ترجمة كورس من شركة IBM في موقع cognitiveclass.ai

عنوان الكورس

IBM Course with Python ML0101ENV3

يقدم هذا الموقع دروساً تعليمية عديدة و مجانية في مجال الذكاء الاصطناعي وتتميز ببساطة لغتها الإنكليزية وسهولة فهمها من قبل الطالب مع شرح مميز مدعم بالأمثلة وكلها مجانية و في نهاية كل منهج يؤدون لك امتحان مجاني ويمنحوك شهادة مجانية كذلك.

عند الانتهاء من هذا الكورس تحديدا خلال السنة الماضية وحصولي على شهادة فيه ولله الحمد أحببت أن أترجم هذا الكورس تحديدا وأضع ضمنه الأمثلة من باب زيادة المحتوى العربي لمن أراد أن يستفيد من ذلك والله ولي التوفيق من قبل و من بعد.

إعداد : م. ضياء الدين أحمد الأجرد

الفهرس

1- مقدمة في تعلم الآلة :

على فرض أن لدينا خلية مريضة من مريض ما و نريد أن نحدد هل هي ضمن ورم خبيث malignant أم سليم benign ، لذلك نطلع على خصائص هذه الخلية ثم يقرر الطبيب الذي لديه خبرة بقراءة هذه الخصائص أن يحدد حالتها .

ثم يكون لدينا بيانات كبيرة للعديد من المرضى (مئات أو الاف) نضعها فيما يسمى dataset كما يلي :

ID	Clean	Unifcell	Uni	….	….	Marg	….	….	calss
10234	6	1	2	…	…	3	…	…	benign
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
20350	7	5	1	…	…	1	…	…	malignant

يضم هذا الجدول مواصفات الخلايا الحالية لدينا من المرضى وحالتها هل خبيثة أم سليمة وعلى فرض أتتنا مواصفات خلية جديدة خارج هذه البيانات السابقة :

2356	4	2	1	…	…	1	…	…	؟

هنا يأتي دور النموذج المبني لغة الآلة ليحدد بدقة طبيعة هذه الخلية دون الحاجة لوجود الخبير أو يكون النموذج مؤكد لرأي الخبير ، وهذا ما نسميه بالتوقع Regression .

هناك مثال آخر في عمل البنوك عندما تريد أن تقرر إعطاء قرض loan للزبون أم لا (في عملية تسمى approved تثبيت من عدمه) ، وأيضا مواقع الانترنت عندما تقدم لزبائنها المنتجات التي يفضلونها .

إذا تعريف تعلم الآلة : يعطي الحاسوب إمكانية التعلم دون وجود برمجة صريحة في اتخاذه القرار .

Machine Learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed .

لدينا كذلك مثال آخر : ليكن لدينا مجموعة صور حيوانات و نريد التمييز فيما بينها ، عند ذلك نحدد الخصائص لهذه الحيوانات (شكل العيون ، الأذان ، الذيل و طوله ، وجود الأجنحة وشكلها ..... و غيرها من الخصائص) ، ثم نضع قواعد rules عديدة if..then لتحديد نوع الحيوان ، و هنا لدينا احتمالان :

- إما نستخدم جميع الخصائص لتحديد كل الاحتمالات عن طريق الاستفادة من جميع الميزات features وتحديد نوع الحيوان (وهذا استهلاك كبير للذاكرة والوقت) .
- أو عند وضع مجموعة قواعد نستخدم فيها بعض الميزات فلن نستطيع عندها تحديد جميع الحالات .

كذلك شركات الاتصالات التي تستخدم معلومات مستخدميها الديموغرافية demographic لتقطيعهم segment وتحديد فيما إذا سيستمرون في الاشتراك لديها أو لا .

وأيضا تطبيقات Chabot في التحدث مع الحاسوب ، والألعاب games تستخدم تقنيات تعلم الآلة .

1- أهم تقنيات تعلم الآلة Major Machine learning techniques :

1-1- التوقع Regression / Estimation :

تستخدم هذه التقنية للتوقع في البيانات المستمرة continues values حيث تُستخدم مثلا لتوقع أسعار المنازل بناء على ميزاتها ، أو لتوقع حجم غاز co_2 المنبعث emission من السيارات بناء على مواصفات المحرك .

1-2- التصنيف Classification :

يستخدم لمعرفة الحالة المدروسة تندرج تحت أي صنف كما رأينا في مثال الخلايا هل خبيثة أو سليمة .

1-3- التجميع Clustering :

وضع الحالات المتشابهة بمواصفاتها ضمن مجموعات كتحديد مجموعات المرضى المتشابهون في أعراضهم ، أو تقسيم زبائن البنك .

4-1- الترابط Associations :

لإيجاد العناصر أو الأحداث التي تترابط في حدوثها مثلا المواد التي تُباع في البقالية (بشكل مترابط أي متعلقة ببعضها البعض) للزبون .

5-1- اكتشاف الاختلاف Anomaly detection :

لتحديد الحالات الغير عادية (إيجاد القيم الشاذة) مثل اكتشاف حالات التزوير fraud في بطاقات الائتمان و الصراف credit card fraud .

6-1- التنقيب المتتالي Sequence mining :

لإيجاد و توقع الحالة التالية : على سبيل المثال توقع تسلسل النقرات على الماوس في المواقع الإلكترونية click-stream .

7-1- تخفيض الأبعاد Dimension reduction :

نقوم بحذف بعض العناصر غير المهمة أو بعض الصفات features التي لسنا بحاجة لها لتخفيض حجم الداتا لدينا .

8-1- نظام التوصية Recommendation Systems :

ويكون متعلق بميزات الأشخاص المتشابهة ثم نقدم لهم الكتب المفضلة أو الموسيقا المفضلة لديهم .

2- استخدام البايثون في تعلم الآلة Python for ML :

تعتبر البايثون لغة قوية powerful ومتعددة الأغراض general-purpose فهي تحوي العديد من المكتبات والدوال للاستعمال في لغة الآلة :

Numpy : numerical library للتعامل مع المصفوفات متعددة الأبعاد

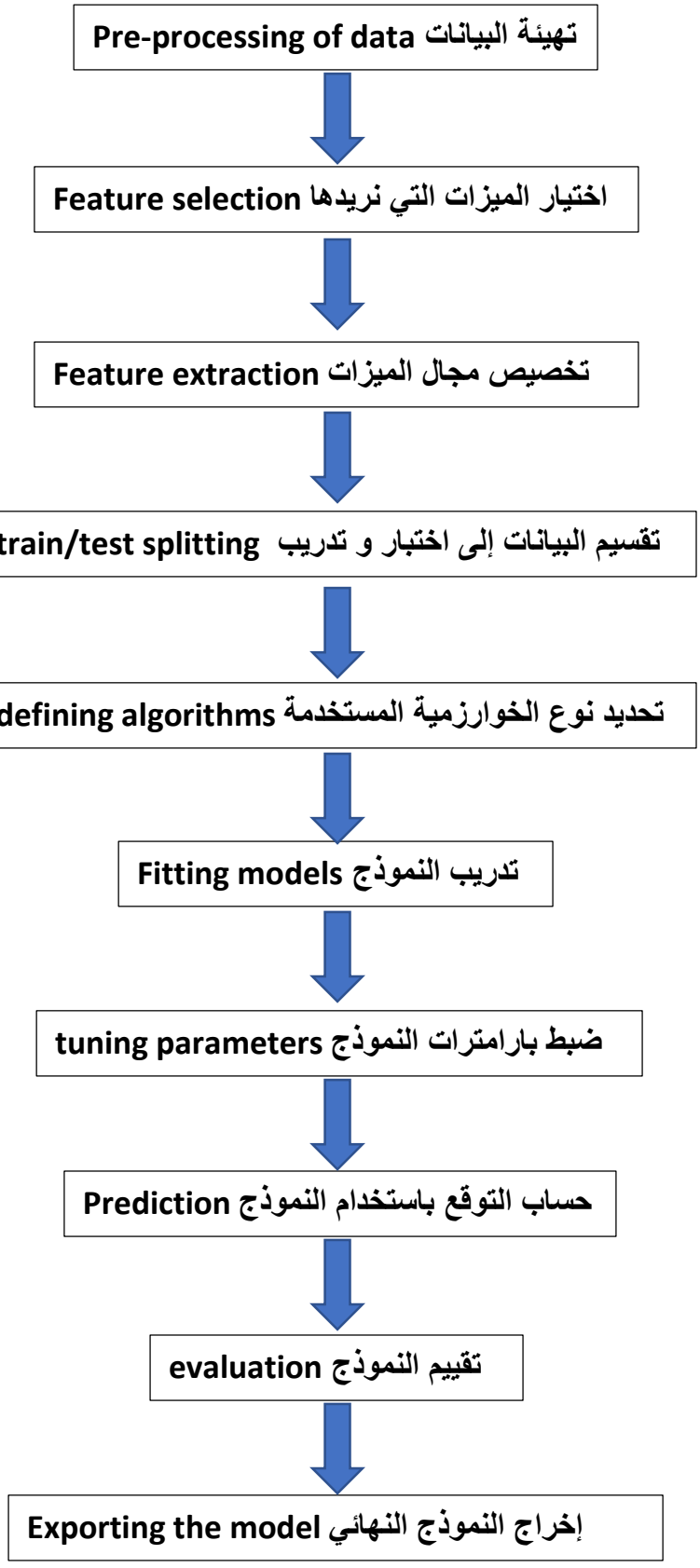
Scipy : scientific library for numerical algorithms & domain **specific** toolbox for signal processing – optimization – statistics

Matplotlib : للرسم ثنائي و ثلاثي الأبعاد

Pandas : for data structure هياكل البيانات , manipulation analysis تحليل وإخراج البيانات والتلاعب بطريقة عرضها بشكلها الرقمي وبشكل متسلسل مع الزمن .

أهم مكتبة تضم خوارزميات تعلم الآلة : **Scikit-learn**

حيث يتم العمل باستخدام أغلب المكتبات السابقة وفق التسلسل التالي :

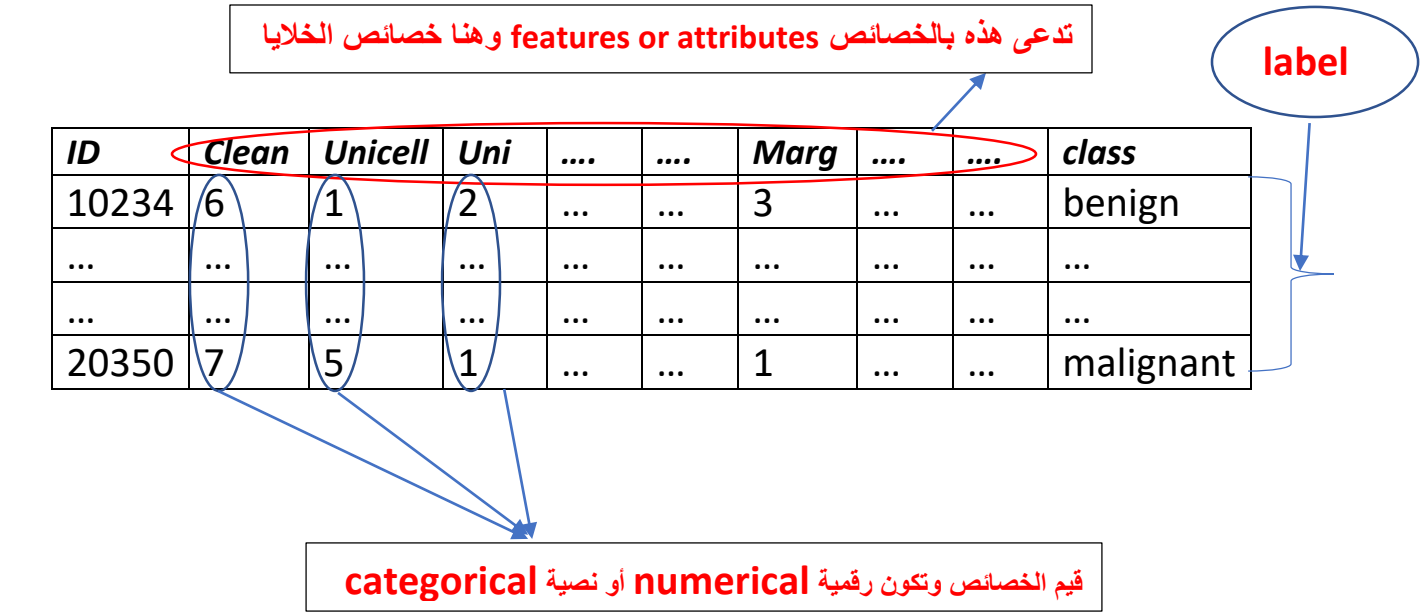


3- التعلم الإشرافي supervised :

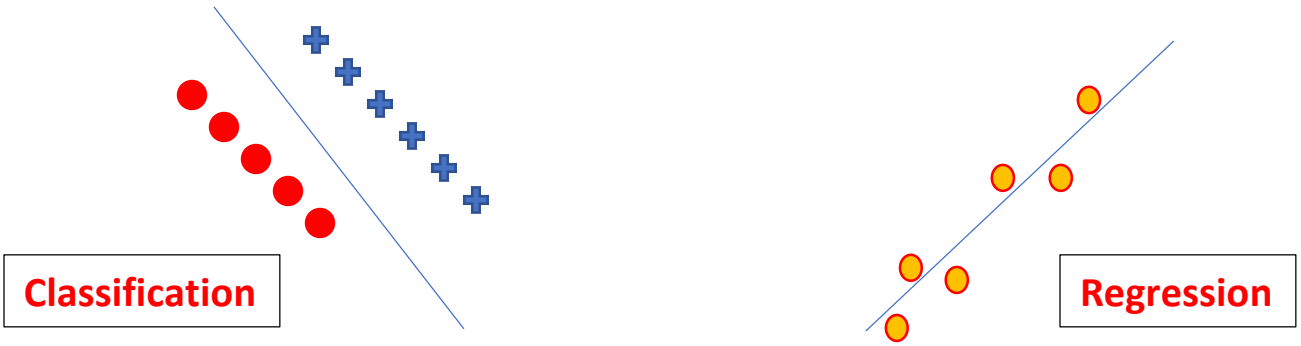
وهو التعليم الذي نشرف عليه observe و نوجه direct تنفيذ المهام execute of tasks ولكن هنا لن نوجه أو نشرف على عمل شخص ما ، بل على نموذج تعلم الآلة ليكون قادر على التعامل مع الحالات الغير متدرب عليها و الجديدة ، ويتم ذلك من خلال التدرب على مجموعة البيانات التي نملكها :

Teaching the model by data set → that knowledge it can predict unknown or future instances .

لو عدنا لمثال بيانات مواصفات خلايا المرضى لدينا data set حيث تكون ضمن جدول والنتيجة يعطينا إياها النموذج مبوبة labeled :



و يوجد نوعين للتعليم الإشرافي :



3-1- التصنيف Classification :

للتعامل مع البيانات المختلفة (رقمية أو نصية) حيث يتم تصنيف البيانات المتشابهة ضمن مجموعات كما رأينا في جدول خلايا المرضى حيث نصنفها سليمة أو خبيثة .

Classification is the process of predicting discrete class labels or categories.

3-2- التوقع Regression :

لإجراء التوقع ضمن القيم المتشابهة فيما بينها ليتوقع قيم جديدة مثل توقع غاز co_2 المنبعث من عوادم السيارات بناء على مواصفات محركاتها .

Regression is the process of predicting continues values.

4- التعلم غير الإشرافي unsupervised :

هنا ندع النموذج يكتشف المعلومات التي نريد توقعها دون توجيه أو إشراف حيث نعطيه البيانات فهو يتدرب على dataset ومن ثم يرسم النتيجة unlabeled وهناك عدة أنواع منها :

4-1- تقليل الأبعاد dimension reduction :

هنا يتم حذف بعض البيانات الزائدة أو الوافرة لتصغير حجمها وتسهيل التصنيف .

4-2- تحليل البيانات market basket analysis :

تعتمد فكرته على أنك تجلب مجموعة عناصر مع بعضها البعض بشكل مشابه نقوم بجلب مجموعة غيرها .

4-3- توقع وجود البيانات density estimation :

وفيها نوجد البيانات التي تتضمن بعض البنى المتشابهة .

4-4- التجميع clustering :

وهي أكثر الطرق المستخدمة حيث يتم تجميع البيانات بالاعتماد على عدة أمور متشابهة فيما بينها وأهم المجالات المستخدمة فيها :

discovering structure اكتشاف البيانات المتشابهة

summarization تجميع البيانات

anomaly detection اكتشاف الشذوذ

المقارنة بين التعلم الإشرافي و غير الإشرافي :

supervised	unsupervised
Regression, classification	clustering
more evaluation methods	fewer evaluation methods
controlled environment	less controlled environment

5- مقدمة في التوقع Regression :

لوعدنا لمثال غاز co_2 المنبعثة من السيارات :

ID	Engine	Cylinders	Fuel	Co_2
0	2.0	4	8.5	198
1	2.4	4	9.6	221
.	.	.	.	.
.	.	.	.	.
9	3	3	9.2	?

هذا الجدول يمثل data set ونريد توقع كمية الغاز المنبعثة من السيارة رقم 9 قبل إنتاجها بالاعتماد على بيانات السيارات الموجودة مسبقا ؟

سيتم هذا بإنشاء نموذج لتعلم الآلة يعتمد على التوقع regression وهنا يكون لدينا نوعين من المتغيرات :

وهي عمود القيم الخير وهنا هو مكية الغاز المنبعثة Y : Dependent variables

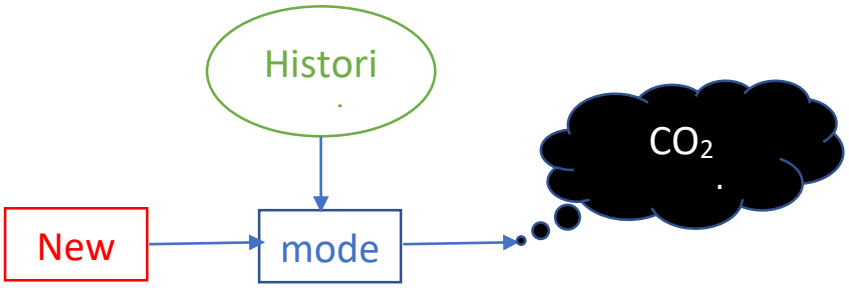
وهي مواصفات العناصر الموجودة لدينا في الجدول x : Independent variables
وهي متغيرات مستقلة لأن قيمها غير مرتبطة بقيم أخرى ندرسها

أما المتغيرات الغير مستقلة (final goal,target,state) في تتعلق بالمتغيرات المستقلة لأنه بناء على الأخير يتم توقع القيم التي نريد .

إذ يمكن أن نعتبر المتغيرات المستقلة على أنها مسببات causes لتلك الحالات states التي تظهر لدينا .

والمفتاح في التوقع هو أن البيانات مستمرة continuous وليست متقطعة discrete وعلى كل حال فإن المتغيرات x يجب ان تكون قيم مقاسة وقابلة للقياس سواء بشكلها النصي categorical أو الرقمي numerical أي يمكن تدريج قياسها بشكل مستمر continuous measurement scale .

إذا الذي نقوم به أننا نستخدم البيانات التي تخص السيارات لدينا historical data وجميع أو أغلب قيم صفاتها features of their attributes لنوجد نموذجا يتوقع لنا كمية co_2 المنبعثة من سيارة جديدة أو غير معروفة ويكون هذا النموذج غير مبرمج صراحة من قبل كما يبين المخطط التالي :



بشكل أساسي لدينا نوعان من هذا التوقع :

- توقع بسيط simple regression
- توقع متعدد multiple regression

عند استخدام متغير مستقل واحد x لتوقع قيمة Y عندها يكون لدينا توقع بسيط وله شكلان:

إما simple linear regression او simple non-linear regression .

وخطية linearity هذا التوقع تعتمد على طبيعة العلاقة التي تربط المتغيرات المستقلة والغير مستقلة فيما بينها .

وعند استخدام أكثر من متغير مستقل لتوقع قيمة Y عندها يكون التوقع من النوع multiple.

المهم يتم استخدام التوقع regression لتوقع قيم مستمرة مثل مسألة التنبؤ بالمبيعات sales forecasting ، حيث يمكن توقع مجموع المبيعات السنوية للبائع بالاعتماد على المتغيرات المستقلة X و التي تمثل (مثلا) : سنوات الخبرة – الشهادة – العمر - ....

كذلك يتم استخدام التوقع في التحليلات النفسية stratification analysis بالاعتماد على عدة عوامل نفسية وديموغرافية للفرد .

وأيضا يستخدم التوقع لتخمين أسعار price estimation المنازل في منطقة ما بالاعتماد على حجم المنزل وعدد غرفه و ....

وأيضا في تخمين دخل العامل Employment income بالاعتماد على ساعات العمل – الشاهدة – طبيعة العمل – الجنس – العمر – سنوات الخبرة ....

يوجد الكثير من المجالات التي نستخدم فيها التوقع مثل التحليلات المالية وتقسيط البيع و الراعية الصحية ... finance,health care, retail

هنالك الكثير من خوارزميات التوقع :

Ordinal regression

Poisson regression

Fast forest quintile regression

Linear , Polynomial , Lasso .Stepwise , Ridge regression

Bayesian linear regression

Neural network regression

Decision forest regression

Boosted decision tree regression

K-nearest neighbors (KNN) regression

6- التوقع الخطي البسيط simple linear regression :

لن تحتاج أي معرفة في الجبر الخطي لتتعلم التوقع الخطي البسيط و ما يلي سيعطيك معرفة شاملة لتستطيع التعامل معه .

كم قلنا سابقا أننا نستخدم متغير واحد من الخصائص لتوقع قيمة غير موجودة مسبقا ولنعود لمثال الغاز المنبعث من السيارات :

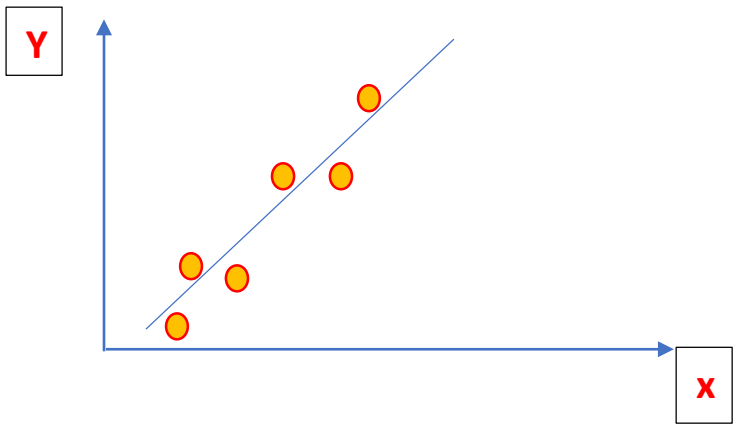
ID	Enginesize	Cylinders	Fuel	Co_2
0	2.0	4	8.5	198
1	2.4	4	9.6	221
.	.	.	.	.
.	.	.	.	.
9	3	3	9.2	?

سنستخدم فقط خاصية حجم المحرك لعمل نموذج توقع خطي بسيط يتوقع قيمة الغاز المنبعث حيث لدينا :

x يمثل المتغير المستقل وهو الحالة state وهو لدينا هنا Engine size

Y يمثل المتغير غير المستقل وهو الهدف target وهو لدينا الانبعاث Emission

لو قمنا برسم هذين المتغيرين بمخطط scatterplot لتمثيل هذه العلاقة Y(x) :



عندما تتغير القيمة في المتغير x تتغير قيمة Y وبالتالي يشير لوجود علاقة بينهما و نلاحظ من خلال المخطط السابق أنها خطية .

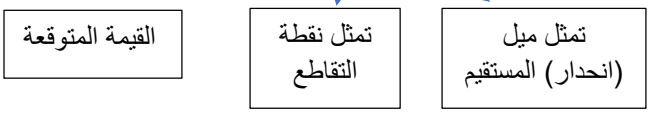
ورسم هذا المستقيم الذي يمثل هذه العلاقة حيث بازدياد حجم المحرك يزداد انبعاث الغاز co_2 وبالتالي يمكننا تخمين قيمة الانبعاث لأي سيارة مجهولة أو نريد تصنيعها ولدينا بقية خصائصها .

يتم تمثيل هذا الخط المستقيم بمعادلة كثير حدود polynomial له الشكل العام :

$$\hat{y} = \theta_0 + \theta_1 x_1$$

ويسمى ب خط التوقع أو الملاءَمة fit line حيث :

$$\hat{y} = \theta_0 + \theta_1 x_1$$

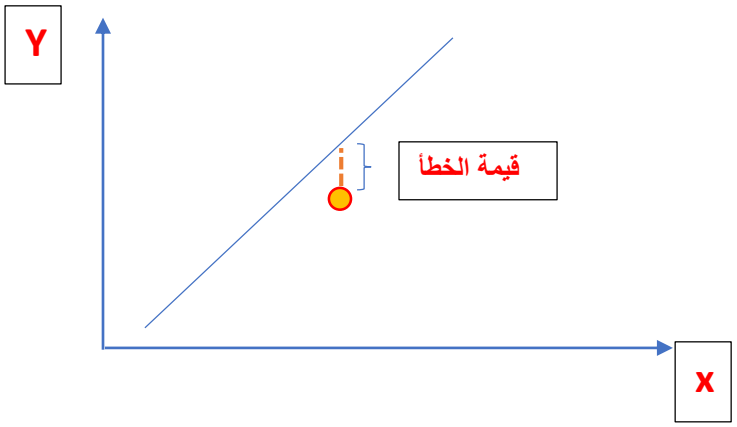


إن كل من الميل slope / gradient ونقطة التقاطع intercept يمثلان بارامترات المستقيم أو معاملاته coefficients حيث يتم ضبطهما للحصول على المعادلة الأنسب التي تمثل البيانات data set التي لدينا .

حيث نسمي القيمة المتوقعة بـ response variable والمتغير x_1 بـ single predictor

لو أردنا أن نجرب بعض القيم التي لدينا على هذه المعادلة من خلال الرسم البياني بأن نوجد الغاز المنبعث للمحرك ذو الحجم x_1=5.4 نجد أنه $\hat{y} = 340$ ولكن هو في الحقيقة من الجدول y=250 إذا هنالك خطأ قيمته الفرق بين القيمة الحقيقية والمتوقعة وقيمته 90 نسمي هذا الخطأ المتبقي residual error وهو بيانيا المسافة العمودية بين النقطة الحقيقية وبين النقطة المتوقعة على المستقيم :

The distance from the data point to the fitted regression line.



مع أخذ متوسط جميع قيم الخطأ بشكل بياني على المخطط نقرر فيما إذا كان الخط المستقيم مناسب أم لا لعملية التوقع لدينا ونعبر عنه رياضيا بمعادلة متوسط الخطأ mean square error :

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)}^{2}$$

The square of the difference between actual and predicted

حيث n عدد النقط لدينا .

الهدف إيجاد الخط المستقيم الذي يكون فيه الخطأ MSE أصغر ما يمكن وليتم ذلك ينبغي الحصول على أفضل قيم للمعاملات وهنا لدينا طريقتين لذلك :

- استخدام الطريقة الرياضية mathematic
- استخدام طريق التحسين optimization

يمكننا رياضيا حساب معاملات الخط المستقيم عن طريق إيجاد متوسط العمود x_1 وهو $\bar{x}$ ومتوسط العمود y وهو $\bar{y}$ عندها يكن لدينا :

$$\theta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\theta_0 = \bar{y} - \theta_0 \bar{x}$$

طبعا لسنا بحاجة لحفظ هذه المعادلات ففي لغات البرمجة python,R,scala يمكننا إيجادهم بسهولة .

الآن بعد العمل الرياضي نجد قيمة هذه المعاملات كالتالي :

$$\theta_1 = 39 \rightarrow \theta_0 = 125.74$$

معامل انبعاث الغاز CO_2 — معامل الانحياز bias

و الآن أصبح بإمكاننا كتابة معادلة المستقيم :

$$\hat{y} = 125.74 + 39\, x_1$$

وبالتالي لو أردنا حساب القيمة المتوقعة التي نريدها حيث x=2.4 تكون القيمة المتوقعة 218.6 .

كما رأينا سهولة التعامل في التوقع الخطي البسيط حيث تكمن في إيجاد معادلة المستقيم ولو أردنا كتابة محاسن pros التوقع الخطي البسيط نجد :

- سريعة و سهلة الفهم very fast and understand
- لا تحتاج لضبط معاملات معقدة مثل طرق k-nearest or NN
- سهلة التفسير high interpretable

في لغة البايثون يمكننا استخدام المكتبة scikit-learn .

نبدأ الآن بتطبيق الكود في البايثون باستخدام jupyter notebook :

Importing Needed packages

```
[4]: import matplotlib.pyplot as plt
     import pandas as pd
     import pylab as pl
     import numpy as np
     %matplotlib inline
```

Downloading Data

To download the data, we will use !wget to download it from IBM Object Storage.

```
In [2]: !wget -O FuelConsumption.csv https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/ML0101ENv3/labs/
/FuelConsumptionCo2.csv
```

Reading the data in

```
[19]: df = pd.read_csv("FuelConsumption.csv")

      # take a look at the dataset
      #df.head()
```

Data Exploration

Lets first have a descriptive exploration on our data.

