	0
2	0	1	0	0	0	0	1	0	1	0	0	0
3	0	1	0	2	0	0	0	0	0	1	0	1

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df_sms['sms ,['

df_sms['label'],test_size=0.20 ,
                                                    random_state=1)
```

```
 #Instantiate the CountVectorizer method
count_vector = CountVectorizer()

 #Fit the training data and then return the matrix
training_data = count_vector.fit_transform(X_train)

 #Transform testing data and return the matrix .
testing_data = count_vector.transform(X_test)
```

تنفيذ خوارزمية التعلم الآلي نايف بايز

سأستخدم sklearn.naive_bayes من sklearn لعمل تنبؤات على مجموعة البيانات الخاصة بنا لاكتشاف الرسائل القصيرة غير المرغوب فيها.

على وجه التحديد، سنستخدم تنفيذ multinomial Naive Bayes. هذا المصنف الخاص مناسب للتصنيف بميزات متقطعة. يأخذ عددًا صحيحًا من الكلمات كمدخلات.

من ناحية أخرى، يعد Gaussian Naive Bayes أكثر ملاءمة للبيانات المستمرة لأنه يفترض أن بيانات الإدخال لها توزيع غاوسي (عادي).

```
from sklearn.naive_bayes import MultinomialNB
naive_bayes = MultinomialNB()
naive_bayes.fit(training_data,y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
predictions = naive_bayes.predict(testing_data)
```

تقييم نموذجنا لاكتشاف الرسائل القصيرة غير المرغوب فيها

الآن بعد أن قمنا بعمل تنبؤات على مجموعة الاختبار الخاصة بنا، فإن هدفنا التالي هو تقييم مدى جودة أداء نموذجنا. هناك آليات مختلفة للقيام بذلك، ولكن دعنا أولاً نلخصها سريعًا.

تقيس Accuracy عدد المرات التي يقوم فيها المصنف بالتنبؤ الصحيح. إنها نسبة عدد التنبؤات الصحيحة إلى العدد الإجمالي للتنبؤات (عدد نقاط بيانات الاختبار).

تخبرنا Precision عن نسبة الرسائل التي صنفناها كرسائل غير مرغوب فيها، والتي كانتفي الواقع رسائل غير مرغوب فيها. إنها نسبة الإيجابيات الحقيقية (الكلمات المصنفة على أنها بريد عشوائي، والتي هيفي الواقع بريد عشوائي) إلى جميع الإيجابيات (جميع الكلمات مصنفة على أنها بريد عشوائي، بغض النظر عما إذا كان هذا هو التصنيف الصحيح)، بمعنى آخر هي نسبة:

[True Positives/(True Positives + False Positives)]

يخبرنا Recall(sensitivity) عن نسبة الرسائل التي كانتفي الواقع بريدًا عشوائيًا تم تصنيفها من قبلنا على أنها رسائل غير مرغوب فيها. إنها نسبة الإيجابيات الحقيقية (الكلمات المصنفة على أنها بريد عشوائي، والتي هيفي الواقع بريد عشوائي) إلى جميع الكلمات التي كانتفي الواقع بريدًا عشوائيًا، وبعبارة أخرى هي نسبة:

[True Positives/(True Positives + False Negatives)]

بالنسبة إلى مشاكل التصنيف المنحرفة skewed في توزيعات التصنيف كمافي حالتنا، على سبيل المثال، إذا كان لدينا 100 رسالة نصية وكانت رسالتان فقط من الرسائل غير المرغوب فيها والباقي 98 ، فإن accuracyفي حد ذاتها ليست مقياسًا جيدًا.

يمكننا تصنيف 90 رسالة على أنها ليست رسائل غير مرغوب فيها (بمافي ذلك الرسالتان اللتان كانتا رسائل غير مرغوب فيها ولكننا نصنفها على أنها ليست رسائل غير مرغوب فيها، وبالتالي ستكون رسائل سلبية خاطئة) و10 رسائل كرسائل غير مرغوب فيها (جميعها إيجابية كاذبة) وما زلنا نحصل على درجة accuracy جيدة إلى حد معقول.

في مثل هذه الحالات، تكون precision وrecallفي متناول اليد. يمكن الجمع بين هذين المقياسين للحصول على F1 score، وهي متوسط مرجح لدرجات precision وrecall. يمكن أن تتراوح هذه النتيجة من 0 إلى 1، حيث يمثل 1 أفضل نتيجة ممكنة لـ F1.

سنستخدم جميع المقاييس الأربعة للتأكد من أن نموذجنا يعمل بشكل جيد. بالنسبة لجميع المقاييس الأربعة التي يمكن أن تتراوح قيمها من 0 إلى 1، فإن الحصول على درجة قريبة من 1 قدر الإمكان يعد مؤشرًا جيدًا على مدى جودة أداء نموذجنا.