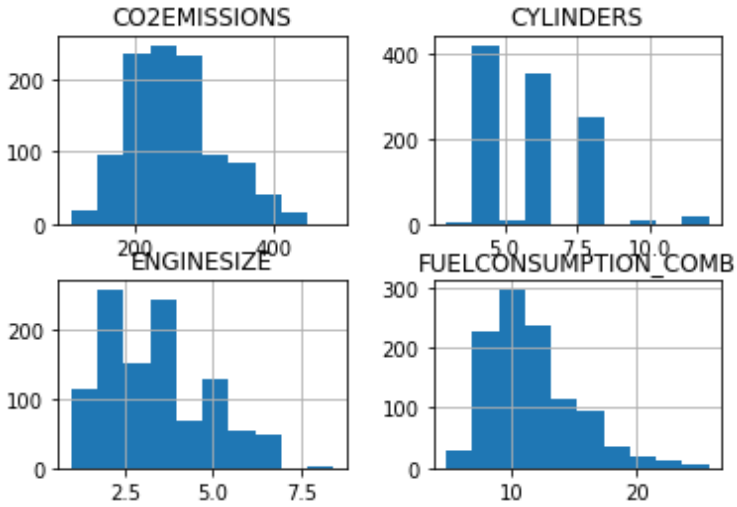
```
[7]: # summarize the data
     df.describe()
```

Lets select some features to explore more.

```
[8]: cdf = df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB','CO2EMISSIONS']]
     cdf.head(9)
```

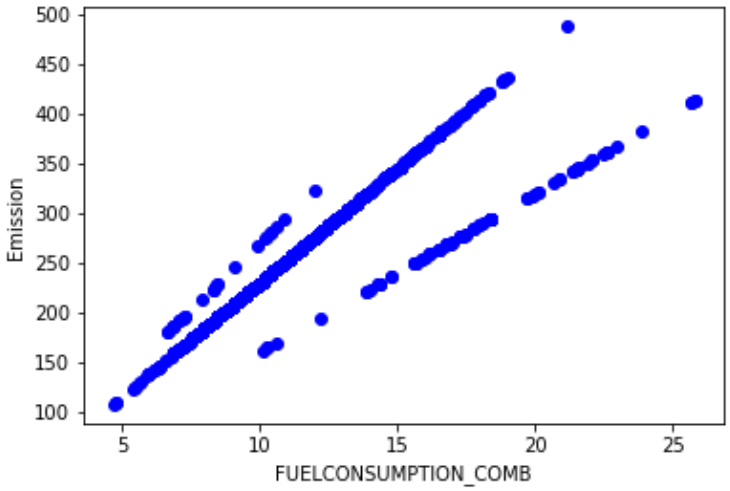
we can plot each of these fearues:

[9]:
```
viz = cdf[['CYLINDERS','ENGINESIZE','CO2EMISSIONS','FUELCONSUMPTION_COMB']]
viz.hist()
plt.show()
```

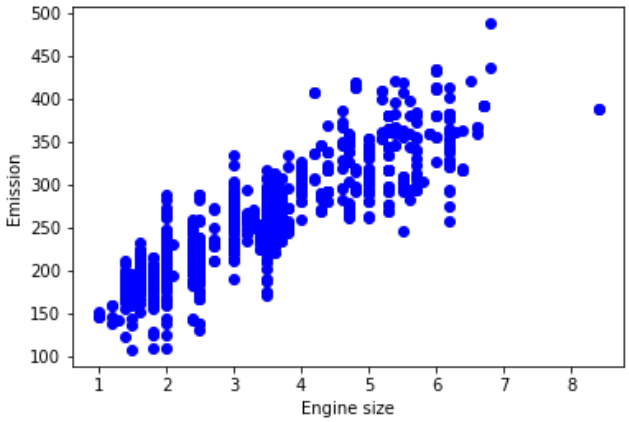


Now, lets plot each of these features vs the Emission, to see how linear is their relation:

[10]:
```
plt.scatter(cdf.FUELCONSUMPTION_COMB, cdf.CO2EMISSIONS,  color='blue')
plt.xlabel("FUELCONSUMPTION_COMB")
plt.ylabel("Emission")
plt.show()
```



[11]:
```
plt.scatter(cdf.ENGINESIZE, cdf.CO2EMISSIONS,  color='blue')
plt.xlabel("Engine size")
plt.ylabel("Emission")
plt.show()
```



Creating train and test dataset

Train/Test Split involves splitting the dataset into training and testing sets respectively, which are mutually exclusive. After which, you train with the training set and test with the testing set. This will provide a more accurate evaluation on out-of-sample accuracy because the testing dataset is not part of the dataset that have been used to train the data. It is more realistic for real world problems.

This means that we know the outcome of each data point in this dataset, making it great to test with! And since this data has not been used to train the model, the model has no knowledge of the outcome of these data points. So, in essence, it is truly an out-of-sample testing.

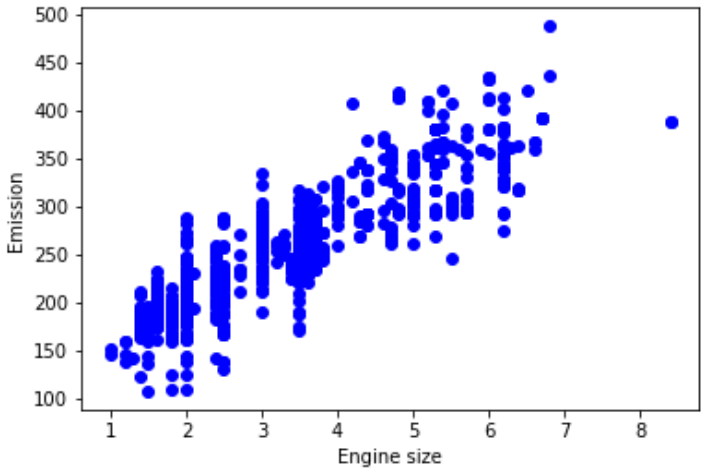
[12]:
```
msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]
```

Simple Regression Model

Linear Regression fits a linear model with coefficients B = (B1, ..., Bn) to minimize the 'residual sum of squares' between the independent x in the dataset, and the dependent y by the linear approximation.

Train data distribution

[13]:
```
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS,  color='blue')
plt.xlabel("Engine size")
plt.ylabel("Emission")
plt.show()
```



Modeling

Using sklearn package to model data.

[14]:
```
from sklearn import linear_model
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['ENGINESIZE']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

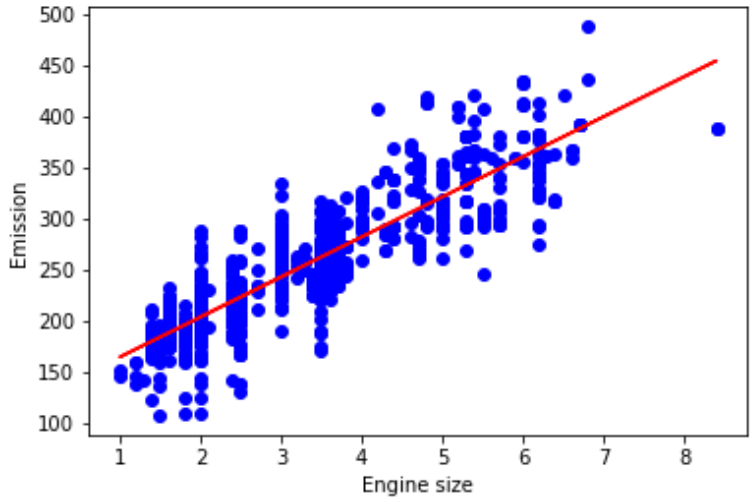
```
Coefficients:  [[39.1757159]]
Intercept:  [125.66702924]
```

Plot outputs

we can plot the fit line over the data:

[15]:
```
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS,  color='blue')
plt.plot(train_x, regr.coef_[0][0]*train_x + regr.intercept_[0], '-r')
plt.xlabel("Engine size")
plt.ylabel("Emission")
```

[15]:
```
Text(0, 0.5, 'Emission')
```



[21]:
```
from sklearn.metrics import r2_score

test_x = np.asanyarray(test[['ENGINESIZE']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
test_y_ = regr.predict(test_x)

print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 22.78
Residual sum of squares (MSE): 917.55
R2-score: 0.67
```

كما نقوم بتجربة الكود ضمن Spyder :

```
#استيراد المكتبات التي نحتاجها
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# قراءة ملف البيانات والاكتفاء بأربع خصائص
dataset = pd.read_csv('FuelConsumptionCo2.csv')
cdf = dataset[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB','CO2EMISSIONS']]
cdf.head(9)

# تقسيم البيانات لنأخذ المتغيرات المستقلة X والغير مستقلة y ولأنه بسيط أخذنا
# حجم المحرك والغاز المنبعث
X = cdf.iloc[:, 0].values
y = cdf.iloc[:, -1].values

# رسم كل من هذه الخصائص الأربعة التي اخترناها
viz = cdf[['CYLINDERS','ENGINESIZE','CO2EMISSIONS','FUELCONSUMPTION_COMB']]
viz.hist()
plt.show()

# رسم مخزن الوقود مع الغاز المنبعث لنرى مدى الخطية بينهما
plt.scatter(cdf.FUELCONSUMPTION_COMB, cdf.CO2EMISSIONS,  color='blue')
plt.xlabel("FUELCONSUMPTION_COMB")
plt.ylabel("Emission")
plt.show()

# رسم المحرك مع الغاز المنبعث لبيان مدى الخطية بينهما
plt.scatter(cdf.ENGINESIZE, cdf.CO2EMISSIONS,  color='blue')
plt.xlabel("Engine size")
plt.ylabel("Emission")
plt.show()
```

```
# رسم السلندر مع الغاز المنبعث لبيان مدى الخطية بينهما
plt.scatter(cdf.CYLINDERS, cdf.CO2EMISSIONS, color='blue')
plt.xlabel("Cylinders")
plt.ylabel("Emission")
plt.show()

# تحديد نسبة بيانات التدريب و الاختبار في البيانات
msk = np.random.rand(len(dataset)) < 0.8
train = cdf[msk]
test = cdf[~msk]

# رسم بيانات التدريب والاختبار لكل من المحرك و الغاز المنبعث
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS,  color='blue')
plt.xlabel("Engine size")
plt.ylabel("Emission")
plt.show()

# بناء النموذج بالاعتماد على بيانات التدريب فقط
from sklearn import linear_model
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['ENGINESIZE']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr.fit (train_x, train_y)

# The coefficients حساب معاملات الخط المستقيم
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)

# رسم بيانات التدريب والخط المستقيم
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS,  color='blue')
plt.plot(train_x, regr.coef_[0][0]*train_x + regr.intercept_[0], '-r')
plt.xlabel("Engine size")
plt.ylabel("Emission")

#اختبار النموذج باستخدام بيانات الاختبار
from sklearn.metrics import r2_score
test_x = np.asanyarray(test[['ENGINESIZE']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
test_y_ = regr.predict(test_x)

# حساب الخطأ للنموذج لبيان دقته
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

7- التوقع الخطي المتعدد multiple linear regression :

عند تواجد أكثر من متغير مستقل عندها يصبح التوقع الخطي من النوع المتعدد كمثالنا السابق بأن لا نعتمد فقط على حجم المحرك وإنما على عدة خصائص أخرى .

بشكل رئيسي يوجد نوعان من التطبيقات التي تستخدم هذا التوقع :

- **نستخدمه عندما نريد تبيان قوة التأثير the strength of effect التي تمتلكها المتغيرات المستقلة على المتغيرات الغير مستقلة أي :**

Independent variable effectiveness on prediction .

مثلا لدينا نموذج اختبار الطالب وعندنا المتغيرات المستقلة التالية :

وقت المراجعة revision time

قلق الامتحان test anxiety

حضور المحاضرات lecture attendance

الجنس gender

التوقع الخطي المتعدد هنا يبين مدى تأثر توقع مستوى الطالب بهذه المتغيرات المستقلة عندما نقر هذا السؤال لكل متغير :

Does …., ….. , …., and….. have any effect on the exam performance of student?

- **نستخدمه لتوقع تأثيرات التغييرات التي تحدث : predicting impacts of changes**

أي لفهم كيف تتغير المتغيرات الغير مستقلة عند تغير المتغيرات المستقلة أي كيف تتغير نتيجة التوقع بتغير المتغيرات المستقلة فمثلا عندما نراقب بيان شخص مريض فالتوقع الخطي المتعدد يمكنه إخبارنا كيف يتغير ضغط الدم (صعودا أو هبوطا up or down) عند كل زيادة أو نقصان في وزن المريض BMI للمريض Patient's Body mass index مع ثبات بقية العوامل holding other factors constant .

وكما هو في التوقع الخطي البسيط تكون العلاقة كثيرة حدود نستخدمها هنا وبالتالي تحديد أي من هذه المتغيرات تؤثر على نتيجة التوقع أي :

$$\text{Co2Em} = \theta_0 + \theta_1 . EngineSize + \theta_2 . Cylinders + \theta_3 . Fule + \cdots$$

وبالتالي القيمة المتوقعة جداء شعاعين كمصفوفات يكون لها الشكل التالي :

$$\hat{y} = \theta^T .* x$$

dot product between parameters vector and features set vector

ففي فضاء متعدد الأبعاد multi-dimension space تكون مصفوفة θ^T شعاع (n*1) للبارامترات المجهولة والشعاع x .

وتدعى هذه البارامترات أوزان (weight) شعاع معادلة التوقع لها الشكل :

$$\theta^T = [\theta_0, \theta_1, \theta_2, \theta_3, \dots \theta_n]$$

وشعاع المتغيرات المستقلة :

$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ . \\ . \\ x_n \end{bmatrix}$$

ففي فضاء أحادي البعد تصبح معادلة التوقع لخط مستقيم كما في التوقع الخطي البسيط ولكن في فضاء متعدد الأبعاد (أي أكثر من متغير x) يصبح مستوي ويدعى عندها plane or hyper-plane وهو المستخدم في التوقع الخطي المتعدد .

ويتم إيجاد هذ المستوي بإيجاد أفضل قيمة للمعاملات ثيتا لأجل توقع أفضل قيمة للهدف target في كل سطر من سطور dataset لدينا ، وكما رأينا سابقا أننا نحتاج لتحقيق ذلك أن نصغر الخطأ قدر الإمكان .

فأفضل البارامترات هي التي تعطي أقل قيمة خطأ ، إذاً كيف نحدد قيم هذه البارامترات ؟

رأينا في التوقع البسيط كيف نحسب الخطأ residual error الفرق بين القيمة المتوقعة والقيمة الحقيقية وكذلك معادلة الخطأ MSE .

أما في التوقع الخطي المتعدد يوجد طريقة أخرى حيث نقوم بتصغير MSE لجميع القيم المتوقعة و ذلك بتخمين أفضل قيم ثيتا مناسبة وأكثر الطرق شيوعا في تخمين هذه القيم :

- The ordinary least squares : حيث تعتمد على إيجاد أصغر قيمة MSE لكل سطر من سطور البيانات dataset لدينا ولكن هذه الطريقة معقدة حسابيا و تستهلك وقت كبير خصوصا عندما يكون لدينا في مصفوفة البيانات أكثر من 10k سطر ، لذلك يمكننا التفكير باستخدامها عند أقل من هذا العدد .
- Optimization approach : تستخدم تكرار عملية تصغير الخطأ للنموذج على البيانات التي نتدرب عليها

Minimizing the error of the model on your training data .

يمكننا مثلا استخدام خوارزمية Gradient Descent GD التي تبدأ بقيم عشوائية لـ ثيتا ثم نعيد تكرار حسابها وفي كل مرة نحسب الخطأ و نحاول الوصول لأقل قيمة له .

وهذه الطريقة مناسبة عندما تكون البيانات ضخمة أي أكثر من 10k سطر .

بعد الحصول على أفضل قيم ثيتا نعوضها في معادلة التوقع ونحسبها فمثلا حصلنا على القيم :

$$\hat{y} = 125 + 6.2\,x_1 + 14\,x_2 + \cdots$$

Engine Size Cylinder

فبمقارنة قيم ثيتا لكلا المتغيرين نجد قيمتها لـ Cylinder أكبر منها لحجم المحرك إذا تأثير cylinder على قيمة غاز co_2 المنبعث المتوقعة أكبر من تأثير حجم المحرك .

والآن لدينا بعض الأسئلة :

1- متى نستخدم التوقع الخطي البسيط ومتى نستخدم المتعدد ؟

كما رأينا سابقا فعند استخدام أكثر من متغير مستقل لتوقع قيمة الهدف نستخدم التوقع المتعدد وعندما يكون لدينا متغير مستقل واحد نستخدم البسيط .

2- كم عدد المتغيرات المستقلة التي يمكننا استخدامها في التوقع المتعدد ؟

هل نستعمل جميع المتغيرات ؟ وهل نضيف متغيرات أخرى لزيادة دقة التوقع ؟
بشكل عام إن زيادة عدد المتغيرات المستقلة دون دراسة نظرية محكمة يمكن أن يسبب نموذج توقع غير ملائم over-fit model وهذا بحد ذاته مشكلة حقيقية لأنه سيسبب تعقيدا للبيانات والحسابات وفي نفس الوقت ليس كافيا للتوقع .
لذلك من الأفضل تجنب استخدام العديد من المتغيرات ومع ذلك يوجد طرق لتجنب حدوث مشكلة overfitting .

3- هل ينبغي أن تكون المتغيرات المستقلة مستمرة ؟

بشكل رئيسي أو عام يمكن استخدام المتغيرات النصية categorical بعد تحويلها للشكل الرقمي مثلا :
لدينا سيارة نوعها أوتوماتيك أو عادي عندها يمكننا أن نفرض إن كانت automatic أن تأخذ القيمة 1 وإن كانت manual أن تأخذ القيمة 0 .

4- ما هي العلاقة بين المتغيرات المستقلة و الغير مستقلة ؟

التوقع المتعدد هو حالة خاصة من التوقع الخطي لذلك هذه العلاقة بين المتغيرات المستقلة و الغير مستقلة تكون خطية وهنالك عدة طرق لاختبار خطية هذه العلاقة مثل استخدام الرسم البياني scatter plots لذلك عند رسمها وتكون غير خطية بيانيا ننتقل عندها للتوقع الغير خطي .

8- تقييم عمل نموذج التوقع Model Evaluation in Regression Models :

الهدف من التوقع هو بناء نموذج ليتوقع بدقة الحالات الجديدة أو المجهولة لذلك في النهاية علينا أن نجري تقييم evaluation لهذا النموذج وسيتم ذلك من خلال طريقتين:

- التدريب و الاختبار على نفس البيانات التي لدينا train and test on the same dataset .
- التدريب على جزء من البيانات لدينا والاختبار على جزء آخر train/test split

مع تحديد محاسن pros ومساوئ cons كل طريقة وسندرس مقاييس metric الدقة accuracy لنماذج التوقع .

1-8- التدريب و الاختبار على نفس البيانات لدينا :

عندما نختار طريقة ما لتقييم نموذج يأتي السؤال : كيف نقيس دقة النموذج الذي وصلنا إليه لنختبره هل هو جيد أم لا ؟ وكيف نثق بما وصلنا إليه ؟ أحد الحلول هو أن نعطي النموذج جزء portion من البيانات التي لدينا أصلا ونجربها عليه لنرى هل سيتوقع نفس النتيجة الحقيقية لدينا أم لا !!!

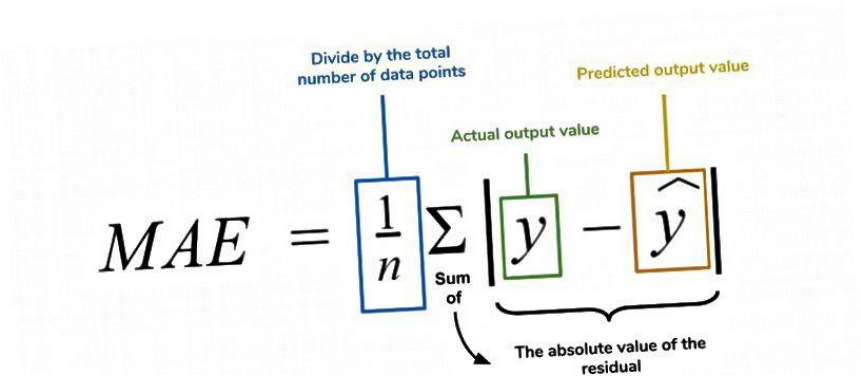
في مثالنا لتوقع انبعاث غاز co_2 من عوادم السيارات استخدمنا حينها جميع البيانات التي لدينا (أي جميع الصفوف من 0 إلى 9) أي قمنا بالتدريب عليها جميعها ولنقم بالاختبار مثلا على الصفوف من 6 إلى 9 بحيث نستخدمها للتحقق من دقة النموذج .

إذا أصبح لدينا مجموعة تدريب train من 0 إلى 9 ومجموعة اختبار test من 6 إلى 9 لها قيم حقيقية ولكن لا نستخدمها في التوقع وإنما فقط للتحقق على الواقع و هي القيم في عمود $EmCo_2$ وتدعى Actual values .

الآن نمرر الـ features (مجموعة الاختبار التي اخترناها من 6 إلى 9) إلى النموذج الذي دربناه ونرى ماذا يتوقع لنا ، ثم نجري مقارنة بين ما يتوقعه لنا النموذج وبين القيم الحقيقة التي لدينا أصلا ، وهذا مؤشر لدقة النموذج من عدمها .

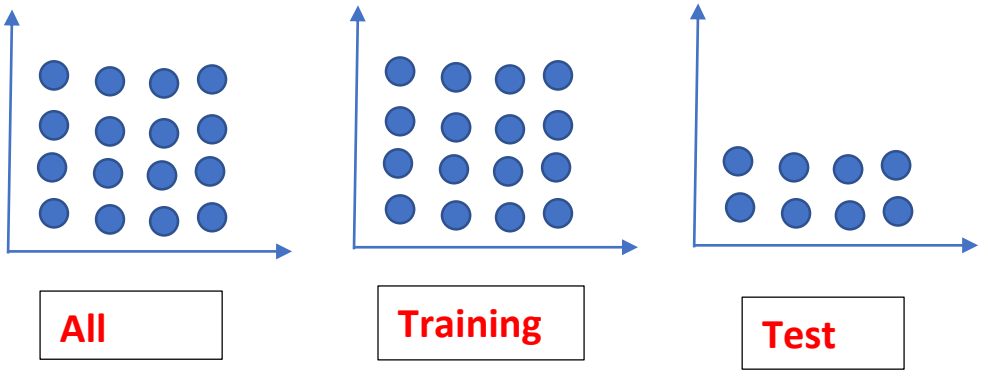
يوجد العديد من المقاييس التي نعتمد عليها لتحديد دقة النموذج ولكن أغلبها يعتمد على التقارب بين القيم المتوقعة و القيم الحقيقية .

أحد هذه المقاييس المعادلة التالية التي تحسب متوسط الخطأ بين القيم الحقيقية y والمتوقعة $\hat{y}$.



إذا خلال هذه الطريقة في التقييم نجري التدريب و الاختبار على نفس مجموعة البيانات وتتميز بالتالي :

- دقة تدريب عالية high training accuracy
- و دقة منخفضة خارج العينات low out-of-sample accuracy لأن النموذج أصلا يعلم بياناته التي استخدمت في التدريب .



وللفصل بين هاتين النقطتين :

أولاً حالة الدقة العالية :

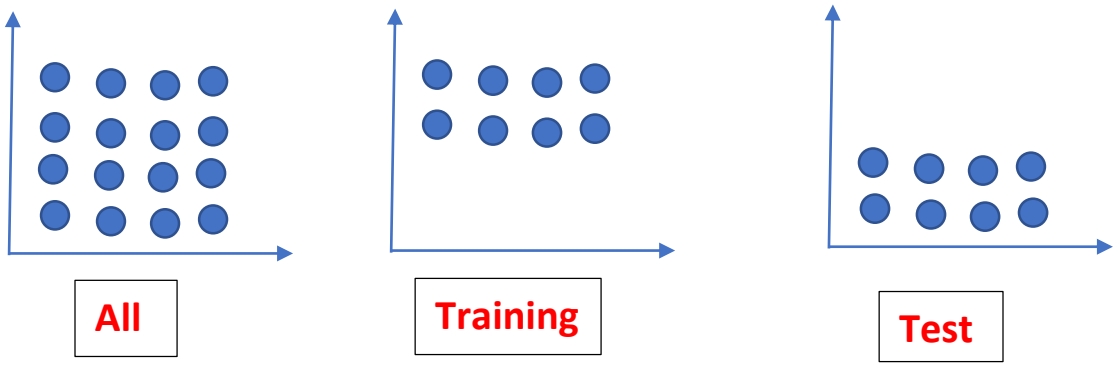
النسبة المئوية للتوقعات الصحيحة التي يوجدها النموذج عند استعمال مجموعة الاختبار وعلى كل حال فإن الدقة العالية ليست بالضرورة أن تكون شيء جيد دوما ، فمثلا ممكن وجود هكذا أمر أن يسبب حالة over-fit أي يصبح غير ملائم للبيانات ليتعدى النموذج مرحلة التوقع و يدخل في حالة ضجيج وتشويه للبيانات المتوقعة .

ثانياً حالة الدقة منخفضة :

وهذا بسبب أنه يتم الاختبار على بيانات من خارج مجموعة dataset لدينا لذلك نحتاج لتحسين هذه النسبة المئوية بالانتقال للطريقة الثانية لتقييم النموذج .

2-8- تقسيم البيانات train/test split :

هنا يتم استخدام جزء من البيانات للتدريب مثلا من 0 إلى 5 و البقية من 6 إلى 9 للاختبار ونتابع بنفس ترتيب التسلسل في الطريقة السابقة من حساب الخطأ :



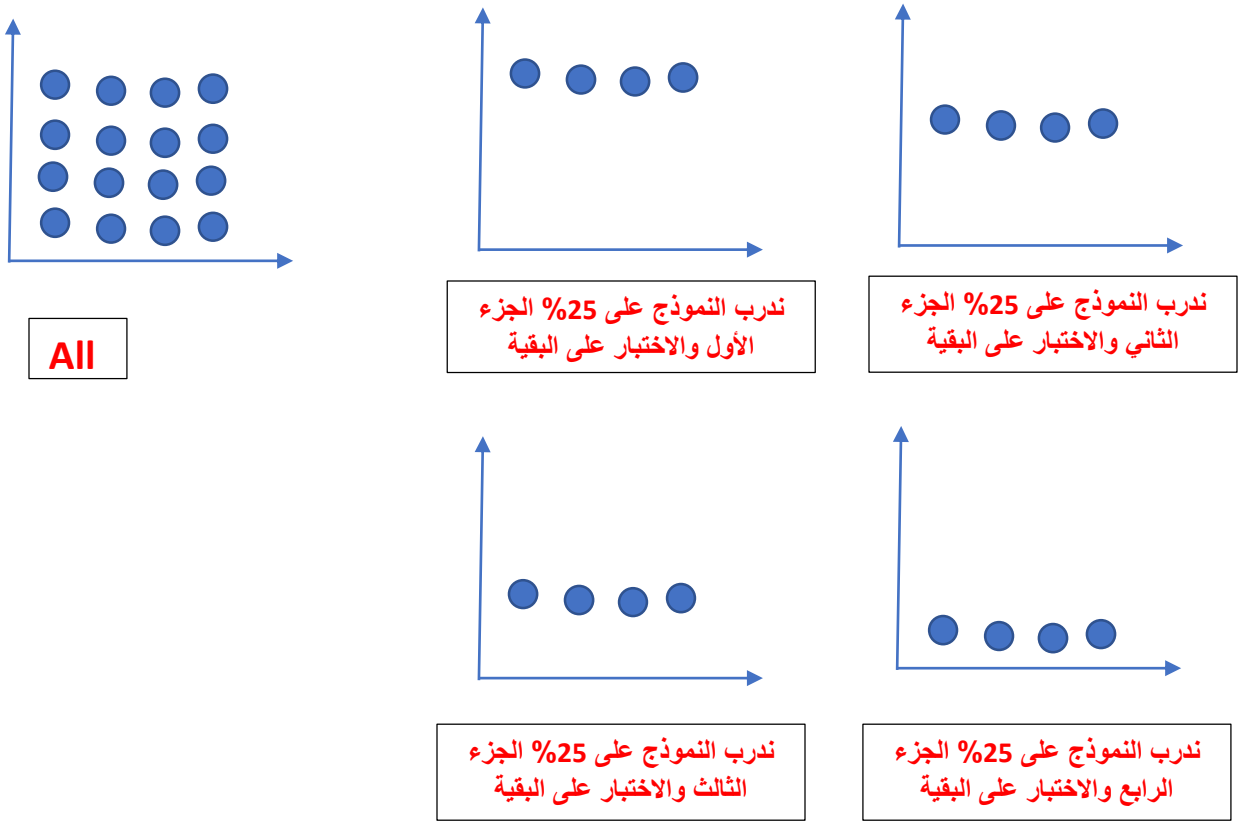
وهنا تم تقسيم الداتا لجزأين منفصلين عن بعض أحدهما للتدريب وآخر للاختبار mutually exclusive .