```
from sklearn.metrics import accuracy score, precision score,
recall_score, f1_score
print('Accuracy score: {}'.format(accuracy_score(y_test,
predictions)))
print('Precision score: {}'.format(precision_score(y_test,
predictions)))
print('Recall score: {}'.format(recall score(y test, predictions)))
print('F1 score: {}'.format(f1_score(y_test, predictions)))
```

```
1 Accuracy score: 0.9847533632286996
2 Precision score: 0.9420289855072463
3 Recall score: 0.935251798561151
4 F1 score: 0.9386281588447652
```

49) تحليل أداء الطالب باستخدام التعلم الآلي Student Performance Analysis using Machine Learning

يتطلب الأمر الكثير من الجهد اليدوي لإكمال عملية التقييم حيث قد تحتوي كلية واحدة على آلاف الطلاب.

في مشروع علوم البيانات هذا، سنقوم بتقييم أداء الطالب باستخدام تقنيات التعلم الآلي وpython.

يمكنك تنزيل مجموعة البيانات التي تحتاجها لهذا المشروع من هنا.

لنبدأ باستيراد المكتبات:

```
 #for some basic operations
import numpy as np
import pandas as pd

 #for visualizations
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import dabl
```

لقراءة مجموعة البيانات:

```
data = pd.read_csv('StudentsPerformance.csv')

 #getting the shape of the data
print(data.shape)
```

```
(1000, 8)
```

لإلقاء نظرة على أول 5 سجلاتفي مجموعة البيانات:

```
data.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

الإحصاء الوصفي

```
data.describe()
```

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

دعنا نتحقق من عدد العناصر الفريدة الموجودةفي العمود الفئوي:

```
data.select_dtypes('object').nunique()
```

```
gender                       2
race/ethnicity               5
parental level of education  6
lunch                        2
test preparation course      2
dtype: int64
```

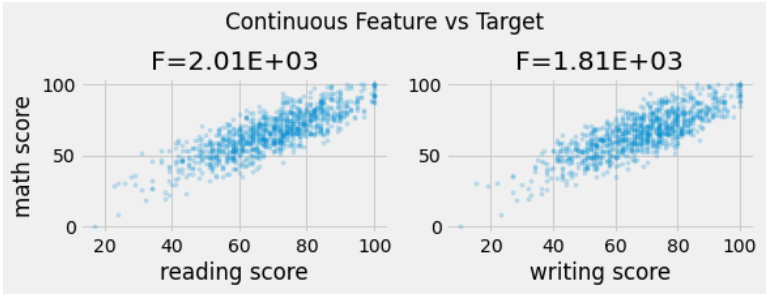
يتيح التحقق من النسبة المئوية للبيانات المفقودةفي كل أعمدة موجودةفي البيانات:

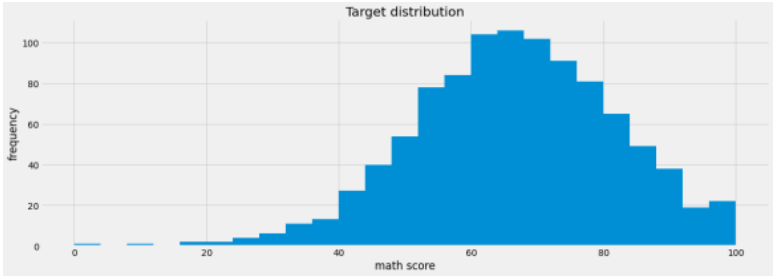
```
no_of_columns = data.shape[0]
percentage_of_missing_data = data.isnull().sum()/no_of_columns
print(percentage_of_missing_data)
```