بهذه الطريقة ندرب النموذج على جزء ونختبر على جزء آخر مما يزيد النسبة المئوية للدقة في حالة out-of-sample .

ومن ثم نقوم بتدريب النموذج على مجموعة الاختبار نفسها بهدف أن لا نخسر أي من البيانات لدينا فمن المحتمل أن تكون ذات قيمة potentially valuable.

مع أن هذه الطريقة أفضل من سابقتها ولكن تكمن مشكلة تقسيم البيانات أيها للتدريب و أيها للاختبار .

لذلك هنالك طريقة لذلك تدعى k-fold cross-validation فمثلا k=4 فنقسم البيانات لأربع أجزاء بحيث لا تدخل بيانات جزء بالجزء الآخر كما يلي :



نتيجة الدقة أول مرحلة 80% والثانية 84% والثالثة 82% والرابعة 86% ثم نأخذ المتوسط الحسابي للكل فنحصل على الدقة الكلية للنموذج :

$$AVG(80+82+84+8) = 83\% = Accuracy$$

ننتبه أن كل جزء fold متباعد عن الآخر أي لا يوجد بيانات تدريب في قسم تكون موجودة في قسم آخر للتدريب و بالتالي حصلنا على نسبة ثابتة من النسبة المئوية للدقة في حالة out-of-sample .

9- مقاييس التقييم evaluation metrics :

سنتعرف على مقاييس الدقة accuracy metrics لتقييم النموذج فهي تشرح وتبين لنا أداء النموذج ، ولنتحدث عن المقاييس التي تقيم النموذج في حالة التوقع regression :

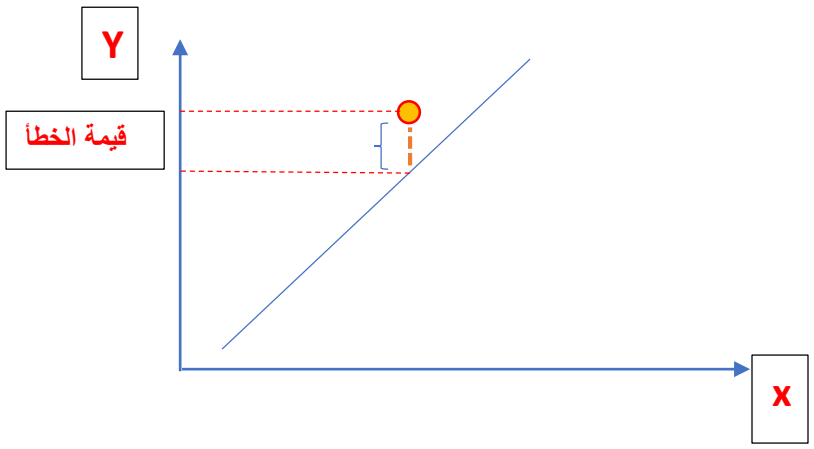
في مثالنا السابق بديهياً سنقارن بين القيم الحقيقية لمجموعة الاختبار والقيم التي توقعها النموذج لدينا .

فمقياس الدقة الذي يحدد لنا أي المقاطع أو الأقسام التي تحتاج للتطوير .

من هذه الطرق :

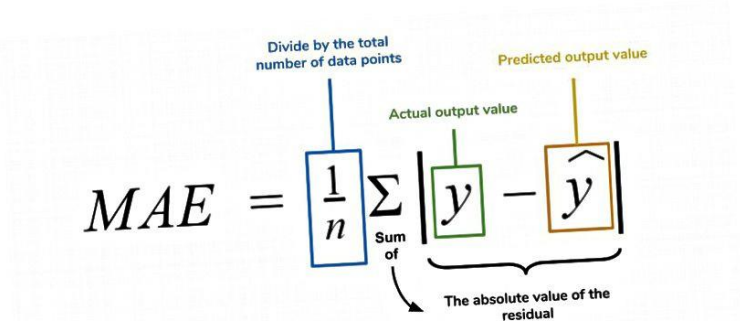
- الخطأ النسبي المطلق Relative Absolute Error RAE
- متوسط الخطأ المطلق Mean Absolute Error MAE
- متوسط مربع الخطأ Mean Squared Error MSE
- جذر متوسط مربع الخطأ Root Mean Squared Error RMSE

وهنالك طرق أخرى ، ولكن نرى أن كل هذه الطرق متعلقة بقيمة الخطأ ، فما هو الخطأ لدينا في التوقع : هو الفرق بين القيمة الحقيقية و القيمة المتوقعة وهي المسافة الشاقولية المبينة في الشكل :



ولأنه يكون لدينا عدة نقاط و بالتالي لدينا عدة قيم للخطأ يتم حسابها بإحدى الطرق التي ذكرناها :

9-1- متوسط الخطأ المطلق MAE وهي أبسط الطرق :



$$MAE = \frac{1}{n}\Sigma\left|y - \hat{y}\right|$$

9-2- متوسط مربع الخطأ MSE وهي أشهر من الطريقة الأولى فبسبب وجود التربيع فالمعادلة موجهة geared إلى أكبر خطأ موجود ويسبب ارتفاع الخطأ بشكل أسي exponentially لدينا مقارنة بالخطأ الصغير :

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)}^{2}$$

The square of the difference between actual and predicted

9-3- جذر متوسط مربعات الخطأ RMSE وميزتها أن هذه القيمة يتم تأويلها interpretable لأنها تعود لنفس وحدة الشعاع y مما يسهل التعامل معه إلى نفس درجة كثير الحدود لدينا :

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

4-9- الخطأ المطلق النسبي RAE ويسمى أيضاً Residual Sum of Error حيث يتم استعمال متوسط القيم الحقيقية $\bar{y}$:

$$RAE = \frac{\sum_{j=1}^{n}|y_j - \hat{y}_j|}{\sum_{j=1}^{n}|y_j - \bar{y}|}$$

5-9- الخطأ التربيعي النسبي RSE

$$RSE = \frac{\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}{\sum_{j=1}^{n}(y_j - \bar{y})^2}$$

وهو كثير الاستخدام في مجال علم البيانات Data Science لإيجاد ما يسمى R-squared وهو ليس الخطأ بحد ذاته per se وإنما مقياس لدقة النموذج :

$$R - squared = 1 - RSE \equiv R^2 = 1 - RSE$$

حيث يعبر عن مدى قرب القيم الحقيقية من الخط المستقيم الممثل للقيم المتوقعة فكلما زادت قيمة R^2 كلما كان النموذج أفضل .

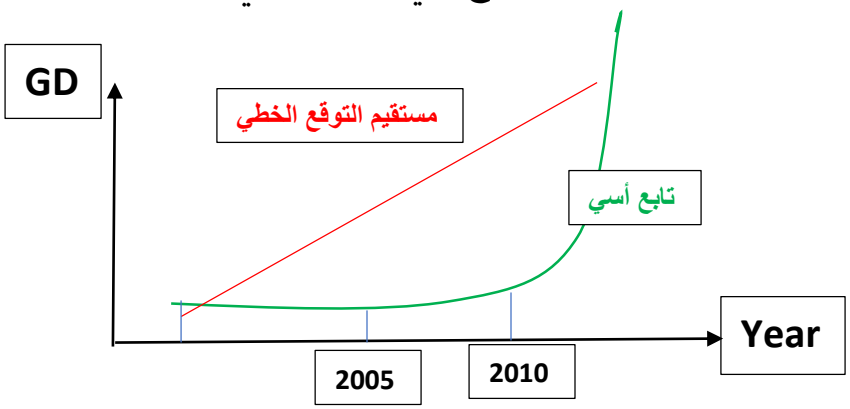
كل هذه الطرق السابقة تقيس قدرة نموذجك على التوقع وتحديد أي منها نستخدم متوقف على نوع النموذج و نوع البيانات و مجال المعرفة لديك .

10-التوقع الغير خطي non-linear regression :

لو كان لدينا مثلا بيانات الإجمالي المحلي للصين China's Gross Domestic GDP من عام 1960 إلى 2014 والدخل المحلي السنوي annual gross domestic بعملة الدولار US أي الدخل السنوي income :

ID	Year	Value
0	1960	5.9 e+10
1	1961	4.9 e+10
.	.	.
.	.	.
9	1969	7.8 e+10
.	.	.
.	.	.

وعن رسم هذه البيانات يكون لها شكل تابع أسي وليس خطي :



هنا لدينا سؤالين :

1- هل يمكن توقع GDP بوقت ما ؟
2- هل نستطيع بناء نموذج توقع خطي بسيط لهذه البيانات ؟

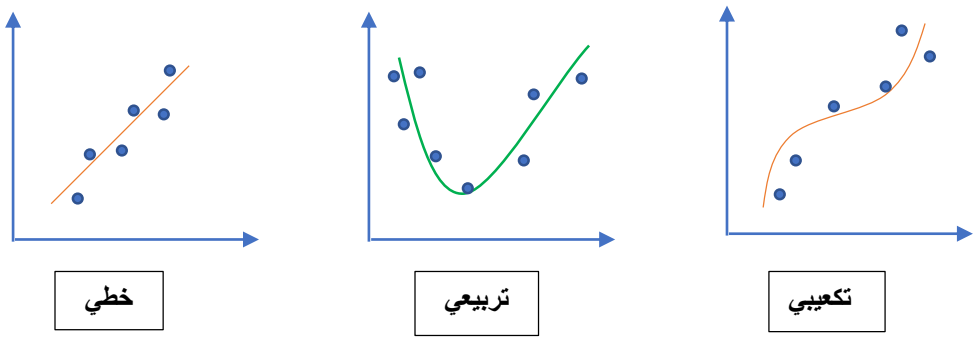
حقيقة عندما نرى أن البيانات تشكل منحني معوج curvy trend فلن يعطي التوقع الخطي نتائج دقيقة ، لأنه يمثَّل بمستقيم وهنا GDP علاقتها قوية مع الزمن وليست خطية .

حيث نرى تغير بسيط لقيم GDP حتى عام 2005 ثم تصعد بعدها بسرعة ثم يتغير ببطء بعد علم 2010 .، فهو يشبه تابع أسي logistical or exponential function لذلك نحتاج توقع غير خطي يشبه بمعادلته الشكل التالي :

$$\hat{y} = \theta_0 + \theta_1\theta_2^x$$

وتكون مهمتنا إيجاد البارامترات ثيتا .

يمكننا عمل نموذج التوقع بعدة أنماط خطي linear أو تربيعي quadratic(parabolic) مكعبي cubic .



وتتزايد الدرجة بحسب البيانات التي لدينا .

المقصود (المضمون) in essence أننا ندعوها جميعها كثير حدود التوقع polynomial regression حيث تحدد العلاقة بين المتغيرات المستقلة x والغير مستقلة y درجة كثير الحدود بحسب درجة x .

ويجب الانتباه لانتقاء pick التوقع المناسب للبيانات التي لديك ، إذا ما هو كثير الحدود الذي يناسب جميع أنواع البيانات ؟

هذا متوقف على نوع بياناتك فمثلا كثير الحدود من الدرجة الثالثة :

$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

وعلينا تخمين قيم ثيتا ليكون النموذج مناسب للبيانات الأساسية underlying data وبالرغم من أن العلاقة غير خطية فيمكننا تحويلها لعلاقة خطية !

لو عدنا لكثير الحدود السابق من الدرجة الثالثة وفرضنا التالي :

$$x_1 = x, x_2 = x^2, x_3 = x^3$$

أصبحت المعادلة كالتالي :

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

وهي كثير حدود خطي يمكننا اعتباره حالة خاصة من التوقع المتعدد التقليدي traditional multiple regression ونعود عندها لاستخدام نفس الآلية التي استخدمناها سابقا وهي Least squares وهي لتخمين البارامترات المجهولة في التوقع الخطي حيث نسعى لتصغير مجموع مربعات الفارق بين البيانات الحقيقية والتي توقعناها عن طريق التابع الخطي السابق Minimizing the sum of the squares of the difference between $\hat{y}, y$: .

إذا بعد كل هذا ما هو التوقع الغير خطي بالضبط ؟

هو يمثل نموذج غير خطي بين المتغيرات المستقلة والغير مستقلة وينبغي أن تكون $\hat{y}$ تابع غير خطي للبارامترات θ وليس بالضرورة للمتغيرات x (features) .

ويكون شكلها أسي ، لوغاريتمي ، عادي ، أو عدة أشكال أخرى وفي كل الحالات تغير $\hat{y}$

مرتبط بتغير θ وليس بالضرورة فقط على تغير x أي التوقع الغير خطي هو نموذج غير خطي للبارامترات .

$$\hat{y} = \theta_0 + \theta_1^2 x$$

$$\hat{y} = \theta_0 + \theta_1 \theta_2^x$$

$$\hat{y} = \log(\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3)$$

$$\hat{y} = \frac{\theta_0}{1 + \theta_1^{(x-\theta_2)}}$$

بالمقابل in contrast في التوقع الخطي لا يمكننا استخدام least squares لملاءَمة البيانات في التوقع الغير خطي وعموماً تخمين البارامترات ليس سهلا دوما .

- إذا كيف نعلم أن المسألة خطية أو غير خطية بطريقة سهلة ؟
  - رسم البيانات التي لدينا والأفضل رسم ثنائي الأبعاد لمتغيرات الخرج بالنسبة للدخل فالمنحني يميز العلاقة خطية أو غير خطية .
  - يمكننا حساب معامل الارتباط correlation coefficient بين المتغيرات المستقلة والغير مستقلة .
  - إذا كانت قيمة معامل الارتباط لكل التغيرات 0.7 فما فوق فالعلاقة أقرب للخطية بين الدخل و الخرج .
  - أو نقوم باستخدام التوقع الخطي فإن كانت النتائج غير مرضية ننتقل للتوقع الغير خطي .
- وكيف نبني نموذج بياناتي إذا كانت الغير خطية موجودة ؟ هنا نستخدم إحدى الطرق التالية :
  - Polynomial regression
  - Non-linear regression model
  - Transform your data

نتيجة : نستخدم التوقع المتعدد عندما نريد بيان قوة تأثير المتغيرات المستقلة على المتغيرات الغير مستقلة .

سنطبق مثال GPD للصين من عام 1964 لعام 2010 :

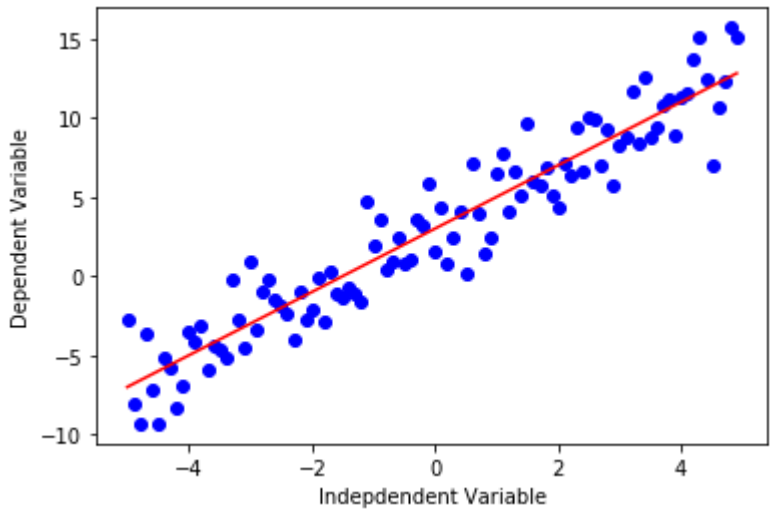
- الخطوة الأولى استيراد المكتبات اللازمة :

Importing required libraries

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

- سنجري رسم للتوقع الخطي لمعادلة بسيطة y=2x+3 من باب التذكرة :

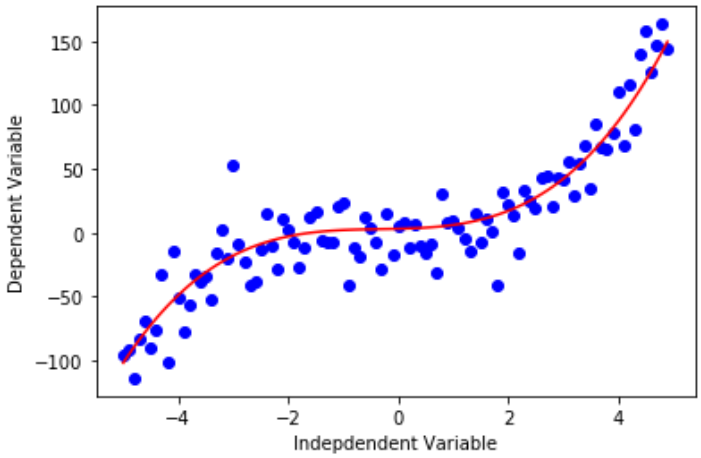
[2]:
```
x = np.arange(-5.0, 5.0, 0.1)

##You can adjust the slope and intercept to verify the changes in the graph
y = 2*(x) + 3
y_noise = 2 * np.random.normal(size=x.size)
ydata = y + y_noise
#plt.figure(figsize=(8,6))
plt.plot(x, ydata,  'bo')
plt.plot(x,y, 'r')
plt.ylabel('Dependent Variable')
plt.xlabel('Indepdendent Variable')
plt.show()
```



- أما التوقع الغير خطي فيكون مثلا كثير حدود درجة ثالثة :

$$y = ax^3 + bx^2 + cx + d$$

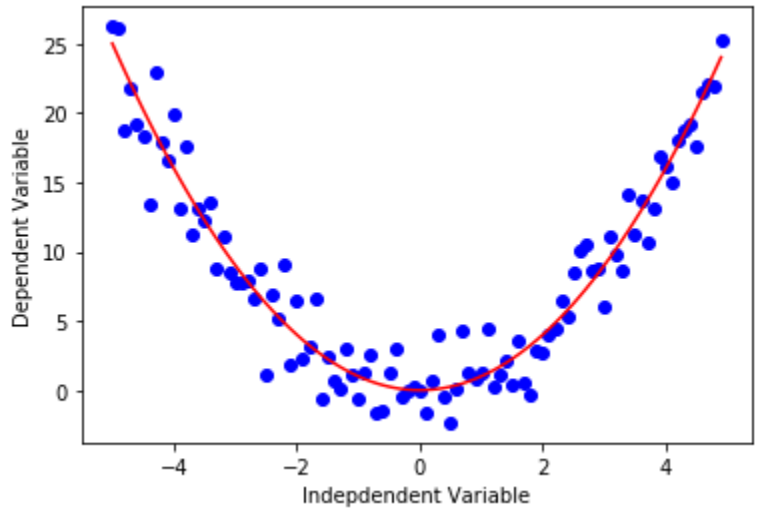
[3]:
```
x = np.arange(-5.0, 5.0, 0.1)

##You can adjust the slope and intercept to verify the changes in the graph
y = 1*(x**3) + 1*(x**2) + 1*x + 3
y_noise = 20 * np.random.normal(size=x.size)
ydata = y + y_noise
plt.plot(x, ydata,  'bo')
plt.plot(x,y, 'r')
plt.ylabel('Dependent Variable')
plt.xlabel('Indepdendent Variable')
plt.show()
```



- والتربيعي له الشكل :

$$Y = X^2$$

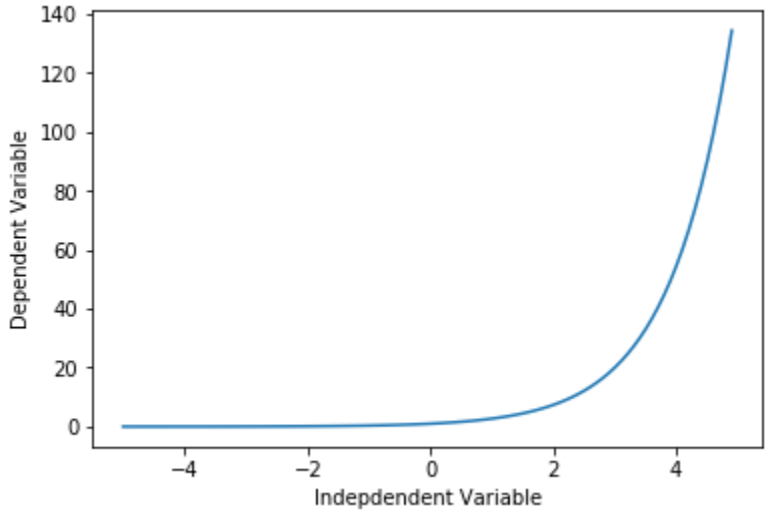
[4]:
```
x = np.arange(-5.0, 5.0, 0.1)

##You can adjust the slope and intercept to verify the changes in the graph

y = np.power(x,2)
y_noise = 2 * np.random.normal(size=x.size)
ydata = y + y_noise
plt.plot(x, ydata,  'bo')
plt.plot(x,y, 'r')
plt.ylabel('Dependent Variable')
plt.xlabel('Indepdendent Variable')
plt.show()
```



- والأسي له الشكل :

$$Y = a + bc^X$$

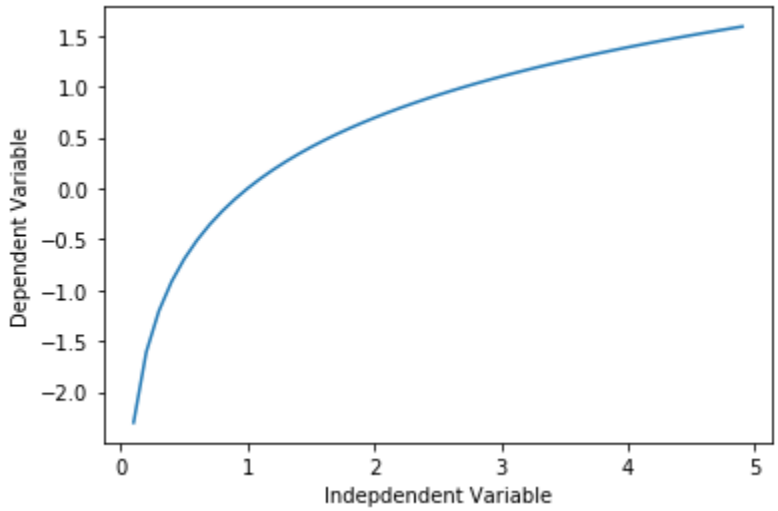
[5]:
```
X = np.arange(-5.0, 5.0, 0.1)

##You can adjust the slope and intercept to verify the changes in the graph

Y= np.exp(X)

plt.plot(X,Y)
plt.ylabel('Dependent Variable')
plt.xlabel('Indepdendent Variable')
plt.show()
```



- اللوغاريتمي :

$$y = \log(x)$$

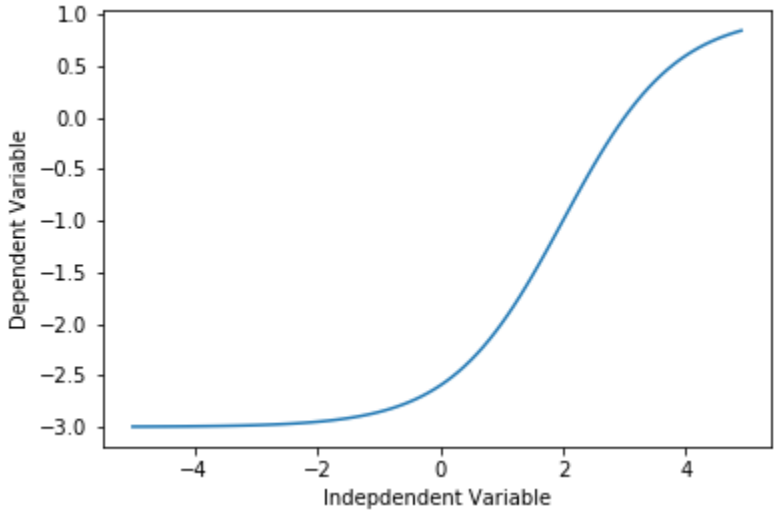
[6]:
```
X = np.arange(-5.0, 5.0, 0.1)

Y = np.log(X)

plt.plot(X,Y)
plt.ylabel('Dependent Variable')
plt.xlabel('Indepdendent Variable')
plt.show()
```



- السجمويد Sigmoidal/Logistic :

$$Y = a + \frac{b}{1 + c^{(X-d)}}$$

[7]:
```
X = np.arange(-5.0, 5.0, 0.1)


Y = 1-4/(1+np.power(3, X-2))

plt.plot(X,Y)
plt.ylabel('Dependent Variable')
plt.xlabel('Indepdendent Variable')
plt.show()
```



بهذا نكون اضطلعنا على الفرق بين التوقع الخطي والغير خطي بأهم أشكاله .

نأتي الآن للمثال الذي سندرسه وهو GPD للصين :

- استيراد المكتبات اللازمة وتحميل البيانات من الرابط والاكتفاء بأول عشر أسطر منها:

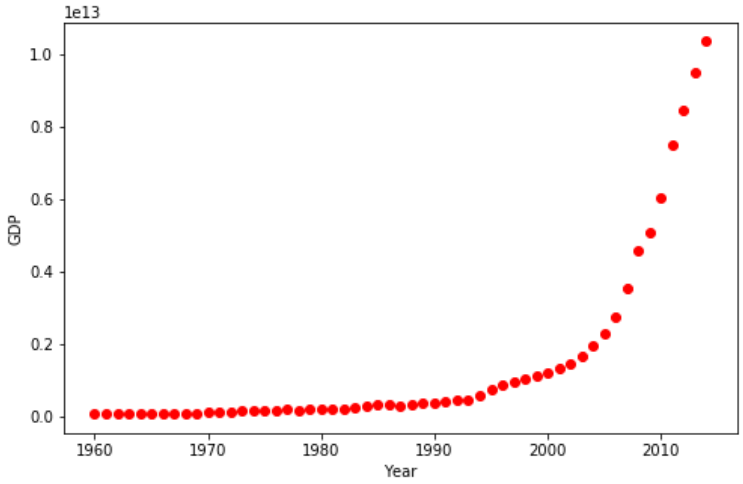
[8]:
```
import numpy as np
import pandas as pd

#downloading dataset
!wget -nv -O china_gdp.csv https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/ML0101ENv3/labs/china_gdp.csv

df = pd.read_csv("china_gdp.csv")
df.head(10)
```

[8]:

	Year	Value
0	1960	5.918412e+10
1	1961	4.955705e+10
2	1962	4.668518e+10
3	1963	5.009730e+10
4	1964	5.906225e+10
5	1965	6.970915e+10
6	1966	7.587943e+10
7	1967	7.205703e+10
8	1968	6.999350e+10
9	1969	7.871882e+10

- الآن لتحديد العلاقة بين المدخلات والمخرجات هل خطية أم غير خطية نرسمها :

```
[9]: plt.figure(figsize=(8,5))
     x_data, y_data = (df["Year"].values, df["Value"].values)
     plt.plot(x_data, y_data, 'ro')
     plt.ylabel('GDP')
     plt.xlabel('Year')
     plt.show()
```



نجد من الرسم البياني أن العلاقة خطية وبالتالي سنستخدم التوقع غير الخطي ولكن أي أنواعه سنستخدم : كثير الحدود – الأسي – اللوغاريتمي – السجمويد .
نلاحظ من الرسم البياني أنها تزداد ببطء في البداية ثم تتسارع في الوسط ثم تعاود للانخفاض (قد يتبادر للذهن الأسي ولكن هذا مخالف حيث الأسي تبدأ ببطء ثم تتزايد دون انخفاض) لذلك سيكون لدالة السجمويد النصيب الأوفر لهذا التوقع ولو أحببنا أن نجرب الأسي ونرى أيهما أفضل فلا بأس بذلك و طبعا على حساب الجهد والوقت :

الشكل العام لدالة logistic :

$$\hat{Y} = \frac{1}{1 + e^{\beta_1(X-\beta_2)}}$$

β_1 : Controls the curve's steepness,

β_2 : Slides the curve on the x-axis.

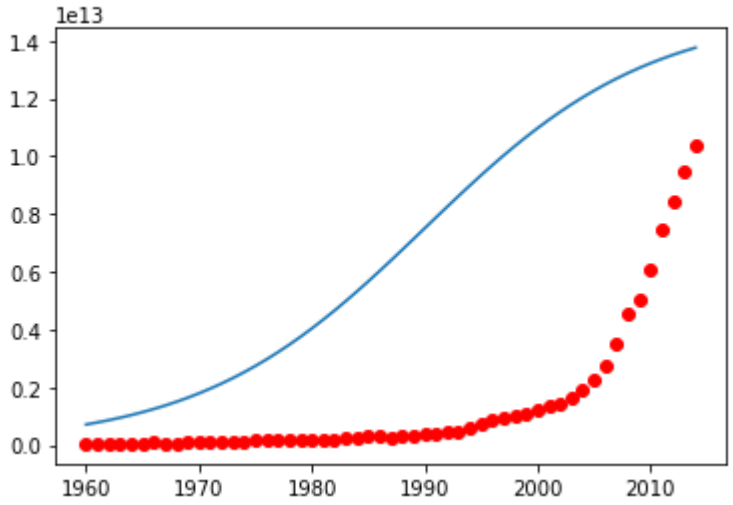
نبدأ ببناء النموذج مع تعريف دالة لهذا التابع :

[11]:
```
def sigmoid(x, Beta_1, Beta_2):
     y = 1 / (1 + np.exp(-Beta_1*(x-Beta_2)))
     return y
```

ثم نضع قيم أولية لثوابت التابع لنرى مدى ملاحقته للبيانات لدينا :

[12]:
```
beta_1 = 0.10
beta_2 = 1990.0

#logistic function
Y_pred = sigmoid(x_data, beta_1 , beta_2)

#plot initial prediction against datapoints
plt.plot(x_data, Y_pred*15000000000000.)
plt.plot(x_data, y_data, 'ro')
```

[12]:
```
[<matplotlib.lines.Line2D at 0x7f2ab05d5080>]
```



كما نرى أنه ليس مناسب بحالته الأولية إذا فمهمتنا الآن إيجاد أفضل بارامترات له .