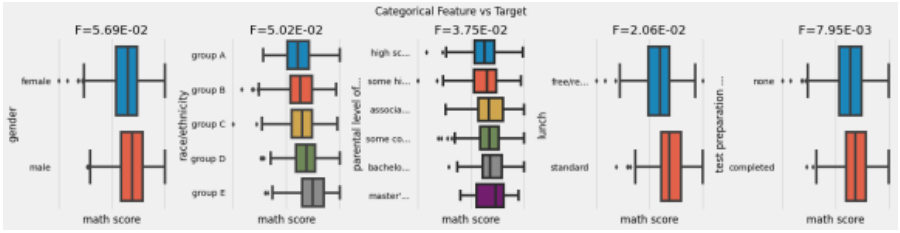
```
gender                       0.0
race/ethnicity               0.0
parental level of education  0.0
lunch                        0.0
test preparation course      0.0
math score                   0.0
reading score                0.0
writing score                0.0
dtype: float64
```

لمشاهدة مقارنة بين جميع الصفات الأخرى فيما يتعلق بدرجات الرياضيات Math Marks:

```
plt.rcParams['figure.figsize'](6 ,18) =
plt.style.use('fivethirtyeight')
dabl.plot(data, target_col = 'math score')
```

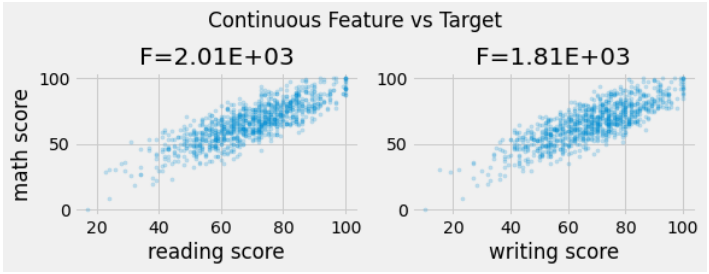


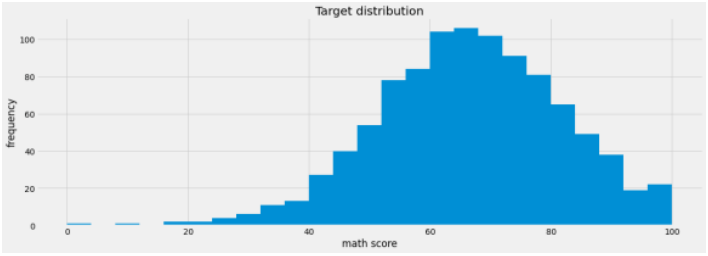


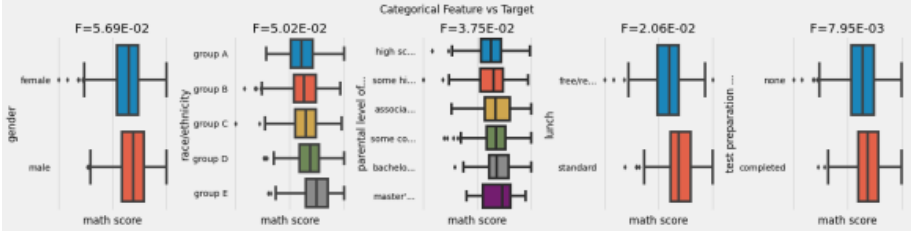


مقارنة بين جميع الصفات الأخرى فيما يتعلق بدرجات القراءة Reading Marks:

```
plt.rcParams['figure.figsize'](6 ,18) =
plt.style.use('fivethirtyeight')
dabl.plot(data, target_col = 'reading score')
```