- نقوم بعملية تنظيم البيانات لتكون بنفس المجال وهي عملية normalize أي السنوات بالألاف أما GPD مرفوعة للقوة 13 كما نرى لذلك نحاول تنظيمها كالتالي :

[13]:
```
# Lets normalize our data
xdata =x_data/max(x_data)
ydata =y_data/max(y_data)
```

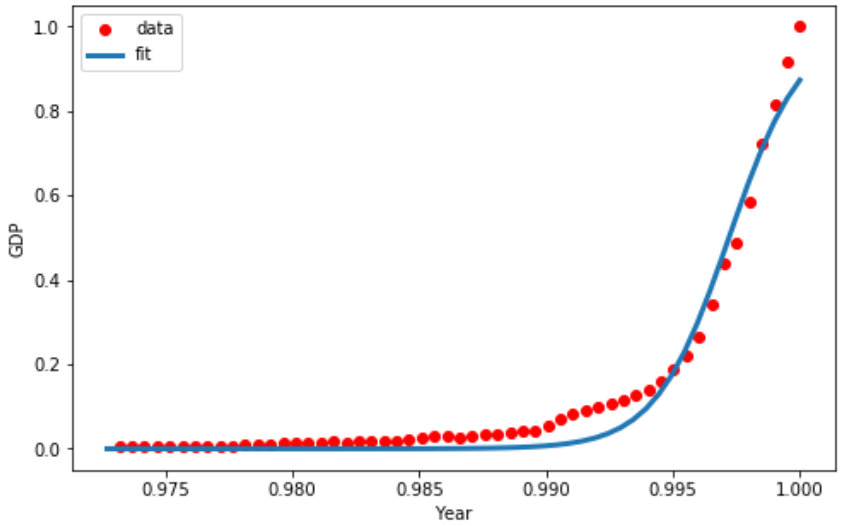
- لإجاد قيم بيتا المناسبة للمنحني نستخدم المكتبة scipy :

```
[14]: from scipy.optimize import curve_fit
      popt, pcov = curve_fit(sigmoid, xdata, ydata)
      #print the final parameters
      print(" beta_1 = %f, beta_2 = %f" % (popt[0], popt[1]))
```

```
beta_1 = 690.447527, beta_2 = 0.997207
```

- الآن نعوضها لنرى النتيجة :

```
[15]: x = np.linspace(1960, 2015, 55)
      x = x/max(x)
      plt.figure(figsize=(8,5))
      y = sigmoid(x, *popt)
      plt.plot(xdata, ydata, 'ro', label='data')
      plt.plot(x,y, linewidth=3.0, label='fit')
      plt.legend(loc='best')
      plt.ylabel('GDP')
      plt.xlabel('Year')
      plt.show()
```



نلاحظ أنه مناسب ولكن هذا لا يكفي فينبغي أن نحسب الدقة باستخدام نفس الأسلوب في التوقع الخطي :
وذلك بتقسيم البيانات لاختبار وتدريب و من ثم تدريب النموذج ثم حساب القيم المتوقع بالاعتماد على بيانات الاختبار ومن ثم حساب الدقة :

[18]:
```
# split data into train/test
msk = np.random.rand(len(df)) < 0.8
train_x = xdata[msk]
test_x = xdata[~msk]
train_y = ydata[msk]
test_y = ydata[~msk]

# build the model using train set
popt, pcov = curve_fit(sigmoid, train_x, train_y)

# predict using test set
y_hat = sigmoid(test_x, *popt)

# evaluation
print("Mean absolute error: %.2f" % np.mean(np.absolute(y_hat - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((y_hat - test_y) ** 2))
from sklearn.metrics import r2_score
print("R2-score: %.2f" % r2_score(y_hat , test_y) )
```

```
Mean absolute error: 0.03
Residual sum of squares (MSE): 0.00
R2-score: 0.98
```

بهذا نكون قد أنهينا التوقع الخطي و الغير خطي مع أمثلة عنها .

11-التصنيف Classification :

11-1- مقدمة :

يعتبر التصنيف في تعلم الآلة من قسم التعلم الإشرافي supervisor حيث من خلاله يمكن تصنيف عدة عناصر مجهولة (أي لا علم لنا عنها) ضمن مجموعات منفصلة (صفوف) .

يحاول التصنيف أن يتعلم العلاقة الموجودة بين مجموعة متغيرات features والهدف target ، حيث يكون الهدف متغير لغوي categorical variable بقيم منفصلة (متقطعة) discrete values ، وكمثال عن ذلك :

توقع منح القروض leans للعملاء في بنك أو مصرف ما :

لدينا بنك ولديه حالة عدم تسديد القروض لديه (تخلف عن الدفع default) :
لدينا البيانات التالية :

age	ed	employ	address	income	deb	credit	order	default
41	3	17	12	176	9.3	11.3	5,009	1
27	1	10	6	31	19.2	1.6	4,0	0
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	..	.
.	.	.	.	.	.	.	.	0

المتغير الأخير من النوع categorical وهو الهدف ونريد أن نتوقع هل سيدفع الزبون القرض المترتب عليه أم لا .
وعندها يحدد البنك منح offer أو منع decline العروض الأخرى لديه لهذا الزبون .
سنستخدم البيانات التي لدينا لبناء مصنّف classifier نمرر له بيانات أي زبون جديد ويعطي النتيجة إما 1 أي defaulter أو 0 أي not defaulter .
نرى أن نوع هذا التصنيف ثنائي binary لأنه يستخدم قيمتين في الخرج 1 أو 0 كما يمكننا بناء مصنّف لأجل عدة أصناف multi-class classification أي ليس ثنائي فقط كما في مثالنا السابق .
مثلا يكون لدينا بيانات مرضى ونريد أن نتوقع نوع العقار الذي سنعطيه للمريض الجديد:

age	sex	Bp	chars	Na	K	Drug
18	m	.	.	.	.	drugY
16	f	.	.	.	.	drugC
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	drugX

هذا المتغير الهدف categorical variable
حيث لدينا ثلاث تصنيفات للدواء المتوقع .

مجالات استخدام التصنيف في أمور عدة :

- تحديد لأي فئة أو نوع ينتمي الزبون
- توقع هل الزبون سيغير سلعته التي سيشتريها churn detection
- توقع هل يستجيب الزبون respond لعروض الشركة أم لا مثلا لمجموعة إعلانات : particular – advertising – camping .
- تمييز الكلام speech recognition .
- تمييز خط اليد hand writing recognition
- تمييز مقاييس حيوية لمريض bio-metric identification
- تصنيف المستندات documents classification

معظم هذه المشاكل تكون قائمة على الاختيار بين عدة احتمالات أو تصنيف النتائج للتعامل معها مثل تصنيف البريد الإلكتروني هل هو spam أو not spam (بريد مؤذي أم لا).

يوجد العديد من خوارزميات التصنيف منها :

- Decision Tree (ID3-C4.5-C5.0)
- Naïve Bayes
- Linear Discriminant Analysis
- K-Nearest Neighbor (KNN)
- Logistic Regression
- Neural Network (NN)
- Support Vector Machine (SVM)

2-11- خوارزمية التصنيف K-Nearest Neighbors KNN :

على فرض قام مزود الاتصالات بتقسيم زبائنه إلى فئات على أساس الخدمات التي يقدمها لهم ضمن الجدول التالي :

X : Independent

Y : dependent

id	region	age	material	address	ed	employ	retire	gender	income	customer categorial
0	2	44	1	9	4	5	0	m	2	1
1	3	33	9	6	2	3	1	f	3	4
..	..	..	..	..	..	..	0	.	..	..
..	..	..	..	..	..	..	..	.	..	..
8	..	..	..	..	..	..	..	.	..	?

target

لدينا خدمات للمشتركين تأخذ قيمتها المتغير Y كما يلي :

1- Basic service

2- E-service

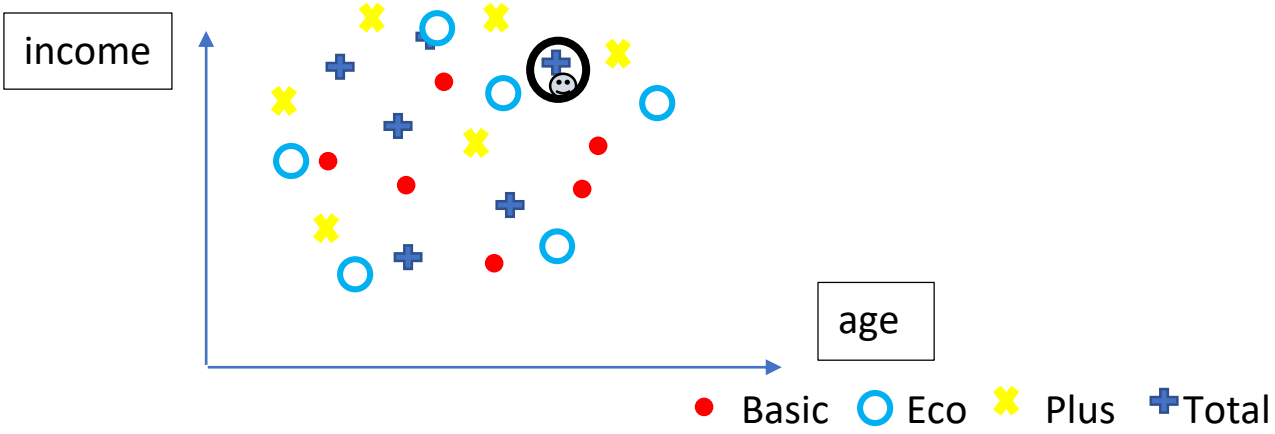
3- Phis-service

4- Total-service

وبالتالي الشركة ستستخدم البيانات الديموغرافية لتتوقع الميزات التي ستشترك فيها مجموعة المشتركين وتقديم الخدمات offers لهم : individual prospective customers

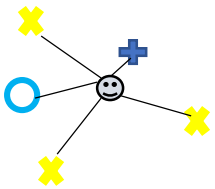
هذه المسألة تعتبر إحدى مسائل التصنيف حيث لدينا أربع مجموعات للخرج Y وسنبني نموذج توقع للقيم الجديدة أو المجهولة وسنستخدم KNN .

من أجل القيام بذلك مبدئيا نستخدم فقط عمودين features من المتغيرات x مثلا : age,income ونسميهم predictors ثم نرسم الزبائن كمجموعات بناء على هذين المتغيرين:



الآن نجد أن الزبون الجديد ☺أقرب للنقطة✚ أي فئة Total فهل يمكننا أن نقول أنه من هذه الفئة ؟ نعم نستطيع ذلك لأن هذه الفئة أول قريب للزبون الجديد first-nearest neighbor ولكن يأتي السؤال التالي : كم هي وثوقية هذا التصنيف ؟ طبعا هو تخمين ضعيف خصوصا إن كانت النقطة القريبة حالة خاصة أو شاذة outlier .

إذا نعود للمخطط السابق وبدل أن نحدد أول نقطة قريبة للزبون نقوم بتحديد أول خمس نقاط قريبة له ونرى عدد النقاط المتشابهة فيما بينها majority vote لتحديد صنف زبوننا الجديد:



نجد وجود ثلاثة من نفس الصنف وهذا أكثر منطقية more sense فمن قبل كان لدينا :

✚→1-NN ثم أصبح لدينا ✖→5-NN وبالتالي يكون تعرف خوارزمية KNN التالي :

هي أحد خوارزميات التصنيف التي تأخذ حزمة bunch من النقاط المحددة labeled points وتستخدمهم لتحديد هوية بقية النقاط حيث تعتمد على التشابه الموجود بين هذه النقاط القريبة لها. based on their similarity to other cases .

تسمى هذه النقاط القريبة من بعضها مع النقطة الجديدة بـ neighbors وهي تعتمد على الحالات المتشابهة من نفس الصنف وتكون قريبة على بعضها البعض لأن المسافة بين كل حالتين هي مقياس التباين dissimilarity .

وهنالك عدة طرق لقياس التشابه أو بشكل معاكس قياس التباين مثل طريقة Euclidian distance وسنتعرف على هذه الطرق لاحقا .

إذا نلخص مراحل خوارزمية KNN كالتالي :

1- نضع قيمة k أي عدد النقاط المتجاورة pick a value for k
2- حساب المسافة بين النقطة الجديدة أو المجهولة وبقية النقاط hold out from each of the cases in the dataset
3- نختار k-observation (ضمن بيانات التدريب training data) الأقرب للنقطة الجديدة أو المجهولة .
4- نتوقع حالة النقطة الجديدة باستخدام الحالات الشائعة لأقرب النقاط لها .

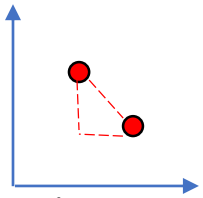
هنالك جزأين في هذا المسار يعطي انطباعا مضطربا للعمل ضمنه وهما :

- كيف نختار قيمة k الأولية ؟
- كيف نقيس التشابه (المسافات) بين الحالات أو النقاط.

لنرى مثال عن طريقة قياس التشابه بين زبونين (نقطتين) من مثالنا السابق :

customer1	customer2	
age	age	
54	50	$Dis = \sqrt{(54-50)^2} = 4$
income	income	
190	200	$Dis = \sqrt{(54-50)^2 + (190-200)^2} = 10.77$
education	education	
3	8	$Dis = \sqrt{(54-50)^2 + (190-200)^2 + (3-8)^2} = 11.87$

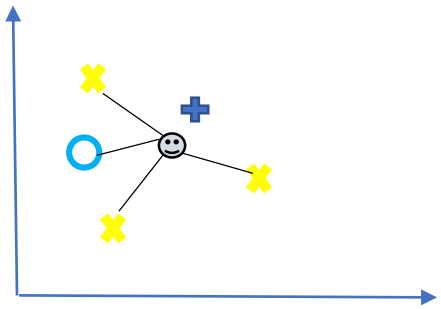
ولدينا منهما فقط ميزة واحدة age فيمكننا قياس المسافة بينهما بطريقة Euclidian distance وتسمى Minkowski distance كالتالي :



نلاحظ من الجدول السابق كيف يتم حساب التباين عند أخذ ميزة واحدة أو اثنتين أو ثلاثة للزبونين .

يمكننا عمل ذلك لجميع البيانات و عمل تقييس normalized لها ضمن مجال معين و هذا معتمد على طبيعة البيانات و التصنيف المستخدم .

وكما رأينا أن هذه الخوارزمية تعتمد على تحديد العناصر الأكثر قربا للنقطة المجهولة فبفرض نريد تحديد حالة الحالة **(؟)** على المخطط chart فما هي قيمة k المناسبة ؟



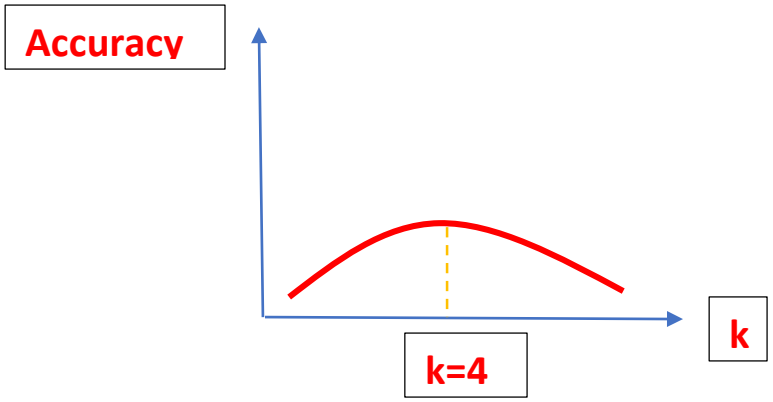
إذا بدأن بقيمة قليلة k = 1 فإنها ستكون أقرب للصنف Basic ✚ ولكنه خيار سيء لأن بقية النقاط حولها من الصنف ✖ على المخطط أي Total و وجود الخيار ○ هو حالة شاذة anomaly وبالتالي اختيار قيمة ضئيلة k=1 يؤدي إلى حالة over-fitting .

وهذا يعني أن النموذج لن يكون بشكله العام مفيدا للعينات من خارج البيانات out-of-sample وهي التي نستخدمها لتدريب النظام وبالتالي فسيكون غير مفيد للقيم الجديدة (المجهولة) ، ولكن ماذا لو اخترنا قيمة كبيرة مثلا k=20 ؟ كذلك سيكون غير ملائم !

إذا ما هي أفضل طريقة لاختيار قيمة k ؟

الحل الأشمل هو أن نحتفظ أو نحجز reserve جزء من البيانات لاختبار دقة النموذج ولفعل ذلك نختار k=1 ثم نستخدم البيانات التي حجزناها للاختبار لحساب دقة النموذج باستخدام

جميع بيانات الاختبار test set ونعيد العملية باختيار قيم أخرى لـ k حتى نحصل على أعلى دقة للنموذج وفي مثالنا السابق ستكون أفضل قيمة هي k=4



طريقة KNN تستخدم أيضا للتوقع بحالة القيم المستمرة وفي هذه الحالة يتم استخدام متوسط القيم المجاورة (الأقرب) لاكتشاف قيمة الحالة الجديدة .

مثل توقع سعر منزل حيث مواصفاته features : عدد الغرف – المساحة – الموقع - ....

يمكننا هنا إيجاد أقرب ثلاث منازل ليس فقط بأخذ المسافة بينها وإنما باعتبار مواصفاتهم البقية ثم نتوقع سعر المنزل الذي نريد تحديده بأخذ متوسط أسعار المنازل الثلاثة المجاورة له .

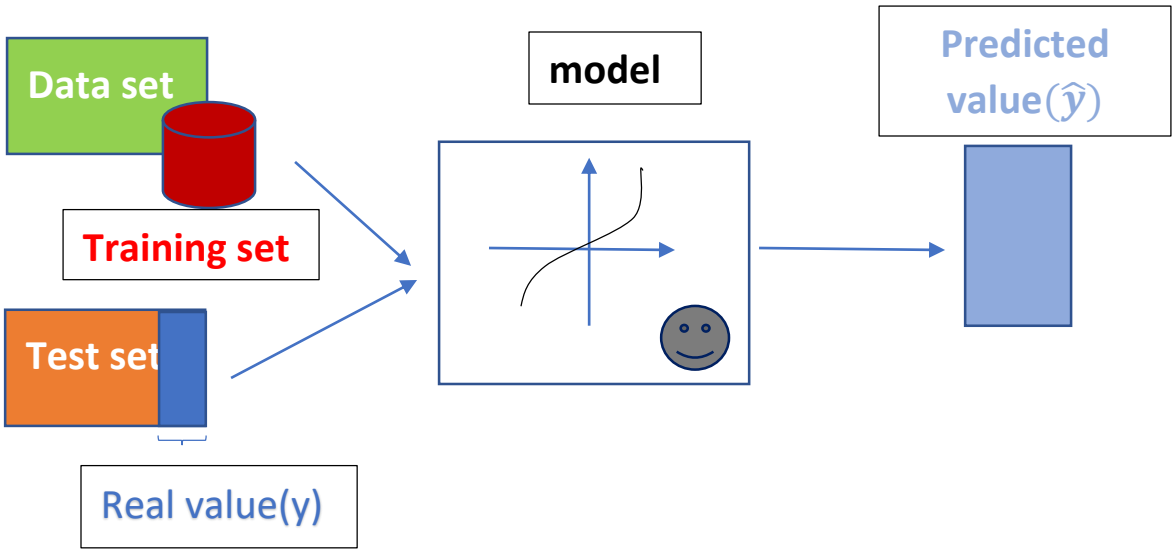
في النهاية يمكننا القول :

Very high (K=100)→ overly generalized model

Very low (K=1)→ highly complex model

- مقاييس التقييم في التصنيف Evaluation metrics in classification :

تحدد هذه المقاييس أداء النموذج فعلى فرض لدينا مجموعة بيانات لزبائن إحدى شركات الاتصالات :



و نريد تحديد النتيجة النهائية churns لعمل هذه الشركة !

بعد تدريب النموذج باستخدام بيانات التدريب training set نريد حساب دقة عمل هذا النموذج باستخدام بيانات الاختبار test set لنوجد القيم المتوقعة ومن ثم نقارن بين القيم الحقيقية في مجموعة الاختبار وبين القيم المتوقعة لتحديد دقة هذا النموذج .

يحدد لنا قياس الدقة أيضا الجزء الذي ينبغي ن نطوره في نموذجنا لتحسين أدائه و هنالك العديد من هذه المعايير سنتعرف على ثلاثة منها وهي :

Jaccard index , F_1-score , Log Loss

- Jaccard index :

وهي أبسط هذه الطرق لقياس دقة النموذج و تعرف أيضا بـ Jaccard similarity coefficient حيث نمثلها بالمعادلة :

$$J(y,\hat{y}) = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|} = \frac{|y \cap \hat{y}|}{|y| + |\hat{y}| - |y \cap \hat{y}|}$$

أي قسمة حجم التقاطع بين القيم المتوقعة و الحقيقية على اجتماعهما .

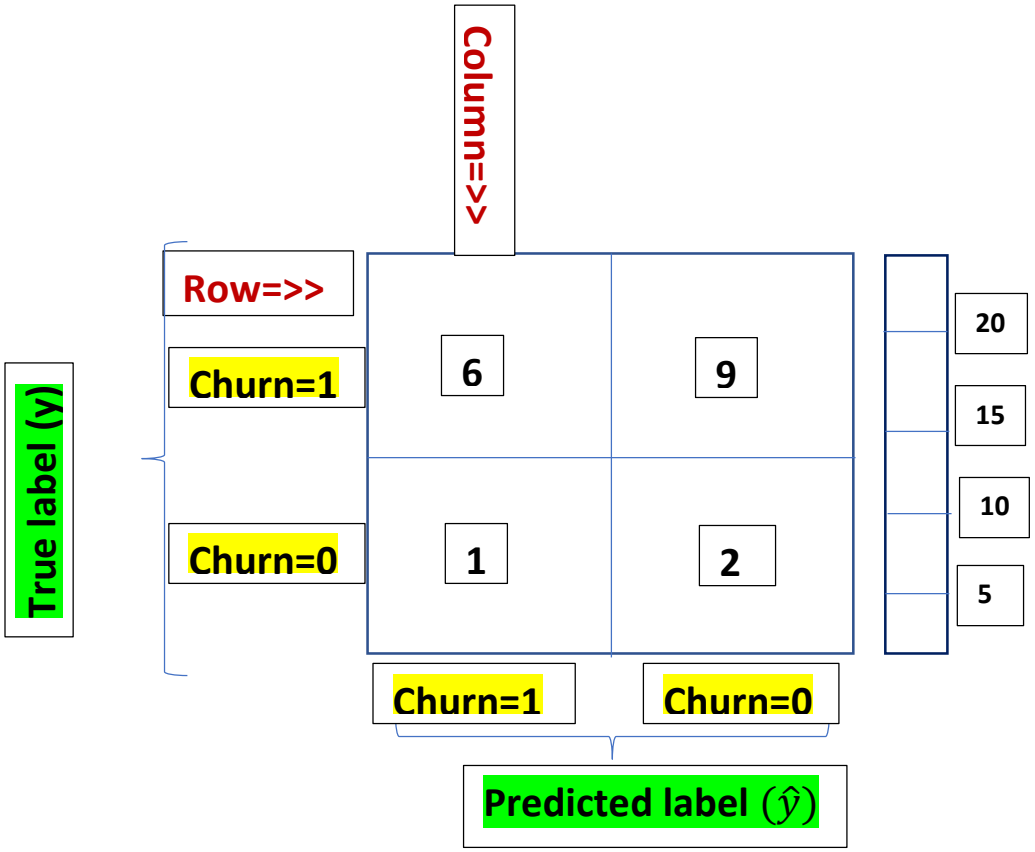
فمثلا لدينا عينات اختبار حقيقية y=[0,0,0,0,0,1,1,1,1,1] وما توقعه النموذج

$\hat{y} = [1,1,0,0,0,1,1,1,1,1]$ أي اختلف فقط بأول قيميتين و اشترك بباقي القيم الثمانية وبالتالي :

$$J(y,\hat{y}) = \frac{8}{10+10-8} = 0.66$$

يجدر بالذكر أنه عندما يوجد تشابه كامل بين مجموعتي القيم فإن دقة النموذج J=1 وعلى الخلاف عندما لا يوجد أي تشابه فالدقة معدومة J=0 .

لننظر لشكل آخر لمعنى دقة النموذج وهو confusion matrix (cm) فعلى فرض أن مجموعة الاختبار لدينا مصفوفة تحوي 40 سطر من البيانات فهذه المصفوفة cm تظهر التوقعات الخاطئة و الصحيحة مقارنة مع القيم الحقيقية .



كل سطر في هذه المصفوفة يبين القيم الحقيقية Actual/true label في مجموعة الاختبار test set وكل عمود يبين القيم المتوقعة predicted value من النموذج .

فلو نظرنا للسطر الأول :

فهو يمثل الزبائن التي قيمة churn الحقيقية لهم في مجموعة الاختبار تساوي 1 وبالتالي لو حسبنا قيمة churn لـ 15 زبون من أصل 40 لوجدنا أن النموذج توقع منهم 6 بقيمة 1 و 9 بقيمة 0 وكذلك الأمر للسطر الثاني .

النتيجة : لدينا 40 سطر من مجموعة الاختبار منهم 15 قيمته الحقيقية 1 و 25 قيمته الحقيقية 0 بحيث :

الـ 15 : 6 منهم توقعهم النموذج بشكل صحيح أي 1

نموذج سيء

9 = = = = = = = =بشكل خاطئ أي 0

الـ 25 : 24 منهم توقعهم النموذج بشكل صحيح أي 0

نموذج جيد

1 ==========خاطئ أي 1

إذا مصفوفة cm تعطي انطباعا عن قدرة النموذج على تصحيح توقعه أو فصل الأصناف عن بعضها البعض .

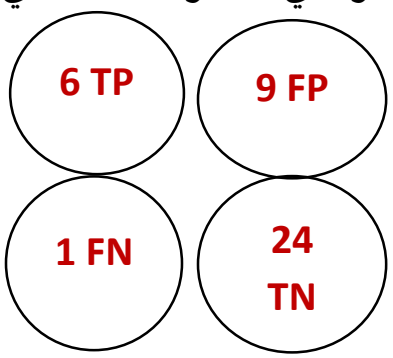
أما في المثال الخاص من التصنيف الثنائي binary (أي صنفين فقط 1 و 0) يمكننا عندها تفسير الأرقام الموجودة في مصفوفة cm كالتالي :

True positive TP (6)

False positive FP (9)

True negative TN (24)

False negative FN (1)



True توقع صحيح

False توقع غير صحيح

Positive أي حالته 1

Negative أي حالته 0

بالاعتماد على كمية كل قسم نستطيع حساب الدقة كما يلي :

Precision = TP/(TP+FP) (تحدد دقة التوقع للحالة)

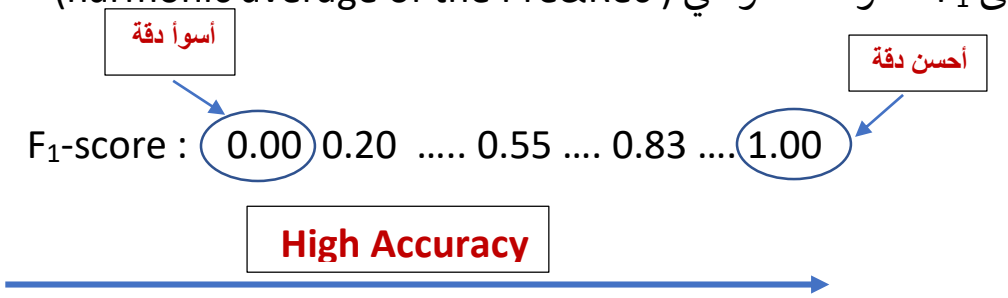
Recall = TP/(TP+FN) (تحدد مدى إيجابية الحالة)

نحسب هاتين القيميتين للحالات التي لدينا :

classes	precision	recall	F_1-score
0	0.73	0.96	0.83
1	0.86	0.40	0.55

$$F_1\text{-score} = 2*(Prc*Rec)/(Prc+Rec)$$

تدعى F_1 المتوسط التوافقي (harmonic average of the Pre&Rec).



A_{vg} Accuracy = (0.83+0.55)/2= 0.72

لننتقل لنوع آخر من مقاييس الدقة للتصنيف :

أحيانا يكون خرج المصنّف هو احتمال وجود صنف عوضا عن صنف آخر مثلا : في التوقع الرمزي logistic regression يكون الخرج احتمالية وجود الزبون the probability of customer churn يعني (1 or yes) وهذه قيمة الاحتمالية بين 0 و 1 .

churn Actual labels (y)		Predicted churn (probability)	log loss	state
1		0.91	0.11	good
1	→ model	→ 0.13	→ 2.04	→ bad
0		0.7	0.04	v.good
0		0.8	0.26	good
0		0.6	0.56	medium

إذا في طريقة Logarithmic loss أو تدعى Log loss تقيس أداء المصنّف عندما يكون الخرج المتوقَّع هو قيمة احتمالية بين 0 و 1 .

فمثلا في الجدول السابق :

القيمة الثانية الحقيقية هي 1 و احتمالية توقعها كانت 0.13 فهي سيئة وينتج عنها قيمة log loss عالية حيث تحسب كالتالي :

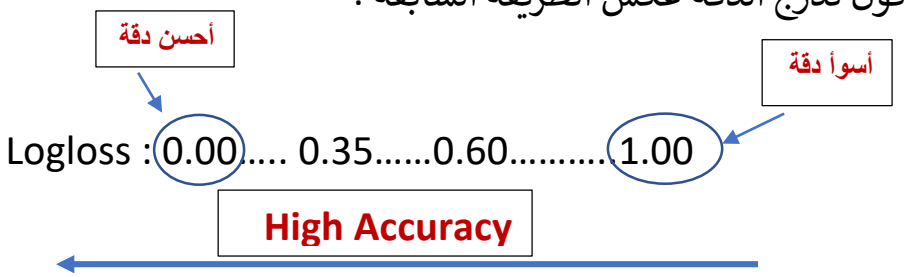
$$y * \log(\hat{y}) + (1 - y) * \log(1 - \hat{y})$$

وتعبر هذه القيمة عن المسافة التي تبعد فيها القيمة المتوقعة عن القيمة الحقيقية فكلما كانت كبيرة كان التوقع سيء .

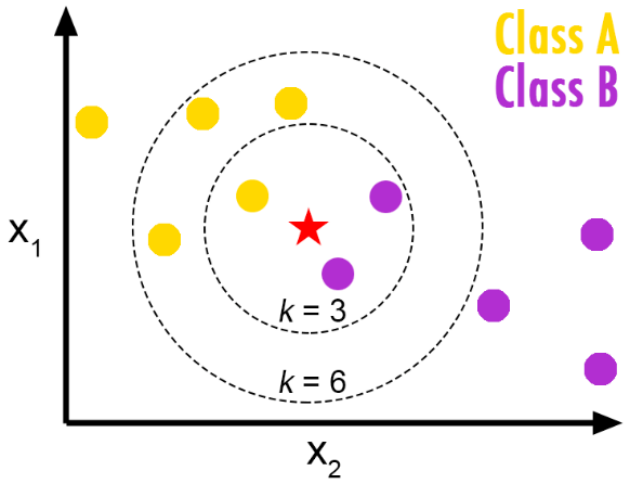
عند حساب قيمة log loss لكل سطر كما سبق نحسب المتوسط الحسابي لكل الصفوف في مجموعة الاختبار :

$$LogLoss = -\frac{1}{n}\sum(y * \log(\hat{y}) + (1-y) * \log(1-\hat{y}))$$

هنا يكون تدرج الدقة عكس الطريقة السابقة :



لدينا مثال عن زبائن شركة ونحاول توقع أحد الزبائن الجدد في أي صنف موجود وسنمثل ذلك بيانيا كالتالي :



نجد أن النجمة هي الزبون الذي نريد توقع صنفه وبالتالي رأينا عندما k=3 و k=6 كيف يتغير صنف النجمة التي نبحث عنه لذلك سنبدأ في توقع ذلك باستخدام البايثون في بيئة jupyter

نبدأ باستيراد المكتبات اللازمة :

```
[1]: import itertools
     import numpy as np
     import matplotlib.pyplot as plt
     from matplotlib.ticker import NullFormatter
     import pandas as pd
     import numpy as np
     import matplotlib.ticker as ticker
     from sklearn import preprocessing
     %matplotlib inline
```

نحمل بيانات شركة الاتصالات من الموقع :

```
[2]: !wget -O teleCust1000t.csv https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/ML0101ENv3/labs/teleCust1000t.csv
```

نظهر البيانات باستعمال pandas :

```
[3]: df = pd.read_csv('teleCust1000t.csv')
     df.head()
```

[3]:

	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	13	44	1	9	64.0	4	5	0.0	0	2	1
1	3	11	33	1	7	136.0	5	5	0.0	0	6	4
2	3	68	52	1	24	116.0	1	29	0.0	1	2	3
3	2	33	33	0	12	33.0	2	0	0.0	1	1	1
4	2	23	30	1	9	30.0	1	2	0.0	0	4	3

نحدد عدد الزبائن لكل صنف في الخرج :

```
[5]: df['custcat'].value_counts()
```

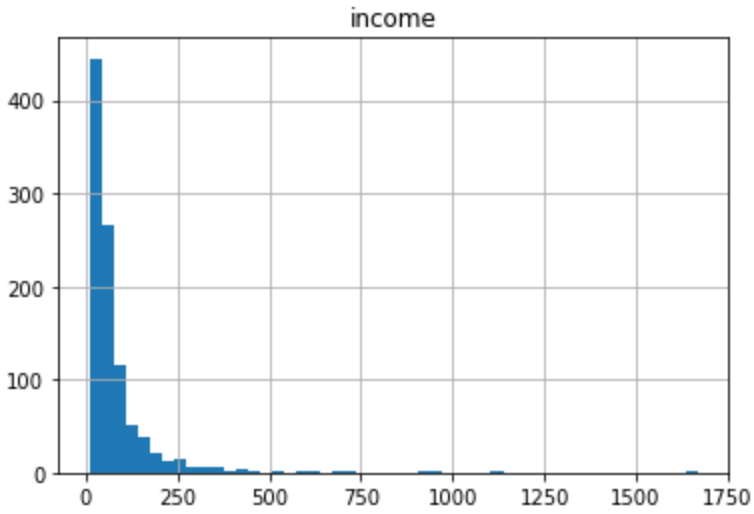
```
[5]: 3    281
     1    266
     4    236
     2    217
     Name: custcat, dtype: int64
```

281 Plus Service, 266 Basic-service, 236 Total Service, and 217 E-Service customers

ويمكننا رسمها بشكل بياني :

```
[6]: df.hist(column='income', bins=50)
```

```
[6]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f6217ea4a90>]],
           dtype=object)
```



لنحدد المتغيرات المستقلة X :

```
[7]: df.columns
```

```
[7]: Index(['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed',
       'employ', 'retire', 'gender', 'reside', 'custcat'],
      dtype='object')
```

```
[11]: X = df[['region', 'tenure','age', 'marital', 'address', 'income', 'ed', 'employ','retire', 'gender', 'reside']] .values  #.astype(float)
X[0:5]
```

```
[11]: array([[ 2., 13., 44.,  1.,  9., 64.,  4.,  5.,  0.,  0.,  2.],
       [ 3., 11., 33.,  1.,  7., 136.,  5.,  5.,  0.,  0.,  6.],
       [ 3., 68., 52.,  1., 24., 116.,  1., 29.,  0.,  1.,  2.],
       [ 2., 33., 33.,  0., 12., 33.,  2.,  0.,  0.,  1.,  1.],
       [ 2., 23., 30.,  1.,  9., 30.,  1.,  2.,  0.,  0.,  4.]])
```

ولنحدد ما هي المخرجات :

```
[12]: y = df['custcat'].values
y[0:5]
```

```
[12]: array([1, 4, 3, 1, 3])
```

نحتاج هنا لتقييس البيانات :

```
[13]: X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
      X[0:5]
```

```
[13]: array([[-0.02696767, -1.055125  ,  0.18450456,  1.0100505 , -0.25303431,
              -0.12650641,  1.0877526 , -0.5941226 , -0.22207644, -1.03459817,
              -0.23065004],
             [ 1.19883553, -1.14880563, -0.69181243,  1.0100505 , -0.4514148 ,
               0.54644972,  1.9062271 , -0.5941226 , -0.22207644, -1.03459817,
               2.55666158],
             [ 1.19883553,  1.52109247,  0.82182601,  1.0100505 ,  1.23481934,
               0.35951747, -1.36767088,  1.78752803, -0.22207644,  0.96655883,
              -0.23065004],
             [-0.02696767, -0.11831864, -0.69181243, -0.9900495 ,  0.04453642,
              -0.41625141, -0.54919639, -1.09029981, -0.22207644,  0.96655883,
              -0.92747794],
             [-0.02696767, -0.58672182, -0.93080797,  1.0100505 , -0.25303431,
              -0.44429125, -1.36767088, -0.89182893, -0.22207644, -1.03459817,
               1.16300577]])
```

نقوم بتقسيم البيانات للتدريب و الاختبار :

```
[14]: 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=4)
      print ('Train set:', X_train.shape,  y_train.shape)
      print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (800, 11) (800,)
Test set: (200, 11) (200,)
```

نستورد المكتبة الخاصة بخوارزمية KNN :

```
[15]: from sklearn.neighbors import KNeighborsClassifier
```

وندرب البيانات باختيار k=4 :

Lets start the algorithm with k=4 for now:

[16]:
```
k = 4
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh
```

[16]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                     weights='uniform')
```

ونتوقع الخرج :

[17]:
```
yhat = neigh.predict(X_test)
yhat[0:5]
```

[17]:
```
array([1, 1, 3, 2, 4])
```

نختبر دقة النموذج لكل من بيانات الاختبار و التدريب :

[18]:
```
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))
```

```
Train set Accuracy:  0.5475
Test set Accuracy:  0.32
```

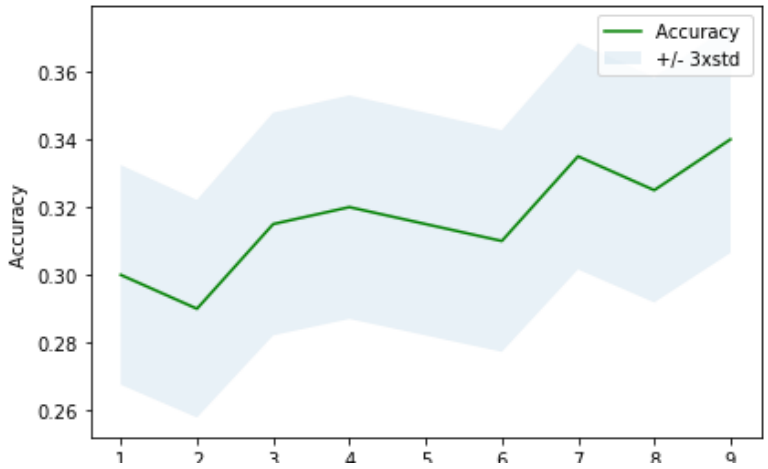
الآن نعيد الخوارزمية لأجل قيم مختلفة لـ k مثلا 10 :

[20]:
```
Ks = 10
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    #Train Model and Predict
    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)


    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

mean_acc
```

[20]:
```
array([0.3  , 0.29 , 0.315, 0.32 , 0.315, 0.31 , 0.335, 0.325, 0.34 ])
```

ومن ثم نرسم دقة النموذج مقارنة بقيم k المختلفة لنرى أعلى دقة عند أي قيمة لـ k :

[21]:
```
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
plt.legend(('Accuracy ', '+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Nabors (K)')
plt.tight_layout()
plt.show()
```



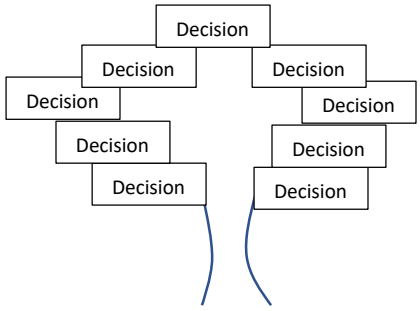
لنجد أن أعلى دقة للنموذج كانت عندما k=9 :

```
[22]: print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

```
The best accuracy was with 0.34 with k= 9
```

3-11- شجرة القرار Decision Trees :

مغزاها احتواء جميع القرارات الممكنة على شكل شجرة .



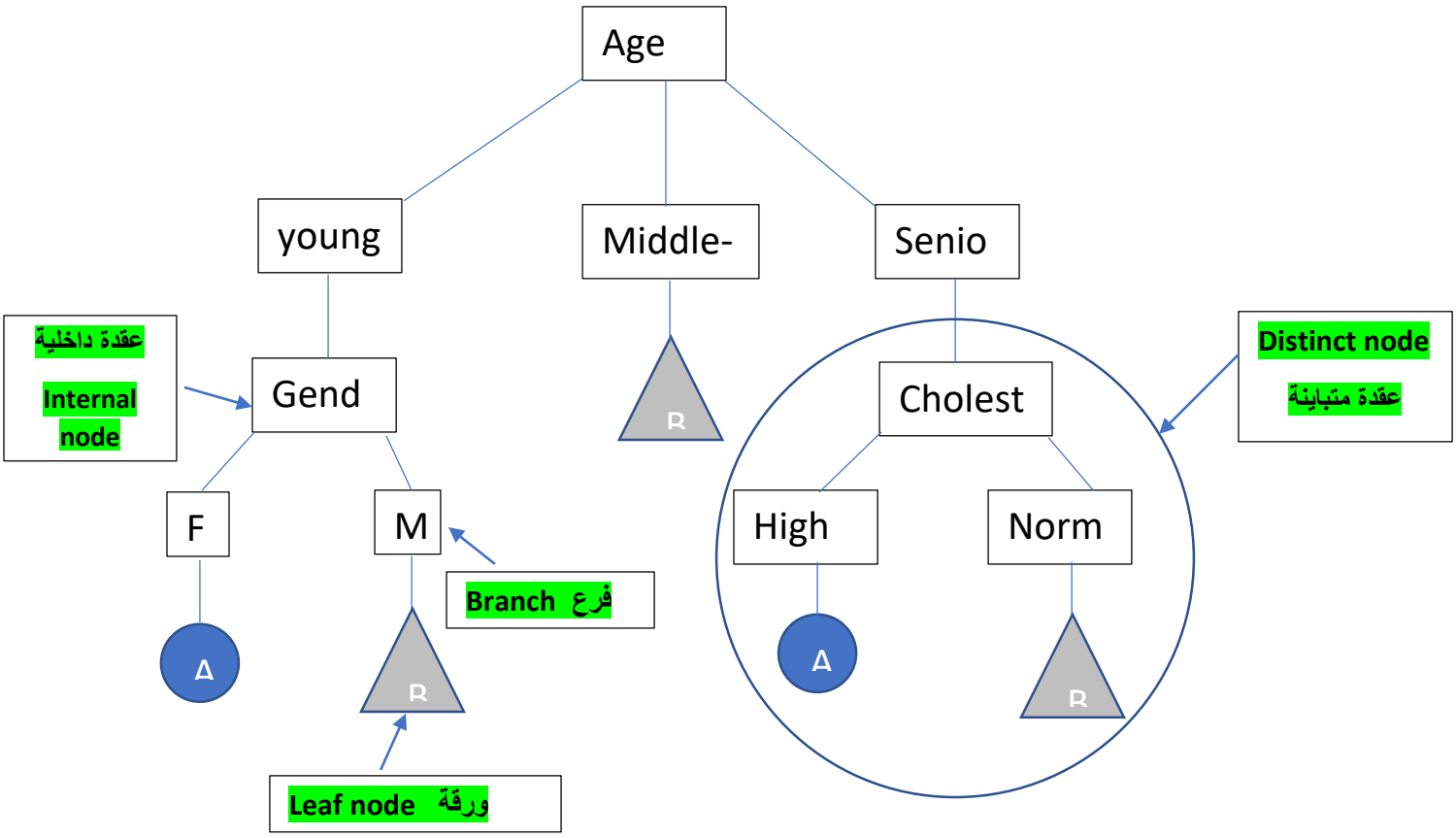
فكيف تفيد في التصنيف ؟ وكيف أنمي شجرة قراراتي ؟

على فرض أنك باحث طبي بجمع بيانات المرضى التالية :

Patient ID	Age	Gender	Bp	cholesterol	Drug
p1	young	F	High	Normal	A
.	.	.	.	.	.
.	.	.	.	.	.
p15	middle	M	Low	High	B

وخلال هذه الدراسة يتبين لنا ان هنالك مرضى يستجيبون للدواء A وآخرون للدواء B وتكمن مهمتنا في إيجاد نموذج لتوقع العلاج لأحد المرضى الجدد ويعاني من نفس المرض بحيث يكون لدينا عنه مجموعة الخصائص features السابقة Age,.... ويكون الهدف هو Drug فكما نرى هنا انه تصنيف ثنائي وسنعتمد طريقة Decision Tree التي تستخدم بيانات التدريب لتتوقع النتائج .

يتم بناء شجرة القرار بتقسيم بيانات التدريب إلى عقد متباينة distinct nodes عن بعضها البعض .



كما نرى أن شجرة القرار تختبر الخاصية attribute مثلا Age وتفرّع عنها الحالات بالاعتماد على نتيجة الاختبار لهذه الخاصية أو لقيمها .

فكل عقدة داخلية تخضع للاختبار و كل فرع يكون نتيجة الاختبار و كل ورقة عقدة تحدد صنف المريض (أو نوع عقاره) .

الآن ما هي طريقة بناء شجرة القرار ؟

يتم بناؤها بأخذ قيم الخصائص attributes واحدة تلو الأخرى بعين الاعتبار :

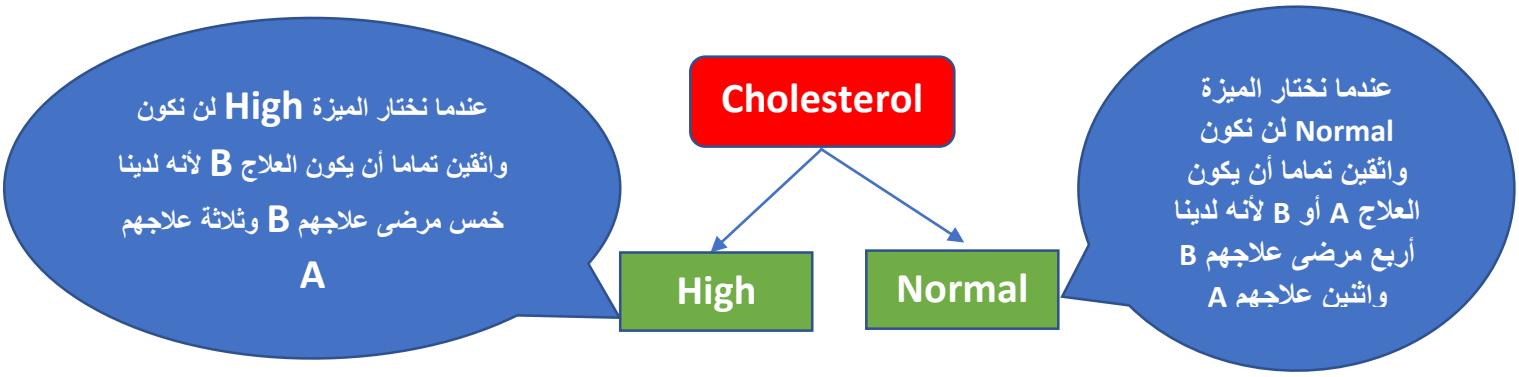
1- Choose an attribute from your dataset
2- Calculate the significance of attribute in splitting of data
3- Split data based on the value of the best attribute.
4- Go step 1.

1- نختار واحدة من الخصائص في البيانات لدينا .
2- نحسب (ثقل – وزن – دلالة) الصفة التي اخترناها في تقسيم البيانات لنرى تأثير هذه الصفة هل هو موجود أم لا .
3- تقسيم البيانات بالاعتماد على قيمة أفضل خاصية .
4- نكرر العملية في كل فرع branch لبقية الخاصيات features المتبقية .

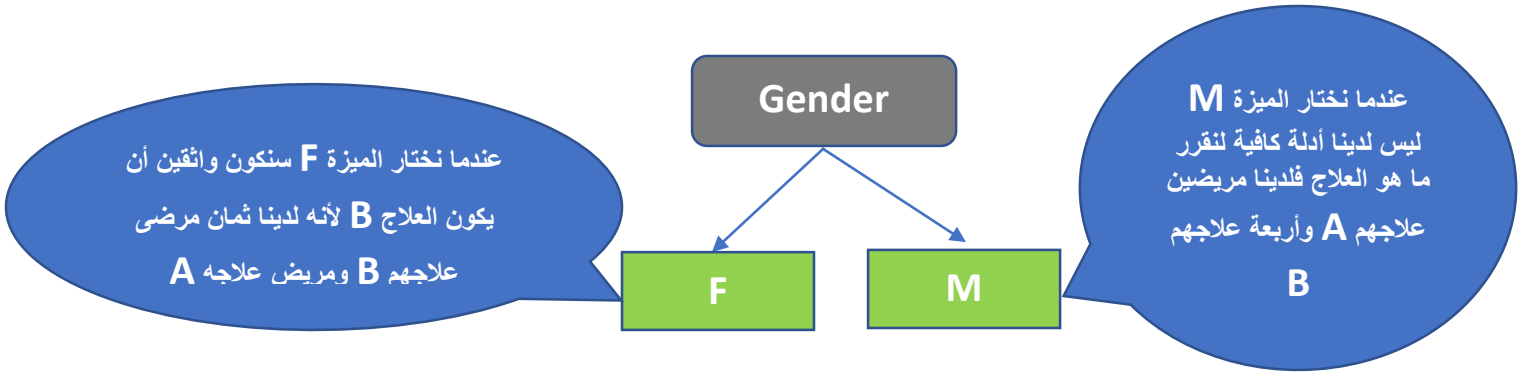
4-11- بناء شجرة Building Decision Trees :

يتم بناء الشجرة بتكرار التقسيم recursive partition لتصنيف البيانات لنقل لدينا 14 مريض في مجموعتنا السابقة و تختار الخوارزمية أغلب الميزات المتوقعة predictive features التي نقسم البيانات بناء عليها أي تختار أي أهم الصفات التي نعتمد عليها لتقسيم البيانات و تصنيفها .

لنبدأ مثلا بصفة cholesterol attribute و بناء عليها سيتم تقسيم البيانات لقسمين كما رأينا في الجدول في مثالنا السابق normal,high :



وبالتالي اختيار هذه الميزة سيء للتقسيم Bad attribute selection for split فلننتقل الآن لخاصية أخرى مثل Gender :



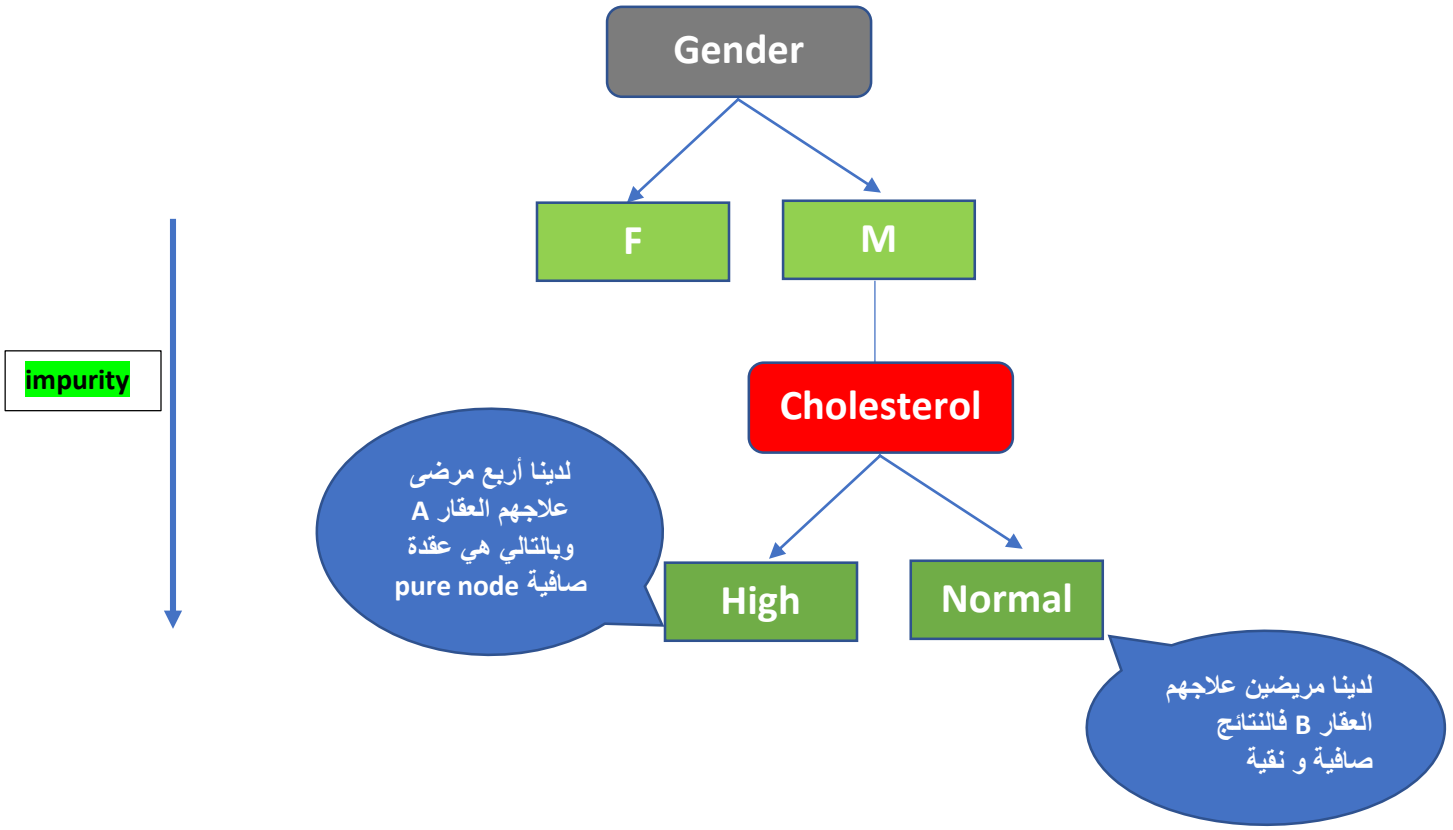
وعلى كل حال هذه الخاصية Gender أفضل من سابقتها لنعتمدها في تقسيم البيانات لأنها أقل ضجيجا وأكثر كفاءة .

Gender attribute is more significant than Cholesterol or more predictive (Less Impurity = Lower Entropy)

فالقدرة على التوقع productiveness تعتمد على إنقاص التشوه impurity في العقد nodes لذلك نحن نبحث عن الصفات features التي تخفض من هذا التشويه في العقد و الأوراق leaves .

لذلك وجدنا سابقا أن الميزة gender هي مرشح جيد good candidate لهذه الحالة وبالتالي سننتقل للخطوة التالية :

الآن لأجل النوع M في الميزة gender نختار الميزة Cholesterol لتقسيم البيانات مرة أخرى في شجرة فرعية subtree تالية :



نعتبر أن العقدة في الشجرة نقية عندما تكون محددة فقط بنوع الهدف المراد الوصول إليه وفي الحقيقة نستخدم تكرار التقسيم في بيانات الاختبار إلى أجزاء (مقاطع) وذلك بهدف تقليل التشوه في كل خطوة و التي يتم حساب impurity عن طريق حساب Entropy للبيانات في كل عقدة فما هي الـ Entropy ؟

هي مقدا العشوائية randomness في البيانات أو مقدار عدم ترتيب أو تنظيم disorder هذه البيانات ، وبالتالي نحن نبحث عن أقل قيمة لها في العقد فهي إذا تقيس التجانس homogeneity في العقد .

مثلا لدينا الآتي :

3 Drug B	1 Drug A
5 Drug B	& Drug A
Entropy is high (bad value)	Entropy is low (good value)
العشوائية كبيرة في القيم	العشوائية قليلة في القيم
4 Drug A	0 Drug A
4 Drug B	8 Drug B
Entropy = 1	Entropy =0
غير مطلوب	وهو مطلوب

إذا يمكننا حسابها كما يلي :

$$Entropy = -p(A)\log\big(p(A)\big) - p(B)\log(p(B))$$

حيث p نسبة (ratio,proportion) الصنف ، لنقوم بحساب Entropy للجدول عندنا قبل تقسيمها :

Patient ID	Age	Gender	Bp	cholesterol	Drug
p1	young	F	High	Normal	A
.	.	.	.	.	.
.	.	.	.	.	.
p15	middle	M	Low	High	B

لدينا خمسة أحداث occurrences لـ A و تسعة لـ B [9B,5A]:S ونحسب الآن :

E = -(9/14)log(9/14) – (5/14)log(5/14) = 0.940

هذا قبل تقسيم البيانات ، نبدأ بتقسيم البيانات باستخدام الخاصية Cholesterol :

Normal : 6B,2A→S[6B,2A]→E=0.811

High : 3B,3A→S[3A,3B]→E=1.00

نجري نفس العملية لكل الخاصيات لدينا...Age,Bp,Gender,

مثلا Gender يكون لدينا:

F : S[3B,4A]→ E=0.985

M : S[6B,1A]→ E=0.592

إذا أيهما أفضل Gender أم Cholesterol ؟

سنختار الخاصية التي تجعل البيانات المفيدة **Information Gain** في الشجرة عالية بعد التقسيم :

The tree with the higher **Information Gain** after splitting.

وهي المعلومات التي تزيد نسبة الوضوح بعد التقسيم و تكون حاصل الفرق بين قيمتي E قبل و بعد التقسيم لكل خاصية :

It is the information that can increase the level of certainty after splitting .