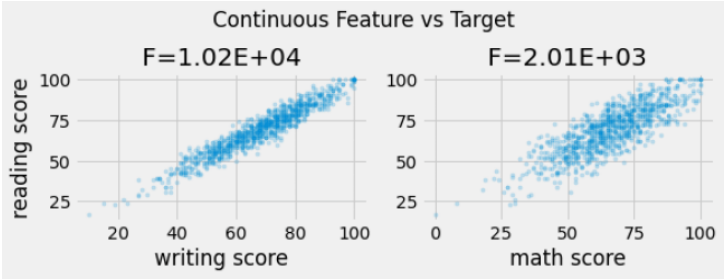


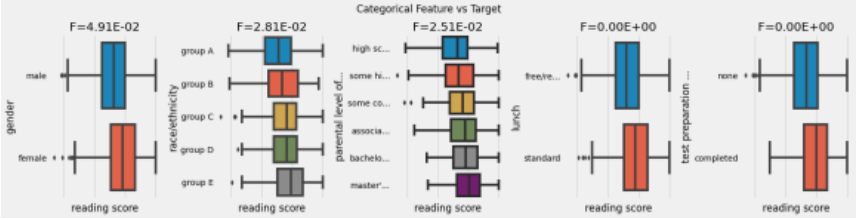




مقارنة بين جميع الصفات الأخرى فيما يتعلق بدرجات القراءة:

```
plt.rcParams['figure.figsize'](6 ,18) =
plt.style.use('fivethirtyeight')
dabl.plot(data, target_col = 'reading score')
```





دعونا نتحقق من تأثير الغداء Lunch على أداء الطلاب:

```
data[['lunch','gender','math score','writing score,'
'       reading score']].groupby(['lunch','gender']).agg('median')
```

		math score	writing score	reading score
lunch	gender			
free/reduced	female	57.0	68.0	67.0
	male	62.0	59.0	61.0
standard	female	67.0	76.0	75.0
	male	72.0	67.0	67.5

دعنا نتحقق من تأثير دورة التحضير للاختبار Test Preparation Course على النتائج:

```
ata[['test preparation course,'
'       gender,'
'       math score,'
'       writing score,'
```

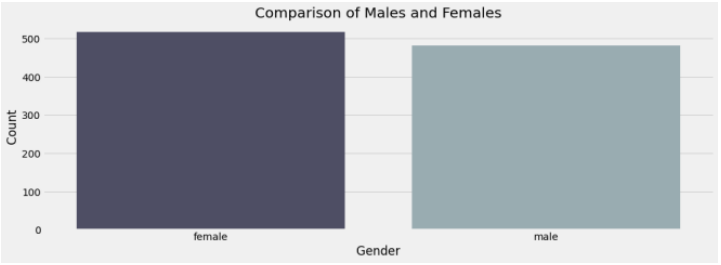
```
'       reading score']].groupby(['test preparation
course','gender']).agg('median')
```

test preparation course	gender			
completed	female	67	79	78
	male	73	70	71
none	female	62	70	71
	male	67	60	63

التمثيل المرئي البيانات

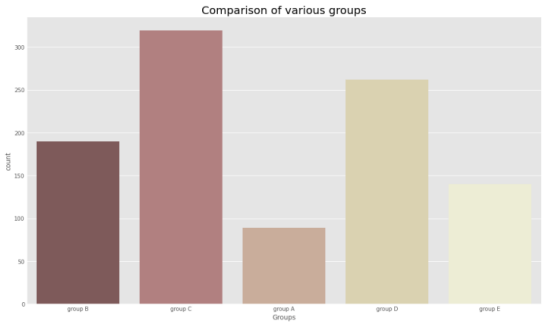
رسم عدد الذكور والإناثفي مجموعة البيانات:

```
plt.rcParams['figure.figsize'](5 ,15) =
sns.countplot(data['gender'], palette = 'bone')
plt.title('Comparison of Males and Females', fontweight = 30)
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```



رسم المجموعات المختلفةفي مجموعة البيانات:

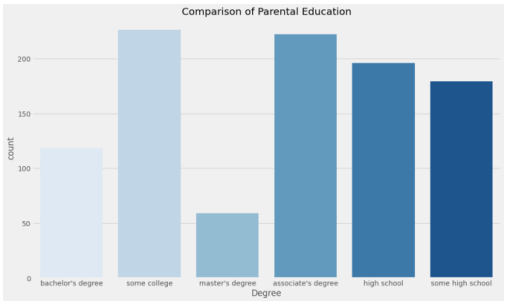
```
plt.rcParams['figure.figsize'](9 ,15) =
plt.style.use('ggplot')