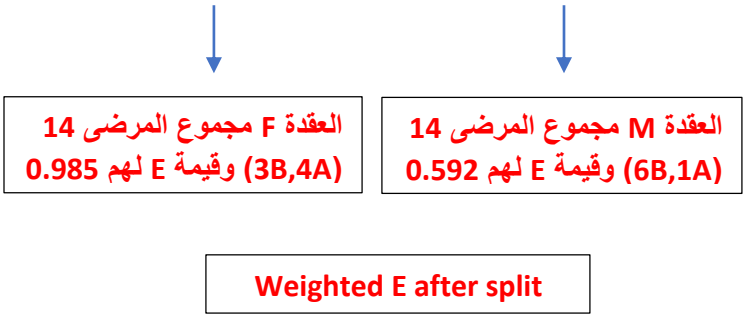
Information Gain (IG) = (Entropy before split) – (Weighted entropy after split)

فالتناسب بين IG و E متعاكس فعندما تتناقص E أي الفوضى في البيانات تزداد IG الوضوحية و العكس صحيح .

فبناء شجرة القرار يعتمد على إيجاد الخاصية ذات الربح الأعلى :

G(S,gender) = 0.940(entropy before split) –

[((3+4)/14)*0.985 + ((6+1)/14)*0.592] = 0.151



G(S, cholesterol) = 0.940 – [(8/14)*0.811 + (6/14)*1.0] = 0.048

نلاحظ أن الخاصية gender ربحها أعلى من خاصية cholesterol فسنعتمدها للتقسيم الأولي ونعيد نفس العملية لبقية الخصائص attributes .

(يجدر بالذكر أن عمل هذه الخطوات لن يكون يدويا طبعا وإنما نجريها هنا حتى نفهمها وإنما العمل يكون برمجي بالبايثون ضمن مكتباتها ^-^) .

سننفذ مثال عن مجموعة المرضى والعلاج الذي يتناولنه وكيف يمكننا بناء شجرة القرار لتصنيفهم باستخدام البايثون :

نبدأ باستيراد المكتبات اللازمة :

```
[1]: import numpy as np
     import pandas as pd
     from sklearn.tree import DecisionTreeClassifier
```

وتحميل بيانات المرضى :

```
[2]: !wget -O drug200.csv https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/ML0101ENv3/labs/drug200.csv
```

وإظهار أول خمس مرضى :

```
[3]: my_data = pd.read_csv("drug200.csv", delimiter=",")
     my_data[0:5]
```

[3]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY

نقوم بحجز بيانات السمات X :

```
[6]: X = my_data[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values
     X[0:5]
```

```
[6]: array([[23, 'F', 'HIGH', 'HIGH', 25.355],
            [47, 'M', 'LOW', 'HIGH', 13.093],
            [47, 'M', 'LOW', 'HIGH', 10.113999999999999],
            [28, 'F', 'NORMAL', 'HIGH', 7.797999999999999],
            [61, 'F', 'LOW', 'HIGH', 18.043]], dtype=object)
```

وبسبب وجود بيانات اسمية categorical نحتاج تحويلها لرقمية ولا نريد إهمالها :

```
[7]: from sklearn import preprocessing
     le_sex = preprocessing.LabelEncoder()
     le_sex.fit(['F','M'])
     X[:,1] = le_sex.transform(X[:,1])


     le_BP = preprocessing.LabelEncoder()
     le_BP.fit([ 'LOW', 'NORMAL', 'HIGH'])
     X[:,2] = le_BP.transform(X[:,2])


     le_Chol = preprocessing.LabelEncoder()
     le_Chol.fit([ 'NORMAL', 'HIGH'])
     X[:,3] = le_Chol.transform(X[:,3])

     X[0:5]
```

```
[7]: array([[23, 0, 0, 0, 25.355],
            [47, 1, 1, 0, 13.093],
            [47, 1, 1, 0, 10.113999999999999],
            [28, 0, 2, 0, 7.797999999999999],
            [61, 0, 1, 0, 18.043]], dtype=object)
```

الآن يمكننا ترميز بيانات الهدف y :

```
[8]: y = my_data["Drug"]
     y[0:5]
```

```
[8]: 0    drugY
     1    drugC
     2    drugC
     3    drugX
     4    drugY
     Name: Drug, dtype: object
```

نبدأ عملية تهيئة شجرة القرار باستيراد المكتبة :

```
[9]: from sklearn.model_selection import train_test_split
```

نقسم البيانات لتدريب واختبار في كل من X,y :

```
[14]: X_trainset,X_testset,y_trainset,y_testset=train_test_split(X,y,test_size=0.3,random_state=3
```

وطبعا ينبغي أن تكون مصفوفتي X_trainset,y_trainset بنفس الأبعاد وكذلك الأمر لأجل مصفوفتي X_testset,y_testset .

نبدأ بإنجاز نموذج التصنيف من نوع شجرة القرار بإنشاء أوبجكت من الكلاس :

```
[26]: drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
      drugTree # it shows the default parameters
```

```
[26]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
                             max_features=None, max_leaf_nodes=None,
                             min_impurity_decrease=0.0, min_impurity_split=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                             splitter='best')
```

نلاحظ أننا اخترنا عمق الشجرة 4 وهو يمكننا تغييره ورؤية الفارق في نتائج التصنيف .

ندرب النموذج على بيانات التدريب :

```
[27]: drugTree.fit(X_trainset,y_trainset)
```

```
[27]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
                             max_features=None, max_leaf_nodes=None,
                             min_impurity_decrease=0.0, min_impurity_split=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                             splitter='best')
```

الآن نجري التوقع :

```
[33]: predTree = drugTree.predict(X_testset)
```

You can print out **predTree** and **y_testset** if you want to visually compare the prediction to the actual values.

```
[29]: print (predTree [0:5])
      print (y_testset [0:5])
```

```
['drugY' 'drugX' 'drugX' 'drugX' 'drugX']
40     drugY
51     drugX
139    drugX
197    drugX
170    drugX
Name: Drug, dtype: object
```

نحسب الآن دقة النموذج :

Evaluation

Next, let's import **metrics** from sklearn and check the accuracy of our model.

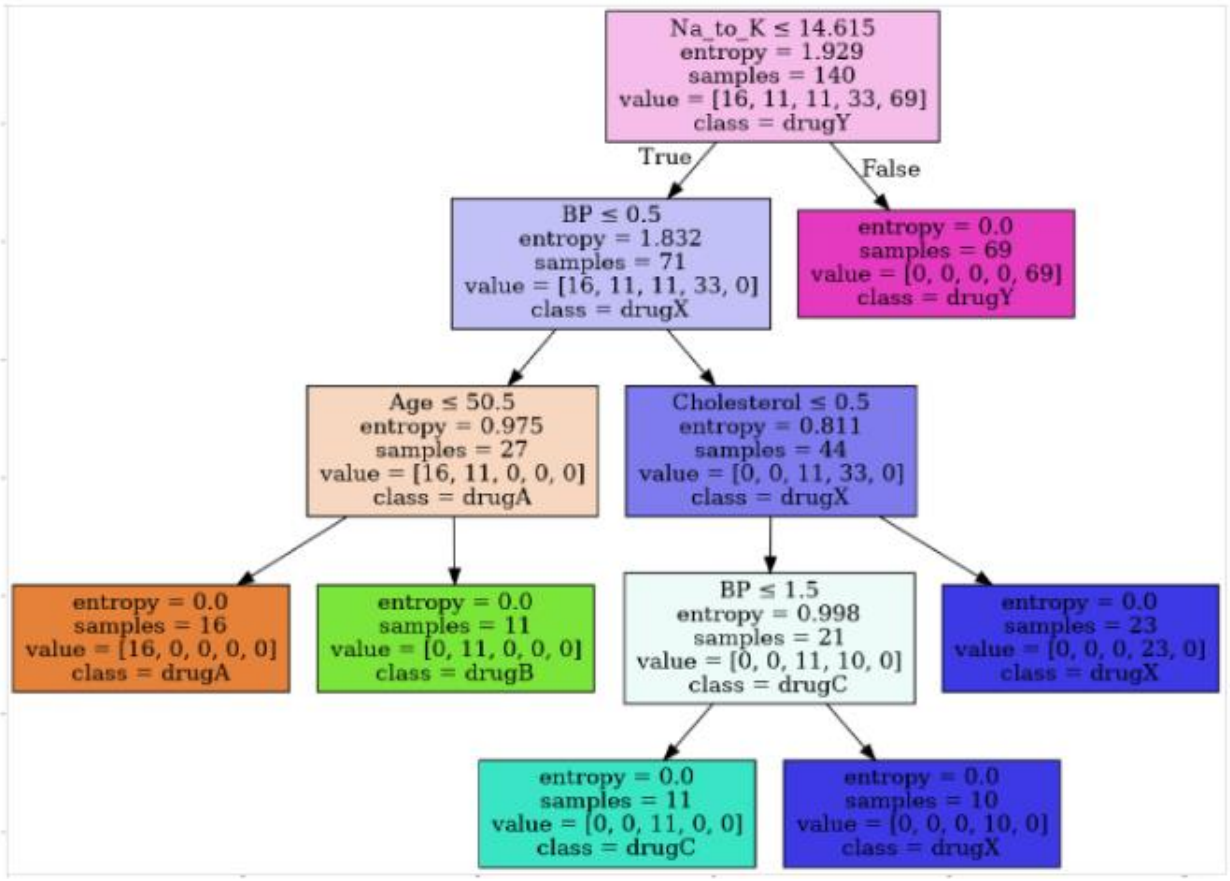
```
[30]: from sklearn import metrics
      import matplotlib.pyplot as plt
      print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, predTree))
```

```
DecisionTrees's Accuracy:  0.9833333333333333
```

لرسم هذه الشجرة يمكننا تنفيذ التالي :

```
[31]: from sklearn.externals.six import StringIO
      import pydotplus
      import matplotlib.image as mpimg
      from sklearn import tree
      %matplotlib inline
```

[36]:
```
dot_data = StringIO()
filename = "drugtree.png"
featureNames = my_data.columns[0:5]
targetNames = my_data["Drug"].unique().tolist()
out=tree.export_graphviz(drugTree,feature_names=featureNames,out_file=dot_data,...
...class_names= np.unique(y_trainset), filled=True,  special_characters=True,rotate=False)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png(filename)
img = mpimg.imread(filename)
plt.figure(figsize=(100, 200))
plt.imshow(img,interpolation='nearest')
```



12-التوقع المنطقي Logistic Regression :

في هذا النوع من التوقع يستخدم في التصنيف وبالتالي لدينا ثلاثة أسئلة :

- ما هو التوقع المنطقي ؟
- أين يستخدم ؟
- ما نوع المشاكل التي نحتاجه فيها ؟

تعتبر هذه الطريقة في التصنيف إحصائية في تصنيف السجلات records في مجموعة البيانات بالاعتماد على قيم الحقول fields .

Logistic regression is a classification algorithm for categorical variables.

بفرض لدينا بيانات شركة اتصالات ونريد تحليلها لنرى هل سيبقى المشترك بالخدمة أم لا ولماذا؟

id	ten	age	add	income	ed	employ	equip	churn
0	11.0	33	7	136.8	4	y	0	yes
1	4	55	4	234.6	.	.	.	no
2	.	.	.	.	.	.	.	yes
3	.	.	.	.	.	.	3	yes
4	7	35	.	.	8	n	4	?

العمود الأخير churn يدل أي الزبائن بقي في الشركة أو الخدمة وأيهم لغى اشتراكه وسنبني نموذجا للتنبؤ باستخدام logistic reg لتحديد حالة الزبون المستقبلية .

يمكننا استخدام واحد أو عدد من المتغيرات المستقلة التي لدينا (age,income,...) ويجدر بالذكر أن هذا التوقع يشبه التوقع الخطي ولكن هنا يمكننا التعامل مع المتغيرات النصية categorical .

فالتوقع الخطي يتوقع قيم مستمرة مثل ثمن منزل – ضغط دم – كمية استهلاك وقود أي رقمية بينما التوقع المنطقي يتوقع قيم ثنائية اسمية مثل (yes or no) (true or false) (successful or un...) فكلها تمثل (1 or 0) .

وهنا تكون المتغيرات الغير مستقلة مستمرة continuous وبحال كانت اسمية categorial فينبغي أن نحولها لقيم مستمرة عن طريق ترميزها indicator-coded أو نعتبرها وهمية dummy .

يستخدم هذا النوع من التوقع في التصنيف الثنائي أو المتعدد وفي عدة حالات مثل :

- توقع احتمالية أن يصاب الشخص بنوبة قلبية heart-attack من عدمها
- = = = = موت مريض مصاب بمرض عضال من عدمه .
- = = = = إصابة مريض بالسكري diabetes.
- = = = = أن يشتري الزبون منتج ما .
- = = = = فشل منتج ما أو نجاحه .

نلاحظ أن هذا النوع من التوقع ليس فقط يصنف النتائج و إنما كذلك يتوقع احتمالية حدوثها.

إذا لتخليص حالات استخدام التوقع المنطقي لدينا أربع حالات :

1- عندما يكون التصنيف ثنائي (0 أو 1) binary data
2- عندما نريد حساب احتمالية وقوع حدث ما أو احتمالية التنبؤ بقيمة ما probabilistic results عندها يعيد لنا النموذج قيمة الاحتمالية ضمن المجال [0,1] لمجموعة عينات من البيانات ثم يتم تشكيل الحالات المتبقية ضمن أصناف مقسمة بناء على الاحتمالية التي أوجدها النموذج .
3- إذا كانت البيانات قابلة للتقسيم بشكل شبه خطي linear decision boundary حيث يكون القرار الفاصل هو مستقيم line أو مستوي plane أي linearly separable ويمكن أن تكون تموجي hyper-plane .
مثلا لدينا مجموعة نقاط لها خاصيتين ولا يتم تمثيلها بكثير حدود كما كنا نمثلها في التوقع الخطي عندها يمكننا الحصول على متراجحة من الشكل :

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 > 0 \equiv half - plane \rightarrow easily\ plotable$$

مع أنه يمكننا باستخدام التوقع المنطقي إنجاز complex decision boundary باستخدام كثير حدود ولكنها خارج مجال دراستنا حاليا .

4- عندما نريد فهم تأثير impact الخصائص بحيث نريد اختيار المتغير (feature) الأفضل بالاعتماد على الأهمية الإحصائية لبارامترات النموذج الذي نبنيه بالتوقع المنطقي

The statistical significance of the logistic regression model parameters .

وبعد إيجاد البارامترات الأفضل عندها يكون للمتغير X ذو الوزن θ الأقرب للصفر التأثير الأقل على التنبؤ بينما المتغير ذو القيمة المطلقة الكبيرة لـ θ له التأثير الأكبر على التنبؤ.
أي سيعطي انطباعا عن تأثير المتغيرات المستقلة على المتغيرات الغير مستقلة عندما نقوم بضبط متغيرات مستقلة أخرى .

لو عدنا لجدول بيانات المشتركين بشركة الاتصال و قمنا بتحليلها عندها نجد التالي :
المتغيرات المستقلة X تنتمي لمجموعة الأعداد الحقيقية n rows * m columns
و المتغيرات الغير مستقلة y تنتمي للمجموعة {0,1} أي :

$$X \in R^{n*m} \ , \qquad y \in \{0,1\}$$

وبشكل مثالي فإن نتيجة التوقع هي احتمالية أن تكون قيمة الخرج y=1 أي :

$$\hat{y} = P(y = 1 \,|X)$$

مهما كانت قيم الميزات X لديه .
وبشكل مقابل أن الزبون ينتمي للصنف 0 عندما :

$$P(y = 0|X) = 1 - \ P(y = 1 \,|X)$$

13-مقارنة التوقع الخطي بالمنطقي logistic vs linear :

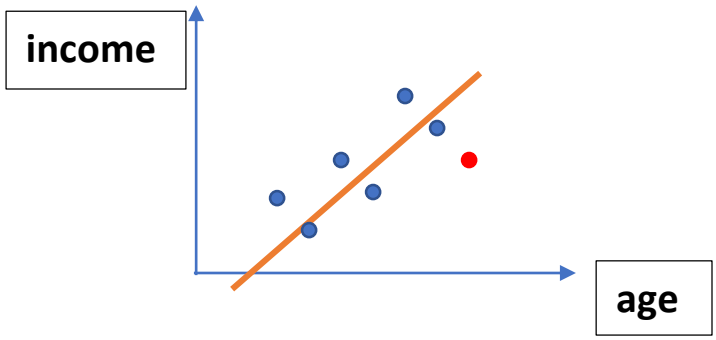
سنرى أنه لا يمكننا استخدام التوقع الخطي في مسائل التصنيف الثنائي و سنتعرف على دالة السيجمويد sigmoid التي هي الجزء الأهم في التوقع المنطقي .

في مثالنا السابق لمعلومات مشتركي شركة الاتصالات لنرى هل يمكن حل المسألة باستخدام التوقع الخطي :

id	ten	age	add	income	ed	employ	equip	churn
0	11.0	33	7		4	y	0	1
1	4	55	4		.	.	.	0
2	.	.	.		.	.	.	.
3	.	.	.		.	.	3	.
4	7	35	.		8	n	4	0

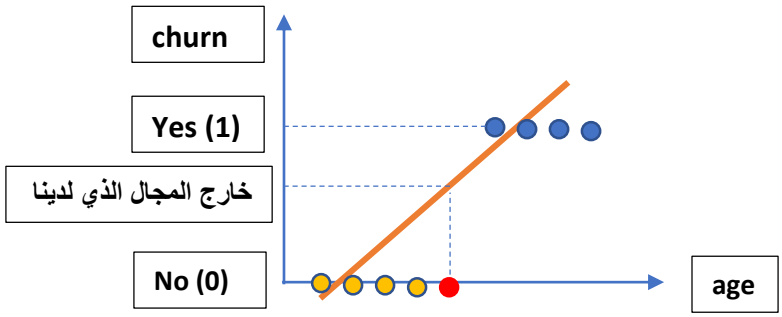
ولكن لن نقوم بتوقع الحقل الأخير churn كونها قيم متقطعة discrete وإنما سنقوم بتوقع الدخل income كونها قيم مستمرة continuous .

نختار متغير مستقل independent وليكن age وسنتوقع متغير غير مستقل وهو income فلنرسم هذه البيانات :



يمكننا باستخدام التوقع الخطي توقع حالة النقطة الحمراء عن طريق المعادلة a+bx التي تعرفنا عليها سابقا .

ولكن ماذا لو أردنا أن نستخدم نفس التوقع الخطي السابق لتوقع حالة churn ؟ لنقم برسم البيانات لنرى كيف تكون بوجود المستقيم السابق الذي يعبر عن التوقع الخطي !



كما نعلم قيم الحقل churn هي إما 0 أو 1 أي yes أو no وبالتالي لو أردنا توقع حالة النقطة الحمراء مثلا فإن المستقيم الذي أوجدنا معادلته في التوقع الخطي سابقا غير قادر على تحديد قيمتها 0 أو 1 وبالتالي لن ينفعنا في مسألة التصنيف الثنائي .

إذا معادلة الخط المستقيم في التوقع الخطي $\theta_0 + \theta_1 x$ لم تكن مناسبة وبالتالي نحتاج إعادة حساب القيم ثيتا لتناسب حالة التصنيف لدينا .

بما أنه لدينا بعد واحد يكون لدينا $\theta^T x = \theta_0 + \theta_1 x$ ومصفوفة ثيتا $\theta^T = [\theta_0, \theta_1]$ وفي حالتنا معادلة الخط : $\theta^T x = -1 + 0.1x$ وباستخدامها لتوقع حالة جديدة x_1=13 حيث x يعبر عن age يصبح لدينا :

$$P_1 = [13] \rightarrow \theta^T x = -1 + 0.1x_1 = 0.3$$

هذه قيمة التوقع التي أوجدها التوقع الخطي رياضيا للدخل 13 مع أننا نعلم أنه ينبغي أن تكون قيمة الخرج إما 0 أو 1 .

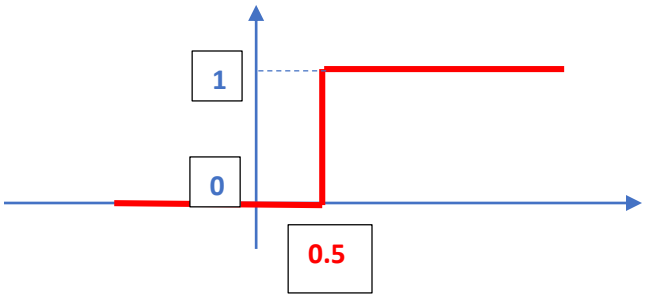
على فرض لدينا عتبة threshold تساوي 0.5 عندها يمكننا أن نكتب :

$$\hat{y} = \begin{cases} 0 \; if \; \theta^T x < 0.5 \\ 1 \; if \; \theta^T x \geq 0.5 \end{cases}$$

بناء على فرضية هذه العتبة فإن قيمة التوقع لدينا 0.3 يمكننا اعتبارها من الصنف (0) ولكن ما هي احتمالية فعلا أن يكون منتمي للصنف (0) .

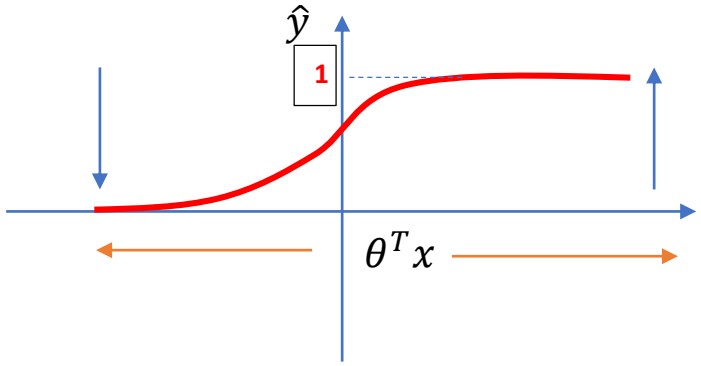
مما سبق نجد أن التوقع الخطي لن يفي بالغرض للتصنيف الثنائي كما بينا ذلك رياضيا و بيانيا.

ولكننا استخدمنا مفهوم العتبة لنستطيع الفصل بين القيم بسبب أن التوقع الخطي يعطي قيم مستمرة وسنفصل بينها على أساس العتبة مثل (0.5) :



وبالتالي اعتمدنا على تابع يدعى الخطوة الواحدية step للفصل بين النتائج مهما كانت موجبة أو سالبة فالذي له قيمة 1 أو 1000 أي موجبة ستنتمي للصنف (1) والعكس بالعكس وهذا ليس عملي أو منطقي !

فالحل لذلك أن نستعمل دالة غير دالة step وهي دالة السجمويد sigmoid بدل أن نستخدم قيم y مباشرة نستخدم قيم احتمالية وقوعها أي بدل المعادلة $\theta^T x = \theta_0 + \theta_1 x + \cdots$ نستخدم المعادلة : $\hat{y} = \sigma(\theta^T x) = \sigma(\theta_0 + \theta_1 x + \cdots)$ ولها الشكل البياني التالي :



المعادلة التي نعبر عنها لهذه الدالة هي :

$$\sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

نلاحظ أنه كلما أصبحت قيمة $\theta^T x$ كبيرة جدا أصبحت القيمة $e^{-\theta^T x} = 0$ أي :

$$\sigma(\theta^T x) = 1$$

والعكس كلما نقصت قيمة $\theta^T x$ أصبحت :

$$\sigma(\theta^T x) = 0$$

إذا دوما نتيجة دالة السيجمويد ضمن المجال [0,1] فنتيجة هذه الدالة يمكن اعتبارها احتمالية وهي مطابقة لما نريد :

P(y=1 | x) P(y=0 |x) = 1- P(y=1 | x)

إذا ما هو خرج نموذجنا الذي بنيناه للتصنيف باستخدام دالة السيجمويد بدل الدالة الواحدية؟

فمثلا احتمالية بقاء زبون مشترك بخدمة في شركة الاتصالات أي churn=1 فإن احتمالية ذلك بالنسبة لسمتيه income,age هي : p(churn=1 | income,age) = 0.8 وبالتالي احتمالية عدم بقائه p(churn=0 | income,age) = 1-0.8 = 0.2

تأتي الآن مهمتنا في تدريب النموذج لتكون بارامتراته مضبوطة بحيث يعمل بنفس الأسلوب ويعطي النتيجة المرغوبة للعناصر الجديدة (أي الزبائن الجديدة) .

ويتم ذلك بإيجاد أفضل قيم لـ ثيتا خلال عملية التدريب ضمن الخطوات التالية :

1- Initialize θ : eg. $\theta = [-1,2]$
2- Calculate $\hat{y} = \sigma(\theta^T x)$ for a customer : eg.
   $\hat{y} = \sigma(\theta^T x) = \sigma(([-1,2]^T * [age = 2, income = 5]) = 0.7$
   هذه هي احتمالية أن يكون الزبون في الصنف 1 .
3- Compare $\hat{y}, y$ → Error = 1-0.7 = 0.3
4- Calculate Error for all customers → Cost=J(θ)
   وهو تابع الكلفة .Cost Fun فكلما كانت قيمته صغيرة كان التوقع أفضل .
5- Change the θ to reduce the Cost Fun.
6- Go back to step 2.

وهنا يأتي سؤالين :

- كيف نغير قيم θ حتى نقلل من قيمة تابع التكلفة ؟
  هنالك طرق عديدة منها *Gradient Descent GD*
- متى نتوقف عن تكرار الخطوات السابقة ؟
  نتوقف عن طريق حساب دقة النموذج وأن تكون النتيجة مقنعة .