sns.countplot(data['race/ethnicity'], palette = 'pink')
plt.title('Comparison of various groups', fontweight = 30, fontsize =
20)
plt.xlabel('Groups')
plt.ylabel('count')
plt.show()
```



رسم مستويات تعليم الوالدين المختلفة:

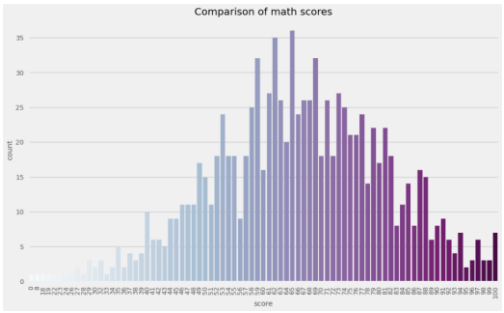
```
plt.rcParams['figure.figsize'](9 ,15) =
```

```
plt.style.use('fivethirtyeight')

sns.countplot(data['parental level of education'], palette = 'Blues')
plt.title('Comparison of Parental Education', fontweight = 30,
fontsize = 20)
plt.xlabel('Degree')
plt.ylabel('count')
plt.show()
```



رسم درجات الرياضيات:

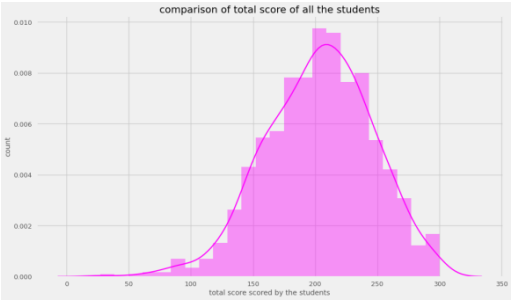
```
plt.rcParams['figure.figsize'](9 ,15) =
plt.style.use('tableau-colorblind10')

sns.countplot(data['math score'], palette = 'BuPu')
plt.title('Comparison of math scores', fontweight = 30, fontsize = 20)
plt.xlabel('score')
plt.ylabel('count')
plt.xticks(rotation = 90)
plt.show()
```



حساب الدرجة الإجمالية لكل طالب:

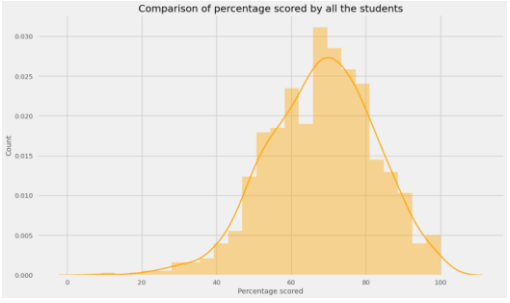
```
import warnings
warnings.filterwarnings('ignore')

data['total_score'] = data['math score'] + data['reading score'] +
data['writing score']

sns.distplot(data['total_score'], color = 'magenta')

plt.title('comparison of total score of all the students', fontweight
= 30, fontsize = 20)
plt.xlabel('total score scored by the students')
plt.ylabel('count')
```

```
plt.show()
```



حساب النسبة لكل طالب:

```
 #importing math library to use ceil
from math import *
import warnings
warnings.filterwarnings('ignore')

data['percentage'] = data['total_score']/3

for i in range:(1000 ,0)
    data['percentage'][i] = ceil(data['percentage'][i])

plt.rcParams['figure.figsize'](9 ,15) =
sns.distplot(data['percentage'], color = 'orange')

plt.title('Comparison of percentage scored by all the students',
fontweight = 30, fontsize = 20)
plt.xlabel('Percentage scored')
plt.ylabel('Count')
plt.show()
```



تعيين الدرجات للتقديرات وفقًا للمعايير التالية:

```
0  - 40 marks : grade E
41 - 60 marks : grade D
60 - 70 marks : grade C
70 - 80 marks : grade B
80 - 90 marks : grade A
90 - 100 marks : grade O
```