ننتقل الآن لتنفيذ مثالنا السابق توقع احتمالية أن يغادر الزبون شركة الاتصالات ؟ باستخدام التوقع المنطقي بالبايثون :

نستورد المكتبات اللازمة :

```
[1]: import pandas as pd
     import pylab as pl
     import numpy as np
     import scipy.optimize as opt
     from sklearn import preprocessing
     %matplotlib inline
     import matplotlib.pyplot as plt
```

نحمل البيانات من الموقع ونظهرها باستخدام *panda* :

```
!wget -O ChurnData.csv https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/ML0101ENv3/labs/ChurnData.csv
```

```
[3]: churn_df = pd.read_csv("ChurnData.csv")
     churn_df.head()
```

[3]:

	tenure	age	address	income	ed	employ	equip	callcard	wireless	longmon	...	pager	internet	callwait	confer	ebill	loglong	logtoll	lninc	custcat	churn
0	11.0	33.0	7.0	136.0	5.0	5.0	0.0	1.0	1.0	4.40	...	1.0	0.0	1.0	1.0	0.0	1.482	3.033	4.913	4.0	1.0
1	33.0	33.0	12.0	33.0	2.0	0.0	0.0	0.0	0.0	9.45	...	0.0	0.0	0.0	0.0	0.0	2.246	3.240	3.497	1.0	1.0
2	23.0	30.0	9.0	30.0	1.0	2.0	0.0	0.0	0.0	6.30	...	0.0	0.0	0.0	1.0	0.0	1.841	3.240	3.401	3.0	0.0
3	38.0	35.0	5.0	76.0	2.0	10.0	1.0	1.0	1.0	6.05	...	1.0	1.0	1.0	1.0	1.0	1.800	3.807	4.331	4.0	0.0
4	7.0	35.0	14.0	80.0	2.0	15.0	0.0	1.0	0.0	7.10	...	0.0	0.0	1.0	1.0	0.0	1.960	3.091	4.382	3.0	0.0

5 rows × 28 columns

نستخدم جزء من البيانات ونجعل الأرقام صحيحة في الهدف *churn* حتى يكون التصنيف ثنائي 0 أو 1 :

```
[4]: churn_df = churn_df[['tenure','age','address','income','ed','employ','equip','callcard','wireless','churn']]
     churn_df['churn'] = churn_df['churn'].astype('int')
     churn_df.head()
```

[4]:

	tenure	age	address	income	ed	employ	equip	callcard	wireless	churn
0	11.0	33.0	7.0	136.0	5.0	5.0	0.0	1.0	1.0	1
1	33.0	33.0	12.0	33.0	2.0	0.0	0.0	0.0	0.0	1
2	23.0	30.0	9.0	30.0	1.0	2.0	0.0	0.0	0.0	0
3	38.0	35.0	5.0	76.0	2.0	10.0	1.0	1.0	1.0	0
4	7.0	35.0	14.0	80.0	2.0	15.0	0.0	1.0	0.0	0

نحدد السمات X والخرج y :

[5]:
```
X = np.asarray(churn_df[['tenure', 'age', 'address', 'income', 'ed', 'employ', 'equip']])
X[0:5]
```

[5]:
```
array([[ 11.,  33.,   7., 136.,   5.,   5.,   0.],
       [ 33.,  33.,  12.,  33.,   2.,   0.,   0.],
       [ 23.,  30.,   9.,  30.,   1.,   2.,   0.],
       [ 38.,  35.,   5.,  76.,   2.,  10.,   1.],
       [  7.,  35.,  14.,  80.,   2.,  15.,   0.]])
```

[6]:
```
y = np.asarray(churn_df['churn'])
y [0:5]
```

[6]:
```
array([1, 1, 0, 0, 0])
```

ثم نقوم بتقييسها normalize كون مجال الخرج بين 0 أو 1 والدخل أرقام حقيقية :

[7]:
```
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)
X[0:5]
```

[7]:
```
array([[-1.13518441, -0.62595491, -0.4588971 ,  0.4751423 ,  1.6961288 ,
        -0.58477841, -0.85972695],
       [-0.11604313, -0.62595491,  0.03454064, -0.32886061, -0.6433592 ,
        -1.14437497, -0.85972695],
       [-0.57928917, -0.85594447, -0.261522  , -0.35227817, -1.42318853,
        -0.92053635, -0.85972695],
       [ 0.11557989, -0.47262854, -0.65627219,  0.00679109, -0.6433592 ,
        -0.02518185,  1.16316   ],
       [-1.32048283, -0.47262854,  0.23191574,  0.03801451, -0.6433592 ,
         0.53441472, -0.85972695]])
```

تقسيم البيانات للتدريب و الاختبار :

```
[8]: 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=4)
     print ('Train set:', X_train.shape,  y_train.shape)
     print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (160, 7) (160,)
Test set: (40, 7) (40,)
```

ندرب النموذج باستخدام بيانات التدريب:

```
[9]: from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import confusion_matrix
     LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
     LR
```

```
[9]: LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
                        intercept_scaling=1, max_iter=100, multi_class='warn',
                        n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
                        tol=0.0001, verbose=0, warm_start=False)
```

ونجري التوقع باستخدام بيانات الاختبار :

```
[10]: yhat = LR.predict(X_test)
      yhat
```

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

نوجد مصفوفة احتمالية التصنيف حيث العمود الأول احتمالية أن يكون من الصنف 1 و العمود الثاني احتمالية أن يكون من الصنف 0 :

[11]:
```
yhat_prob = LR.predict_proba(X_test)
yhat_prob
```

[11]:
```
array([[0.54132919, 0.45867081],
       [0.60593357, 0.39406643],
       [0.56277713, 0.43722287],
       [0.63432489, 0.36567511],
       [0.56431839, 0.43568161],
       [0.55386646, 0.44613354],
       [0.52237207, 0.47762793],
       [0.60514349, 0.39485651],
       [0.41069572, 0.58930428],
       [0.6333873 , 0.3666127 ],
       [0.58068791, 0.41931209],
       [0.62768628, 0.37231372],
       [0.47559883, 0.52440117],
       [0.4267593 , 0.5732407 ],
       [0.66172417, 0.33827583],
       [0.55092315, 0.44907685],
       [0.51749946, 0.48250054],
       [0.485743  , 0.514257  ],
       [0.49011451, 0.50988549],
       [0.52423349, 0.47576651],
       [0.61619519, 0.38380481],
       [0.52696302, 0.47303698],
       [0.63957168, 0.36042832],
       [0.52205164, 0.47794836],
       [0.50572852, 0.49427148],
       [0.70706202, 0.29293798],
       [0.55266286, 0.44733714],
       [0.52271594, 0.47728406],
       [0.51638863, 0.48361137],
       [0.71331391, 0.28668609],
       [0.67862111, 0.32137889],
       [0.50896403, 0.49103597],
       [0.42348082, 0.57651918],
       [0.71495838, 0.28504162],
       [0.59711064, 0.40288936],
       [0.63808839, 0.36191161],
       [0.39957895, 0.60042105],
       [0.52127638, 0.47872362],
       [0.65975464, 0.34024536],
       [0.5114172 , 0.4885828 ]])
```

نحسب دقة النموذج باستخدام jaccard index :

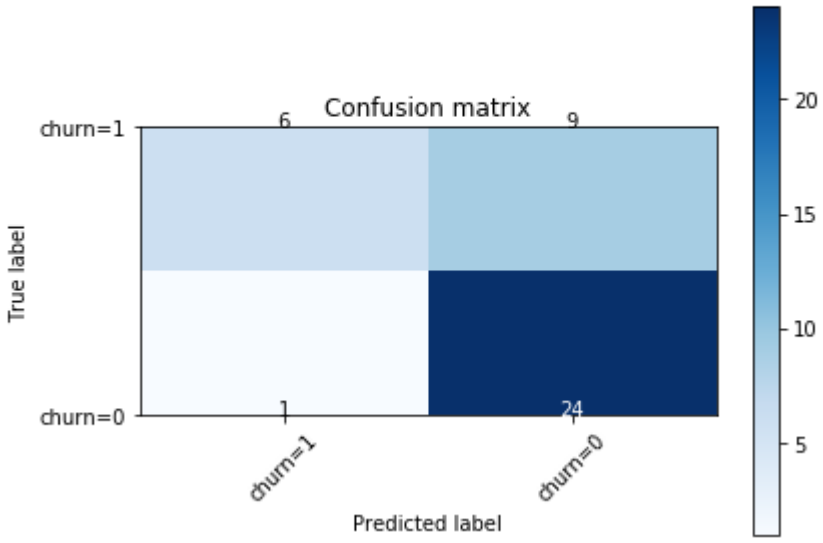
```
[12]: from sklearn.metrics import jaccard_similarity_score
      jaccard_similarity_score(y_test, yhat)
```

```
[12]: 0.75
```

وكذلك حساب cm :

```
[14]: # Compute confusion matrix
      cnf_matrix = confusion_matrix(y_test, yhat, labels=[1,0])
      np.set_printoptions(precision=2)


      # Plot non-normalized confusion matrix
      plt.figure()
      plot_confusion_matrix(cnf_matrix, classes=['churn=1','churn=0'],normalize=False,title='Confusion matrix')
```

```
Confusion matrix, without normalization
[[ 6  9]
 [ 1 24]]
```



يمكننا أن نجرب توقع قيمة باستخدام logloss :

```
[16]: from sklearn.metrics import log_loss
      log_loss(y_test, yhat_prob)
```

```
[16]: 0.6017092478101185
```

كما يمكننا طباعة تقرير عن F1-score :

```
[17]: print (classification_report(y_test, yhat))
```

	precision	recall	f1-score	support
0	0.73	0.96	0.83	25
1	0.86	0.40	0.55	15
micro avg	0.75	0.75	0.75	40
macro avg	0.79	0.68	0.69	40
weighted avg	0.78	0.75	0.72	40

حيث :

precision = TP / (TP + FP)

Recall = TP / (TP + FN)

14- التصنيف Support Vector Machine SVM :

ليكن لدينا بيانات عن الألوف من خلايا البشر الذين يحيط بهم خطر الإصابة بمرض السرطان:

Id	clump	unitsize	unitshape	….	….	class
100025	*5*	*1*	*1*	…	…	benign
…	*5*	*4*	*3*		…	malignant
…	….	…	…	….	…	….
1000015	*3*	*4*	*2*	…	…	*?*

نريد توقع حالة مريض جديد (هل ورم حميد benign أم ورم خبيث malignant) لدينا بيانات خلاياه .

يمكننا استخدام SVM لهذا الغرض حيث تعتبر خوارزمية تصنيف إشرافية تعتمد على إيجاد الفاصل بين الأصناف supervised algorithms by finding a separator وذلك بداية من خلال :

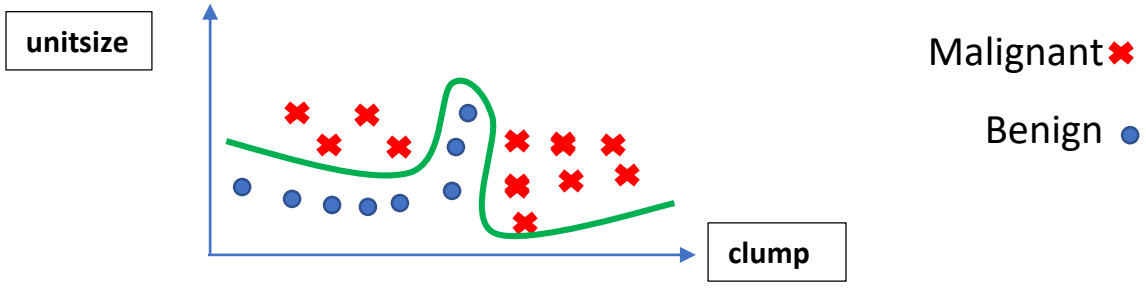
1- Mapping data to a high-dimensional feature space

تخطيط البيانات سواء رقمية أو نصية في فضاء أبعاد متعدد (ليس فقط ثنائي كما رأينا سابقا) .

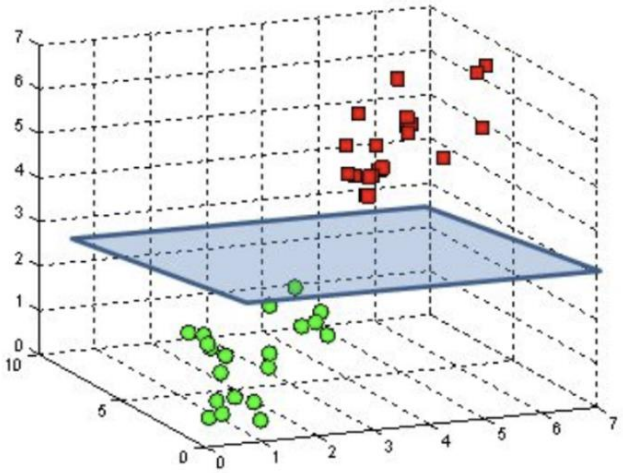
2- Finding a separator

تخمين الفاصل المناسب بين البيانات وذلك من خلال تحويل نمط البيانات أو تقييسها لنتمكن من رسم أفضل فاصل بينها سواء مستوي أو متعرج hyperplane .

مثلا إذا قمنا بتوزيع بيانات الجدول السابق بالاعتماد على سميتين هما unitaize,clump نجد أنه لا يمكن فصلها عن بعضها خطيا وإنما بشكل متعرج .



فيكون الفصل بين البيانات بعد ترميزها في فضاء ثلاثي أبعاد :



الآن ننتقل لطريقة ترميز البيانات في فضاء ثلاثي الأبعاد .

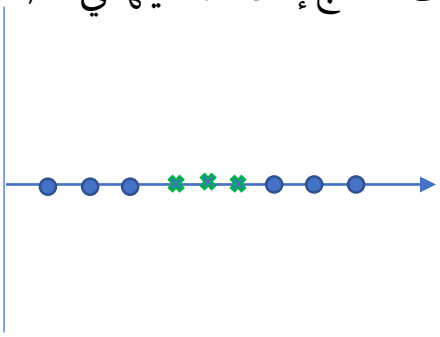
وهنا يخطر ببالنا سؤالين :

- كيف نحول البيانات التي لدينا بطريقة ما حتى يتم فصلها بمستوي ؟
- كيف نجد المستوي الأفضل للفصل بين البيانات بعد عملية التحويل ؟

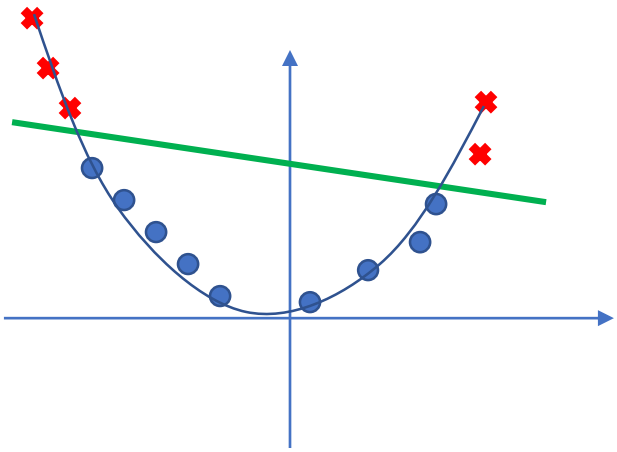
نبدأ بعملية تحويل البيانات transforming data :

لأجل التبسيط نفرض أن البيانات التي لدينا أحادية البعد أي لدينا صفة واحدة للبيانات :

نلاحظ أنه لا يمكننا فصلها بشكل خطي لذلك نحتاج إعادة تشكيلها في فضاء ثنائي الأبعاد



يتم ذلك بزيادة أبعاد البيانات و إعادة رسم x باستخدام تابع له خرجين $\emptyset(x) = [x, x^2]$ عندها يتم فصل البيانات خطيا :



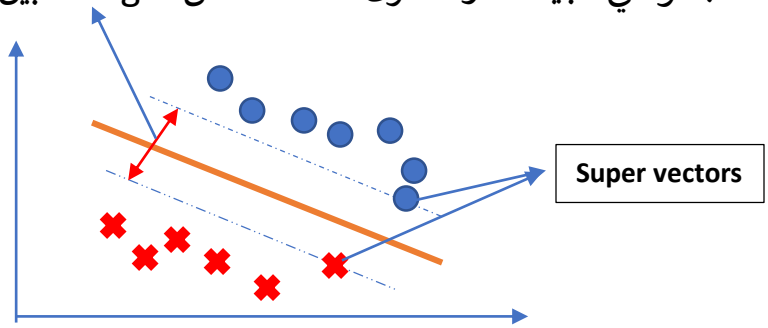
بشكل أساسي تدعى عملية تحويل البيانات لأبعاد أعلى باسم kernelling و يسمى التابع المستخدم لذلك kernel function ويوجد لها عدة أنواع :

Linear , Polynomial , Radial Basic Function RBF , Sigmoid

طبعا هذه كلها مضمنة في مكتبات جاهزة في البايثون وبالتالي لتحديد أي منها نستخدم يكون بتجربة أكثر من نمط واختيار الأفضل من ناحية النتائج و الدقة .

ننتقل للخطوة التالية : كيف نجد المستوي الأفضل بعد التحويل :

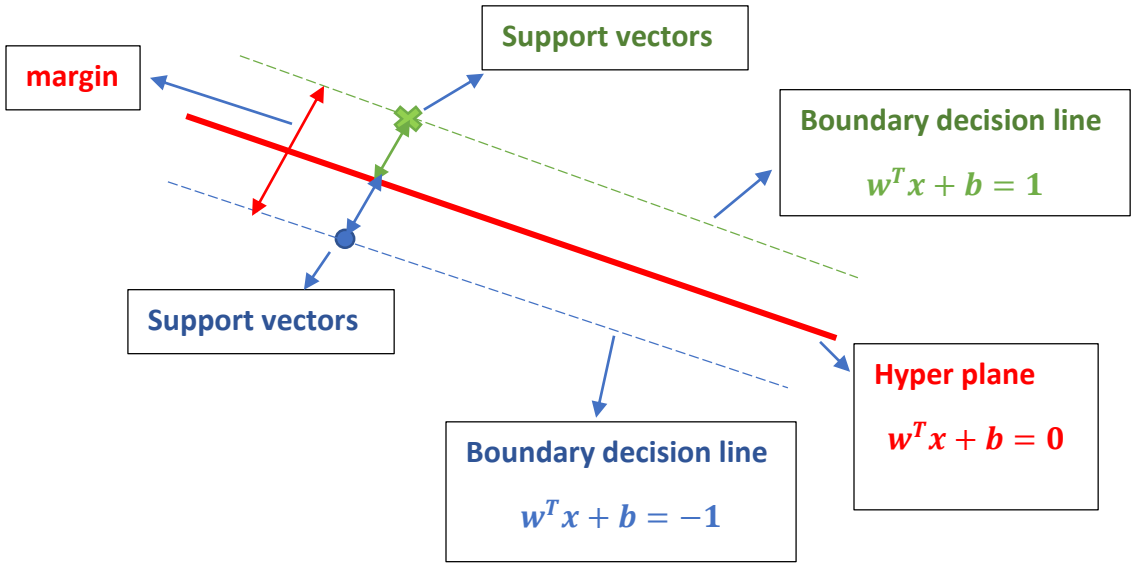
بشكل عام أهم نقطة هو أن يكون المستوي فاصل بين مجموعتي البيانات وكما نرى أن هناك مجموعتي البيانات وكما نرى هناك هامش margin بين مجموعتي البيانات كما يبين الشكل:



فالهدف إيجاد hyperplane يملك أكبر هامش بين الصنفين و تدعى النقاط الأقرب من كلا الصنفين للخط والمستوي الفاصل بـ support vectors فهدفنا الوحيد هو هذه النقاط وإهمال بقية نقاط التدريب .

إذا سنجد الفاصل الذي يملك أكبر هامش بين النقاط المحاذية support vectors .

وسنلاحظ أن لكل من الخط الفاصل والهامشين أو الحدين boundary معادلته الخاصة .



إذا عن طريق بيانات التدريب نحاول جعل الفاصل hyperplane يملك أكبر هامش ويمكن أمثَلة هذه العملية باستخدام Gradient Descent GD .

إذا خرج هذه الخوارزمية هي قيمة w,b لإيجاد معادلة المستقيم و تحديد فيما إذا النقاط المعطاة هي تحت هذا الخط أو فوقه .

فعندما تحدد معادلة المستقيم قيمة أكبر من الصفر فالنقطة تنتمي للصنف الأول أي فوق الخط و عندما تعطي المعادلة قيمة تحت الصفر فالنقطة تنتمي للصنف الثاني أي تحت الخط

Find w and b such that :

$$\emptyset(w) = \frac{1}{2}\, w^T w \; is \; minimized :$$

$$and \; for \; all \; \{(x_i, y_i)\} : \; y_i(w^T x_i + b) \geq 1$$

محاسن و مساوئ SVM :

محاسنها Advantages :

1- الدقة في فضاءات عالية الأبعاد Accurate in high-dimensional spaces
2- Memory efficient توفر الذاكرة لأنها تستخدم جزء من بيانات التدريب في تحديد تابع القرار أو معادلة الخط الفاصل .

مساوئها Disadvantages :

1- هذه الخوارزمية معرضة لحالة over-fitting وذلك حين تكون عدد السمات number of features أكبر من عدد العينات number of sample .
2- لا تعطي احتمالية التوقع التي تعطيها أغلب طرق التصنيف no probability estimation .
3- تستعمل لحالات البيانات القليلة small datasets فهي ليست ذو كفاءة حسابية عندما تكون البيانات ضخمة مثلا أكثر من ألف سطر .

الحالات التي تستخدم فيها SVM :

- تمييز الصور و Image recognition and handwriting digit recognition . خط اليد
- تحديد نوع الخط والبحث Text category assignment : text-mining tasks . في النص
- اكتشاف ملفات التجسس Detecting spam
- تحليل المشاعر. Sentiment analysis
- تصنيف الجينات Gene expression classification
- التوقع. Regression
- اكتشاف الحالات الشاذة Outlier detection
- تجميع الأصناف المتشابهة Clustering

سنقوم بتنفيذ خوارزمية SVM ضمن مثال باستخدام البايثون :

نبدأ باستيراد المكتبات التي نستعملها :

[1]:
```
import pandas as pd
import pylab as pl
import numpy as np
import scipy.optimize as opt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
%matplotlib inline
import matplotlib.pyplot as plt
```

وبعد تحميل الملف cell_samples.csv كما نفعل سابقا نحدد أول خمس مرض للتسهيل :

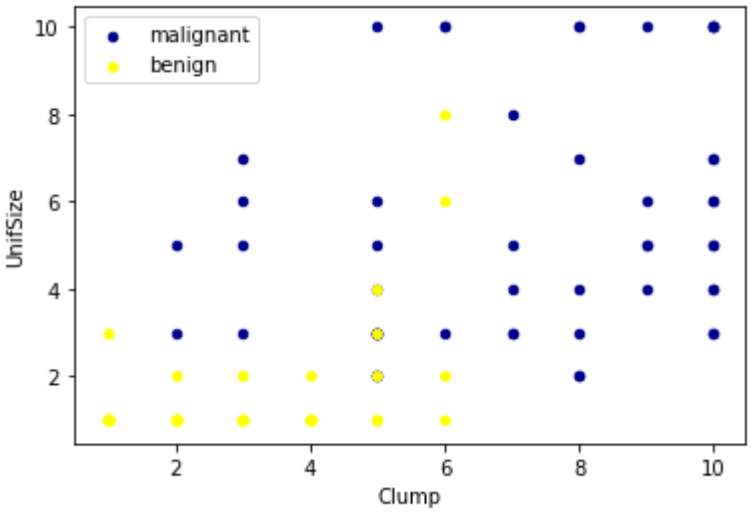
[3]:
```
cell_df = pd.read_csv("cell_samples.csv")
cell_df.head()
```

[3]:

	ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

ثم نرسم توزع المرضى بحسب الخاصيتين Clump,Unitsize :

```
[4]: ax = cell_df[cell_df['Class'] == 4][0:50].plot(kind='scatter', x='Clump', y='UnifSize', color='DarkBlue', label='malignant');
cell_df[cell_df['Class'] == 2][0:50].plot(kind='scatter', x='Clump', y='UnifSize', color='Yellow', label='benign', ax=ax);
plt.show()
```



وقبل البدء في التقسيم نحدد ماهية (نوعية) البيانات رقمية أم نصية :

```
[5]: cell_df.dtypes
```

```
[5]: ID             int64
     Clump          int64
     UnifSize       int64
     UnifShape      int64
     MargAdh        int64
     SingEpiSize    int64
     BareNuc       object
     BlandChrom     int64
     NormNucl       int64
     Mit            int64
     Class          int64
     dtype: object
```

نلاحظ وجود بعض الأسطر نصية categorical نهملها :

```
[6]: cell_df = cell_df[pd.to_numeric(cell_df['BareNuc'], errors='coerce').notnull()]
     cell_df['BareNuc'] = cell_df['BareNuc'].astype('int')
     cell_df.dtypes
```

```
[6]: ID             int64
     Clump          int64
     UnifSize       int64
     UnifShape      int64
     MargAdh        int64
     SingEpiSize    int64
     BareNuc        int64
     BlandChrom     int64
     NormNucl       int64
     Mit            int64
     Class          int64
     dtype: object
```

نحدد الآن مصفوفة البيانات التي سنعمل عليها :

```
[7]: feature_df = cell_df[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize', 'BareNuc', 'BlandChrom', 'NormNucl', 'Mit']]
     X = np.asarray(feature_df)
     X[0:5]
```

```
[7]: array([[ 5,  1,  1,  1,  2,  1,  3,  1,  1],
            [ 5,  4,  4,  5,  7, 10,  3,  2,  1],
            [ 3,  1,  1,  1,  2,  2,  3,  1,  1],
            [ 6,  8,  8,  1,  3,  4,  3,  7,  1],
            [ 4,  1,  1,  3,  2,  1,  3,  1,  1]])
```

كما رأينا في الأصناف إما سليمة أو خبيثة ونحن فقط نريد أن نتوقع الحالات التي صنفها 2 أي السليمة :

```
[8]: cell_df['Class'] = cell_df['Class'].astype('int')
     y = np.asarray(cell_df['Class'])
     y [0:5]
```

```
[8]: array([2, 2, 2, 2, 2])
```

ونقوم بتقسيم البيانات لتدريب و اختبار :

```
[9]: X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
     print ('Train set:', X_train.shape,  y_train.shape)
     print ('Test set:', X_test.shape,  y_test.shape)

     Train set: (546, 9) (546,)
     Test set: (137, 9) (137,)
```

ندرب المودل باختيار المنحني rbf radial basis function :

```
[13]: from sklearn import svm
      clf = svm.SVC(kernel='rbf',gamma='auto')
      clf.fit(X_train, y_train)

[13]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
        decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
        max_iter=-1, probability=False, random_state=None, shrinking=True,
        tol=0.001, verbose=False)
```

نجري اختبار للنموذج بعد التدريب :

```
[12]: yhat = clf.predict(X_test)
      yhat [0:5]

[12]: array([2, 4, 2, 4, 2])
```

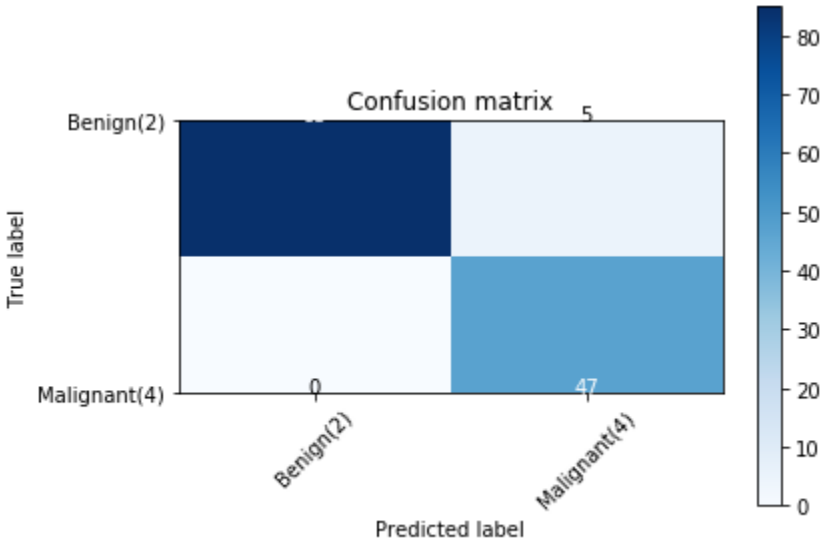
الآن نجري التقييم للنموذج باستخدام confusion matrix cm :

[22]:
```
from sklearn.metrics import classification_report, confusion_matrix
import itertools
```

[23]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

[24]:
```
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, yhat, labels=[2,4])
np.set_printoptions(precision=2)

print (classification_report(y_test, yhat))

# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=['Benign(2)','Malignant(4)'],normalize= False,  title='Confusion matrix')
```

```
              precision    recall  f1-score   support

           2       1.00      0.94      0.97        90
           4       0.90      1.00      0.95        47

   micro avg       0.96      0.96      0.96       137
   macro avg       0.95      0.97      0.96       137
weighted avg       0.97      0.96      0.96       137

Confusion matrix, without normalization
[[85  5]
 [ 0 47]]
```



ويمكننا استخدام طريقة التقييم f1-score :

[17]:
```
from sklearn.metrics import f1_score
f1_score(y_test, yhat, average='weighted')
```

[17]:
```
0.9639038982104676
```

وكذلك استخدام jaccard index :

```
[18]: from sklearn.metrics import jaccard_similarity_score
      jaccard_similarity_score(y_test, yhat)
```

```
[18]: 0.9635036496350365
```

ولإجراء مقارنة بسيطة لنختار kernel خطي linear ونقارن بالدقة :

```
[28]: # write your code here
      clf2 = svm.SVC(kernel='linear')
      clf2.fit(X_train, y_train)
      yhat2 = clf2.predict(X_test)
      print("Avg F1-score: %.4f" % f1_score(y_test, yhat2, average='weighted'))
      print("Jaccard score: %.4f" % jaccard_similarity_score(y_test, yhat2))
```

```
Avg F1-score: 0.9639
Jaccard score: 0.9635
```

للتذكرة اختيارات kernel :

1.Linear
2.Polynomial
3.Radial basis function (RBF)
4.Sigmoid

15- مقدمة للتجميع Intro Clustering :

بفرض لدينا مجموعة بيانات زبائن و نريد تطبيق شرائح لهذه الزبائن customer segmentation أي تجميع الزبائن ذوو الصفات المتشابهة ضمن مجموعات و هي استراتيجية هامة significant strategy حيث تمكن أصحاب العمل من ترويج منتجاتهم للفئة المناسبة .

مثلا مجموعة ما فيها زبائن حيث الأرباح عالية و المخاطر قليلة high-profit and low-risk وهذا مناسب جدا لعرض منتجات purchase products أو للاشتراك بخدمة subscribe for a service مما يجعل رجال الأعمال يكرسون devote وقتا إضافيا للحفاظ retaining على هؤلاء الزبائن .

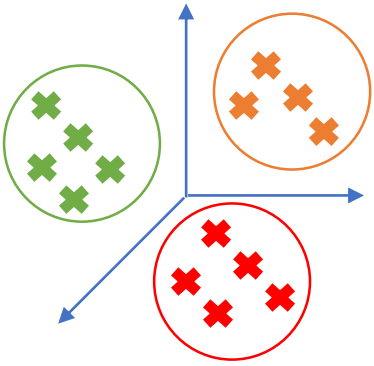
فالتجميع العام ليس ملائما feasible لكميات هائلة من البيانات المتنوعة لذلك نحتاج لطريقة تحليلية لاستخراج deriving فئات و مجموعات من البيانات الضخمة .

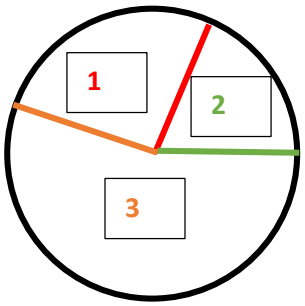
يمكننا تجميع الزبائن بالاعتماد على عدة عوامل factors مثلا لدينا بيانات الزبائن التالية :

customer ID	age	edu	years employ	income	.....	.....	defaulted
1	22	2	3	190	...	...	1
2	34	1	20	100	...	....	0
...	...	...	...	...	...	...	...
...	...	...	...	...	...	.....	...
9	50	1	5	60	...	...	1

سنعمل بهذه البيانات قدر الإمكان لتحديد أي الزبائن تشبه بعضها الآخر و من إحدى هذه الطرق المعتمدة adopted approach هي التجميع clustering .

وهي من طرق التعليم الغير إشرافي unsupervised التي تعتمد على التشابه بين صفات الزبائن ، حيث يتم تقسيم الزبائن إلى مجموعات محصورة mutually exclusive groups تسمى clusters كما نرى هنا مثلا ثلاثة أقسام ، ثم يمكننا إنشاء تعريف لكل قطاع معتبرين الصفات العامة لكل قطاع على حدا .