```
def getgrade(percentage, status):
  if status == 'Fail:'
    return 'E'
  if(percentage >= 90):
    return 'O'
  if(percentage >= 80):
    return 'A'
  if(percentage >= 70):
    return 'B'
  if(percentage >= 60):
    return 'C'
  if(percentage >= 40):
    return 'D'
  else:
    return 'E'

data['grades'] = data.apply(lambda x: getgrade(x['percentage'],
x['status']), axis = 1 )

data['grades'].value_counts()
```

```
#Output
B    260
C    252
D    223
A    156
O     58
E     51
Name: grades, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder

 #creating an encoder
le = LabelEncoder()

 #label encoding for test preparation course
data['test preparation course'] = le.fit_transform(data['test
preparation course'])

 #label encoding for lunch
data['lunch'] = le.fit_transform(data['lunch'])

 #label encoding for race/ethnicity
 #we have to map values to each of the categories
data['race/ethnicity'] = data['race/ethnicity'].replace('group A', 1)
data['race/ethnicity'] = data['race/ethnicity'].replace('group B', 2)
data['race/ethnicity'] = data['race/ethnicity'].replace('group C', 3)
data['race/ethnicity'] = data['race/ethnicity'].replace('group D', 4)
data['race/ethnicity'] = data['race/ethnicity'].replace('group E', 5)

 #label encoding for parental level of education
data['parental level of education'] = le.fit_transform(data['parental
level of education'])

#label encoding for gender
data['gender'] = le.fit transform(data['gender'])

 #label encoding for pass_math
```

```
data['pass_math'] = le.fit_transform(data['pass_math'])

 #label encoding for pass_reading
data['pass_reading'] = le.fit_transform(data['pass_reading'])

 #label encoding for pass_writing
data['pass_writing'] = le.fit_transform(data['pass_writing'])

 #label encoding for status
data['status'] = le.fit_transform(data['status'])
```

تحضير البيانات

تقسيم المتغيرات التابعة والمستقلة

```
x = data.iloc[14:,:]
y = data.iloc[14,:]

print(x.shape)
print(y.shape)
```

```
#Output
(1000, 14)
(1000,)
```

تقسيم مجموعة البيانات إلى مجموعات تدريب واختبار

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.25, random_state = 45)

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
#Output
(750, 14)
(750,)
(250, 14)
(250,)
```

```
 #importing the MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

 #creating a scaler
mm = MinMaxScaler()

 #feeding the independent variable into the scaler
x_train = mm.fit_transform(x_train)
x_test = mm.transform(x_test)
```

تطبيق تحليل المكونات الرئيسية (PCA)

```
from sklearn.decomposition import PCA

 #creating a principal component analysis model
```

```
pca = PCA(n_components = None)

 #feeding the independent variables to the PCA model
x_train = pca.fit_transform(x_train)
x test = pca.transform(x test)

 #visualising the principal components that will explain the highest
share of variance
explained_variance = pca.explained_variance_ratio_
print(explained_variance)

 #creating a principal component analysis model
pca = PCA(n_components = 2)

 #feeding the independent variables to the PCA model
x_train = pca.fit_transform(x_train)
x_test = pca.transform(x_test)
```

النمذجة

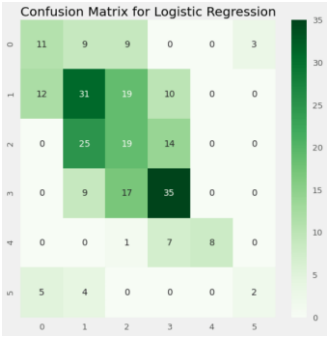
الانحدار اللوجستي

```
from sklearn.linear_model import  LogisticRegression

 #creating a model
model = LogisticRegression()

 #feeding the training data to the model
model.fit(x_train, y_train)

 #predicting the test set results
y_pred = model.predict(x_test)

 #calculating the classification accuracies
print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))
```

```
Output-
Training Accuracy : 0.3933333333333333
Testing Accuracy : 0.424
```

طباعة مصفوفة الارتباك

```
from sklearn.metrics import confusion_matrix

 #creating a confusion matrix
cm = confusion_matrix(y_test, y_pred)

 #printing the confusion matrix
plt.rcParams['figure.figsize'](8 ,8) =
sns.heatmap(cm, annot = True, cmap = 'Greens')
plt.title('Confusion Matrix for Logistic Regression', fontweight = 30,
fontsize = 20)
plt.show()
```

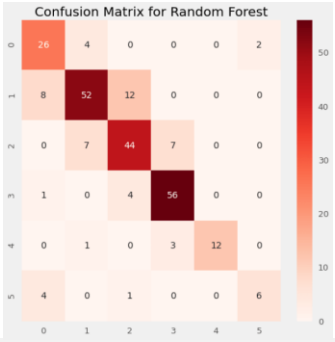


الغابة العشوائية

```
from sklearn.ensemble import RandomForestClassifier

 #creating a model
model = RandomForestClassifier()

 #feeding the training data to the model
model.fit(x_train, y_train)

 #predicting the x-test results
y_pred = model.predict(x_test)

 #calculating the accuracies
print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))
```

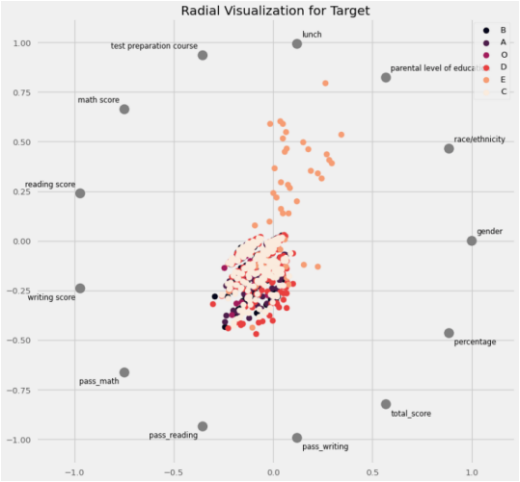
```
Output-
Training Accuracy : 0.9986666666666667
Testing Accuracy : 0.784
```

```
from sklearn.metrics import confusion_matrix

 #creating a confusion matrix
cm = confusion_matrix(y_test, y_pred)

 #printing the confusion matrix
plt.rcParams['figure.figsize'](8 ,8) =
sns.heatmap(cm, annot = True, cmap = 'Reds')
plt.title('Confusion Matrix for Random Forest', fontweight = 30,
fontsize = 20)
plt.show()
```




```
from pandas.plotting import radviz
```

```
fig, ax = plt.subplots(figsize=(12, 12))
new_df = x.copy()
new_df["status"] = y
radviz(new_df, "status", ax=ax, colormap="rocket")
plt.title('Radial Visualization for Target', fontsize = 20)
plt.show()
```



إنه يعطي فكرة واضحة أن الطلاب الذين يحصلون على درجات منخفضة جدًا لديهم ارتباط كبير بالغذاء وتعليم الوالدين.

المصادر

1. 180 Data Science and Machine Learning Projects with Python, Aman Kharwal, https://medium.com/coders-camp/180-data-science-and-machine-learning-projects-with-python-6191bc7b9db9.
2. 12 Machine Learning Projects on Object Detection, Aman Kharwal , https://amankharwal.medium.com/12-machine-learning-projects-on-object-detection-46b32adc3c37.
3. 10 Machine Learning Projects on Time Series Forecasting Aman Kharwal, https://medium.com/coders-camp/10-machine-learning-projects-on-time-series-forecasting-ee0368420ccd.
4. 20 Machine Learning Projects on Future Prediction with Python, Kharwal, https://amankharwal.medium.com/20-machine-learning-projects-on-future-prediction-with-python-93932d9a7f7f.
5. 5 Machine Learning Projects for Healthcare, Kharwal, https://medium.datadriveninvestor.com/5-machine-learning-projects-for-healthcare-bbd0eac57b4a
6. Top 47 Machine Learning Projects for 2022 [Source Code Included], https://data-flair.training/blogs/machine-learning-project-ideas/.
7. Use Cases, https://cainvas.ai-tech.systems/

Machine Learning

By Example

50 Machine Learning Projects Solved and Explained with Python

By: Dr. Alaa Taima