نرى من المخطط ثلاث قطاعات :

CLUSTER	SEGMANT NAME
cluster-1	Affluent & Middle aged
cluster-2	Young education & Middle income
cluster-3	Young & Low income

بحيث كل قطاع يشكل نسبة مئوية من المجموع الكلي وبالتالي باعتماد هذا التقسيم يمكننا إرجاع كل زبون في جدول البيانات السابق إلى قطاعه الذب ينتمي إليه :

Customer ID	Segment
1	Young & Low income
2	Affluent & Middle aged
…	….
9	Young education & Middle income

وبهذا يمكننا استهداف أو توجيه إعلانات محددة أو بضاعة معينة تناسب الزبائن من خلال التقسيمات التي ينتموا إليها .

فتعريف القطاع cluster هو مجموعة المكونات المتشابهة similar بما بينها و لا تشبه النقاط (البيانات) الموجودة في قطاعات أخرى وهنا يتبادر للذهن ما الفرق بين التجميع clustering و التصنيف classification ؟

يعتبر التصنيف من نوع التعليم الإشرافي supervised حيث كل وحدة من بيانات التدريب تنتمي لصنف معين وندرب النموذج (مثلا باستخدام Decision tree) لإيجاد الصنف لعنصر جديد وتكون البيانات labeled أي معنونة .

أما في التجميع clustering فهو من التعليم الغير إشرافي unsupervised والبيانات لدينا تكون unlabeled ونستخدم مثلا (K-means) لنجمع البيانات المتشابهة فيما بينها ثم نعيدهم للجدول لفرز العناصر كلها .

ولبيان الفرق أكثر بين الطريقتين لدينا مجالات و طرق الفرز وهي :

- Retail / Marketing : Identifying buying patterns of customers البيع بالتقسيط

- Recommending new books/movies to new customers الترويج لمنتجات جديدة لزبائن جدد

- Banking : Fraud detection in credit card use Identifying clusters of customers كشف الاحتيال و تصنيف العملاء

- Insurance industry : Fraud detection in claim analysis and Insurance risk of customers كشف الاحتيال في قطاع التأمين من قبل الزبائن

- Publication media : Auto-categorizing news based on their content and Recommending similar news articles to readers . تجميع المقالات المتشابهة وتقديم المناسب للقراء .

- Medicine : Charactering patient behavior تشخيص سلوك المرضى

- Biology : Clustering genetic markers to identify familyist التجميع الجيني لبيان الروابط الأسرية و القربى.

وبشكل عام مجالات استخدام التجميع clustering :

- Exploratory data analysis تحليل البيانات المكتشفة
- Summary generation = reducing the scale تلخيص البيانات
- Outlier detection for fraud detection or noise removed اكتشاف القيم الشاذة لأجل اكتشاف حالات الاحتيال أو إزالة الضجيج .

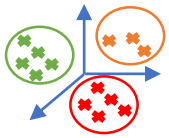
- Pre-processing step for other data-mining tasks or as apart of complex system تستخدم في ماحل ما قبل مهام التنقيب عن البيانات أو كجزء من الأنظمة المعقدة

16-1- الخوارزميات الرئيسية في التجميع :

1- Partitioned-based clustering :

ينتج عنها قطاعات من النوع sphere-like clusters أي قطاعات متكتلة وجنب بعضها الآخر ، وتستخدم هذه الطرق للبيانات الكبيرة و المتوسطة الحجم مثلا ألف سطر .

- Relatively efficient فعالة و واقعية
- Eg. K-means, K-median, Fuzzy c-means

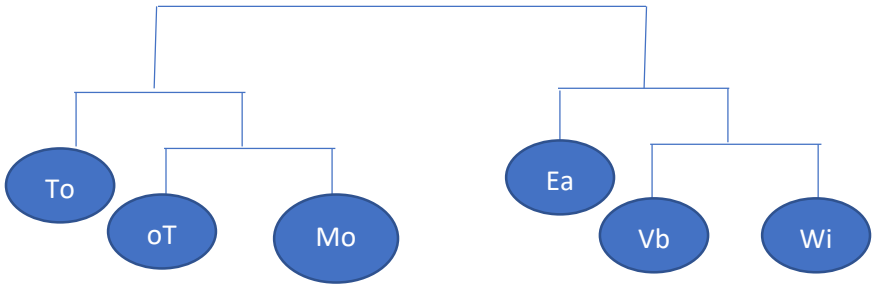


2- Hierarchical clustering :

Produces trees of clusters

ينتج عنها شجرة قطاعات وتستخدم للبيانات الصغيرة الحجم .

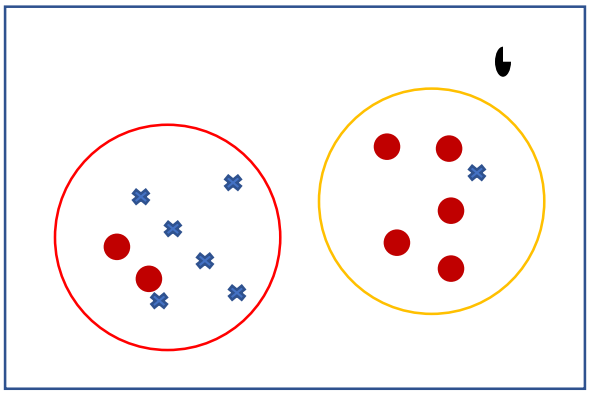
Eg . Agglomerative , Divisive



3- Density-based clustering :

Produces arbitrary shaped clusters

يتم توليد مجموعات عشوائية عند وجود ضجيج مثل DBSCAN .



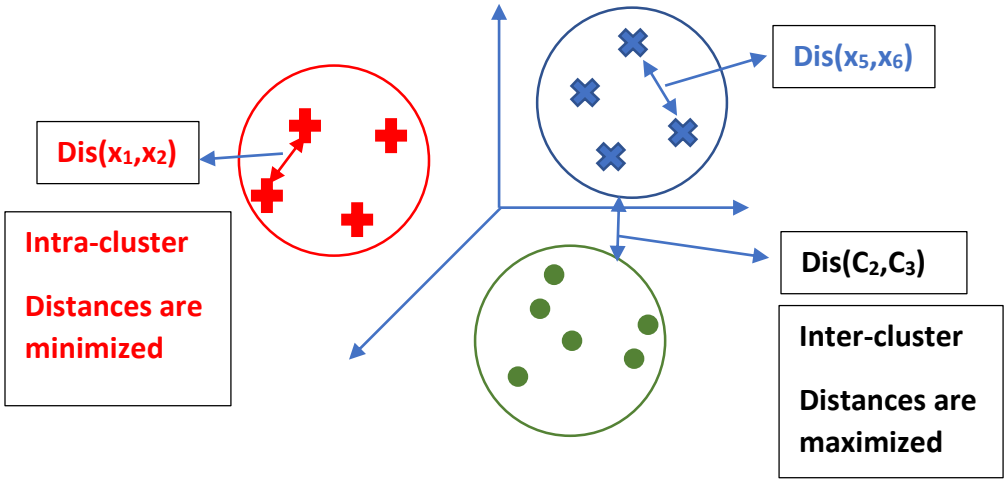
- **خوارزمية K-means clustering :**

فرضا لدينا مجموعة بيانات زبائن و نريد تنفيذ customer segment أي عملية تجزيئ على الزبائن بحيث يصبحون مجموعات بنفس الصفات و من هذه الطرق هي k-means حيث تعتبر هذه الخوارزمية من نوع partitioning أي تقسيم البيانات إلى قطاعات تسمى k non-overlapping subsets مجموعات جزئية غير متداخلة دون تكتلات داخلية cluster-internal فمكونات كل قطاع متشابهة جدا ومختلفة عن القطاعات الأخرى وهنا يتم طرح سؤالين :

1- كيف نجد العينات المتشابهة ؟

2- كيف نقيس مقدار التشابه بين عنصرين بغض النظر عن موقعهما ضمن البيانات ؟

ننوه انه بالرغم من كون k-means تجمع البيانات في قطاعات بالاعتماد على التشابه فيما بينها فإنه يمكننا استعمال مقياس التباين dissimilarity عوضا عن مقياس التشابه .



بشكل تقليدي conventionally تستعمل المسافة بين العينات لتشكيل القطاعات فطريقة k-means تحاول تصغير المسافة بين العناصر في كل قطاع على حدا intra-cluster وتكبير المسافة بين كل قطاع وآخر inter-cluster .

فكيف نقيس الاختلاف أو المسافة بين حالتين (زبونين مثلا) ؟

customer1	customer2	
age	age	
54	50	$Dis = \sqrt{(54-50)^2} = 4$
income	income	
190	200	$Dis = \sqrt{(54-50)^2 + (190-200)^2} = 10.77$
education	education	
3	8	$Dis = \sqrt{(54-50)^2 + (190-200)^2 + (3-8)^2} = 11.87$

لقد مرت معنا سابقا ولكن نذكر أنه ينبغي أن نقوم بعملية تقييس normalize السمات للحصول على مقياس دقيق للتباين accurate dissimilarity measure .

يوجد مقاييس تباين أخرى تستخدم لنفس الهدف ولكنها تعتمد على نوع البيانات بشكل كبير و على المجال الذي ننفذ عليه عملية الفرز فمثلا نستعمل :

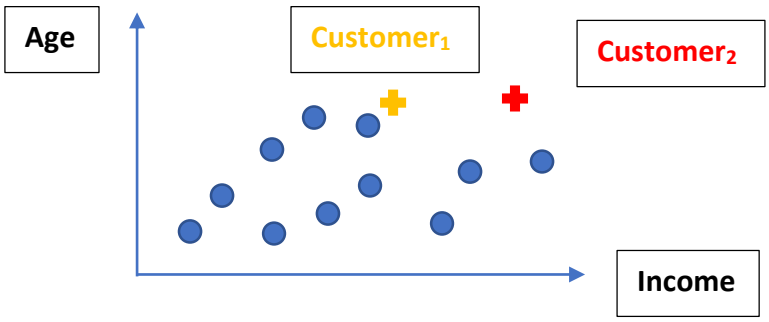
Euclidean distance , Cosine similarity , average distance

إذا مقياس التشابه مهم جدا بحسب طبيعة البيانات و الميزات لتشكيل القطاعات في عملية الفرز.

نعود الآن لفهم طريقة عمل k-means مع تبسيط البيانات :

Customer ID	Age	Income
1	32	450
2	23	300
3	44	120
....	....	....
....	....	....

نلاحظ الدخل قيمته بالمئات و العمر بالعشرات لذلك نحتاج التقييس ونلاحظ أن الفضاء ذو بعدين 2-dimentional سنرسم مخطط scatterplot للبيانات ونوزعها بمحورين :



سنحاول فرز هذه البيانات (الزبائن) في مجموعات متمايزة (قطاعات) بالاعتماد على بعدين فقط .

1- بداية ينبغي أن نحدد عدد القطاعات ففي طريقة k-means يتم فرض يشكل عشوائي مركز كل قطاع أي نفرض قيمة k وهي تحدد عدد القطاعات وهي مسألة ليست سهلة وسنناقشها فيما بعد . (Initialize k=3 centroid randomly) يتم تسمية هذه المراكز c_1,c_2,c_3 (centroid of clusters) و ينبغي أن يكون لها نفس حجم الميزات للعناصر الموجودة في المجموعة .

ويوجد طريقتين لاختيار هذه المراكز :

- أن نختارها عشوائيا 3 مراكز خارج مجموعات البيانات مثلا :

$C_1 = [8.,5.]$ $C_2 = [5.,5.]$ $C_3 = [6.,3.]$

- بعد اختيار المراكز نبدأ بفرز البيانات (الزبائن) للمركز الأقرب له عن طريق حساب المسافة بين كل عنصر و المراكز واختيار المركز الأقرب .
- إذا سنحصل على مصفوفة تسمى distance-matrix لها الشكل التالي :

$$\begin{matrix} C_1 & C_2 & C_3 \end{matrix}$$
$$\begin{bmatrix} d(p_1,c_1) & d(p_1,c_2) & d(p_1,c_3) \\ d(p_2,c_1) & d(p_2,c_2) & d(p_2,c_3) \\ \dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots \\ d(p_n,c_1) & d(p_n,c_2) & d(p_n,c_3) \end{bmatrix}$$

سيكون هدف k-means تقليل المسافة بين النقطة و المركز القطاع الذي تنتمي إليه و تكبير المسافة بين المراكز .

لن نقول أن هذه العملية دقيقة كفاية لأننا اخترنا المراكز بشكل عشوائي منذ البداية فالنموذج فيه خطأ يتم حسابه كالتالي :

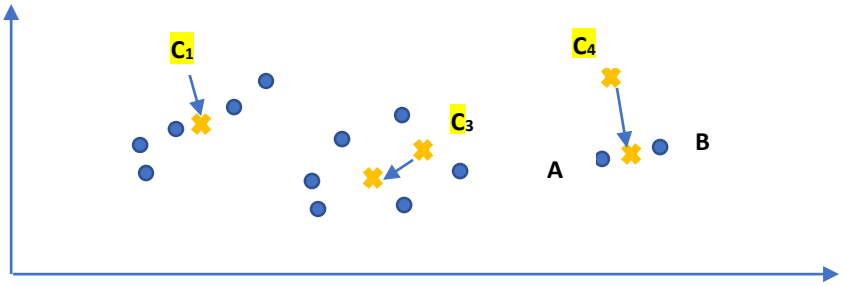
$$SSE = \sum_{i=1}^{n} (x_i - c_i)^2$$

: $the\ sum\ of\ the\ squared\ difference\ between\ each\ point\ and\ its\ centroid$

مجموع مربعات الأخطاء بين كل نقطة و مركز القطاع الذي تنتمي إليه والآن أصبحت مهمتنا تقليل هذا الخطأ .

- في الخطوة الثانية نقوم بتحريك المراكز لتكون في متوسط mean بيانات القطاع الذي ترتبط به .

فمثلا لدينا إحداثيات نقطتين A(7.4,3.6),B(7.8,3.8) فالمركز الجديد يصبح متوسطهم أي c_2((7.4+7.8)/2,(3.8+3.6)/2) = (7.6,3.7) .



ثم نعيد حساب المسافة بين نقاط كل قطاع والمركز الجديد له ثم حساب الخطأ من جديد و نعيد تقسيم القطاعات بحسب المسافات الجديدة .

نستنتج أن خوارزمية *k-means* تكرارية بحيث نكرر آخر خطوتين بتحريك المراكز و حساب المسافات حتى نحصل على أفضل تقسيم للقطاعات most dense clusters .

وعلى كل حال عندما نستخدم خوارزميات heuristic لا يوجد أي ضمان أننا سنقترب للحل الأمثل global optimum والنتيجة ستعتمد على القطاعات الأولية ولكن يمكننا اعتبار النتيجة التي نحصل عليها مناسبة local optimum فليس بالضرورة الوصول لأفضل نتيجة.

ولكن لا مشكلة في تكرار الخوارزمية كثيرا بغية الحصول على الحل الأفضل كونها خوارزمية سريعة .

سنتعرف على دقة هذه الخوارزمية وصفاتها بشكل ملموس more concretely :

1- Randomly placing k centroids ,one for each cluster

نضع المراكز عشوائيا للقطاعات وكلما تباعدت القطاعات عن بعضها كان أفضل .

2- Calculate the distance of each point from each centroids .

حساب مسافة كل نقطة عن المراكز باستخدام Euclidean distance وهي الأكثر شيوعا .

3- Assign each data point (object) to its closest centroid for creating a cluster .

نربط كل نقطة بالمركز الأقرب لها و نشكل القطاع .

4- Recalculate the position of the k centroids .

بعد تشكيل القطاعات نعيد حساب مركز كل قطاع بحيث يكون موقعه الجديد هو متوسط النقاط التي ضمن قطاعه .

5- Repeat 2→4 until the centroids no longer move .

نعيد الخطوات من 2 إلى 4 حتى لا يكون هنالك أي تحرك ملحوظ للمراكز .

السؤال : كيف نحدد جودة goodness هذه القطاعات التي نتجت من k-means أي كيف نحسب دقة هذه الطريقة ؟

يوجد نوعين لطرق حساب الدقة :

- External approach : طرق خارجية

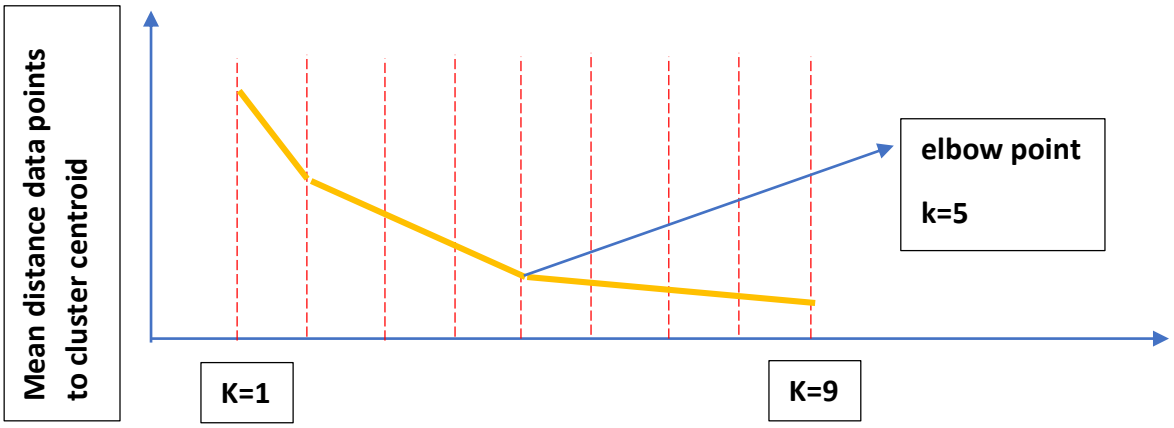
  Compare the clusters with the ground truth if it is available .

التحقق من جودة القطاعات على أرض الواقع إن أمكن ذلك ، حيث هذه الخوارزمية k-means غير إشرافية unsupervised وهذه الطرق غير واقعية لعدم وجود حقائق نستطيع مقارنتها فيها .

- Internal approach : طرق داخلية

  Average the distance between data points with a cluster .

نحسب قيمة متوسط المسافة بين النقاط في القطاع ، كذلك يعتبر متوسط المسافات للنقاط عن مراكزهم في القطاع مقياس للخطأ .



يبقى تحديد عدد k أمر فيه لبس ambiguous لأنه متعلق بشكل و توزع البيانات وهنالك طرق لتحديدها لكن أعم هذه الطرق أن نحدد قيمة k بداية عشوائيا ثم نقيس الدقة ونكرر العملية عدة مرات .
المشكلة أنه في زيادة عدد القطاعات أي زيادة قيمة k فإن المسافة بين المراكز و النقاط ستنخفض وبالتالي زيادة k يقلل الخطأ لذلك عندما نرسم دقة النموذج فإن نقطة التحول elbow point أي نقطة الانعطاف تتحدد عند النقطة التي يبدأ عندها متوسط المسافات بين النقاط و المراكز بالانخفاض وهذه الطريقة تسمى elbow .
لنلخص ما سبق :

k-means is a partitioned-based clustering which is :

a- Relative efficient on Med&Large sized dataset .
b- Produces sphere-like clusters ,because the clusters are shaped around the centroids .
c- Its drawback is that we should pre-specify the number of clusters and this is not easy task .

سنأخذ مثاليين عن k-means وهما :

- K-means on a random generated dataset
- Using k-means for customer segmentation

نبدأ بالمثال الأول :

Import libraries

Lets first import the required libraries. Also run **%matplotlib inline** since we will be plotting in this section.

[1]:
```
import random
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets.samples_generator import make_blobs
%matplotlib inline
```

سنشكل مجموعة بيانات عشوائية ونحدد مراكزها باستخدام k-means :

Lets create our own dataset for this lab!

First we need to set up a random seed. Use **numpy's random.seed()** function, where the seed will be set to **0**

[2]:
```
np.random.seed(0)
```

Next we will be making *random clusters* of points by using the **make_blobs** class. The **make_blobs** class can take in many inputs, but we will be using these specific ones.

Input

- **n_samples**: The total number of points equally divided among clusters.
  - Value will be: 5000
- **centers**: The number of centers to generate, or the fixed center locations.
  - Value will be: [[4, 4], [-2, -1], [2, -3],[1,1]]
- **cluster_std**: The standard deviation of the clusters.
  - Value will be: 0.9

Output

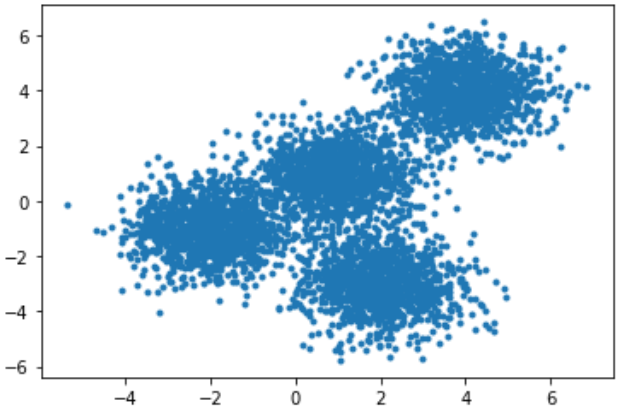
- **X**: Array of shape [n_samples, n_features]. (Feature Matrix)
  - The generated samples.
- **y**: Array of shape [n_samples]. (Response Vector)
  - The integer labels for cluster membership of each sample.

```
[3]: X, y = make_blobs(n_samples=5000, centers=[[4,4], [-2, -1], [2, -3], [1, 1]], cluster_std=0.9)
```

Display the scatter plot of the randomly generated data.

```
[4]: plt.scatter(X[:, 0], X[:, 1], marker='.')
```

```
[4]: <matplotlib.collections.PathCollection at 0x7fecac06fdd8>
```



إلى هذه المرحلة نكون قمنا بتشكيل مجموعات البيانات العشوائية وسنجري الآن إعدادات k-means :

بتحديد عدد المراكز و القطاعات وطريقة إيجاد المراكز المناسبة بعد تدريب النموذج :

Setting up K-Means

Now that we have our random data, let's set up our K-Means Clustering.

The KMeans class has many parameters that can be used, but we will be using these three:

- **init**: Initialization method of the centroids.
  - Value will be: "k-means++"
  - k-means++: Selects initial cluster centers for k-mean clustering in a smart way to speed up convergence.
- **n_clusters**: The number of clusters to form as well as the number of centroids to generate.
  - Value will be: 4 (since we have 4 centers)
- **n_init**: Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.
  - Value will be: 12

Initialize KMeans with these parameters, where the output parameter is called **k_means**.

Initialize KMeans with these parameters, where the output parameter is called **k_means**.

[5]:
```
k_means = KMeans(init = "k-means++", n_clusters = 4, n_init = 12)
```

Now let's fit the KMeans model with the feature matrix we created above, **X**

[6]:
```
k_means.fit(X)
```

[6]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=12, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

بعد حساب المراكز نحتاج لرسم البيانات :

Now let's grab the labels for each point in the model using KMeans' **.labels_** attribute and save it as **k_means_labels**

[7]:
```
k_means_labels = k_means.labels_
k_means_labels
```

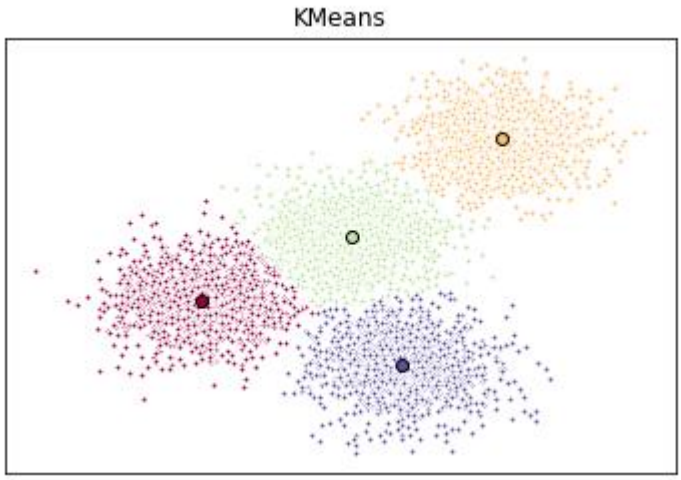
[7]:
```
array([0, 3, 3, ..., 1, 0, 0], dtype=int32)
```

We will also get the coordinates of the cluster centers using KMeans' **.cluster_centers_** and save it as **k_means_cluster_centers**

[8]:
```
k_means_cluster_centers = k_means.cluster_centers_
k_means_cluster_centers
```

[8]:
```
array([[-2.03743147, -0.99782524],
       [ 3.97334234,  3.98758687],
       [ 0.96900523,  0.98370298],
       [ 1.99741008, -3.01666822]])
```

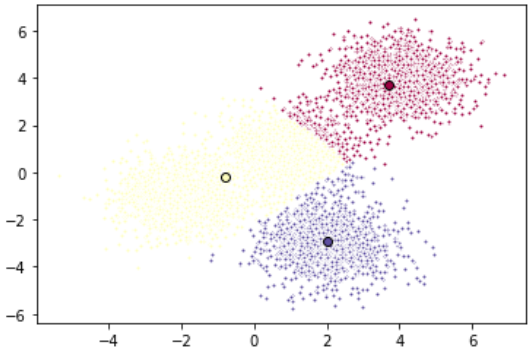
[9]:
```
# Initialize the plot with the specified dimensions.
fig = plt.figure(figsize=(6, 4))

# Colors uses a color map, which will produce an array of colors based on
# the number of labels there are. We use set(k_means_labels) to get the
# unique labels.
colors = plt.cm.Spectral(np.linspace(0, 1, len(set(k_means_labels))))

# Create a plot
ax = fig.add_subplot(1, 1, 1)

# For loop that plots the data points and centroids.
# k will range from 0-3, which will match the possible clusters that each
# data point is in.
for k, col in zip(range(len([[4,4], [-2, -1], [2, -3], [1, 1]])), colors):

    # Create a list of all data points, where the data poitns that are
    # in the cluster (ex. cluster 0) are labeled as true, else they are
    # labeled as false.
    my_members = (k_means_labels == k)

    # Define the centroid, or cluster center.
    cluster_center = k_means_cluster_centers[k]

    # Plots the datapoints with color col.
    ax.plot(X[my_members, 0], X[my_members, 1], 'w', markerfacecolor=col, marker='.')

    # Plots the centroids with specified color, but with a darker outline
    ax.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col,  markeredgecolor='k', markersize=6)

# Title of the plot
ax.set_title('KMeans')

# Remove x-axis ticks
ax.set_xticks(())

# Remove y-axis ticks
ax.set_yticks(())

# Show the plot
plt.show()
```



نلاحظ كيف تم تقسيمها ضمن أربع قطاعات .

فلو أردنا تقسيمها ضمن ثلاث قطاعات دون إعادة توليد البيانات من جديد :

[10]:
```
# write your code here
k_means3 = KMeans(init = "k-means++", n_clusters = 3, n_init = 12)
k_means3.fit(X)
fig = plt.figure(figsize=(6, 4))
colors = plt.cm.Spectral(np.linspace(0, 1, len(set(k_means3.labels_))))
ax = fig.add_subplot(1, 1, 1)
for k, col in zip(range(len(k_means3.cluster_centers_)), colors):
    my_members = (k_means3.labels_ == k)
    cluster_center = k_means3.cluster_centers_[k]
    ax.plot(X[my_members, 0], X[my_members, 1], 'w', markerfacecolor=col, marker='.')
    ax.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col,  markeredgecolor='k', markersize=6)
plt.show()
```



نأتي الآن للمثال الثاني حول استخدام k-means في تقسيم الزبائن لمجموعات :

نحمل البيانات ونظهرها :

[13]:
```
import pandas as pd
cust_df = pd.read_csv("Cust_Segmentation.csv")
cust_df.head()
```

[13]:

	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Defaulted	Address	DebtIncomeRatio
0	1	41	2	6	19	0.124	1.073	0.0	NBA001	6.3
1	2	47	1	26	100	4.582	8.218	0.0	NBA021	12.8
2	3	33	2	10	57	6.111	5.802	1.0	NBA013	20.9
3	4	29	2	4	19	0.681	0.516	0.0	NBA009	6.3
4	5	47	1	31	253	9.308	8.908	0.0	NBA008	7.2

العنوان Address نصي ولدينا خوارزمية k-means لن تتعامل مباشرة مع المعلومات النصية لأن طريقة Euclidean distance في حساب المسافات عن المراكز لا تتعامل مع البيانات المتقطعة لذلك نزيله :

[14]:
```
df = cust_df.drop('Address', axis=1)
df.head()
```

[14]:

	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Defaulted	DebtIncomeRatio
0	1	41	2	6	19	0.124	1.073	0.0	6.3
1	2	47	1	26	100	4.582	8.218	0.0	12.8
2	3	33	2	10	57	6.111	5.802	1.0	20.9
3	4	29	2	4	19	0.681	0.516	0.0	6.3
4	5	47	1	31	253	9.308	8.908	0.0	7.2

كما نرى توزيع بياناتنا ليس متناسقا فمنها بفئة العشرات ومنها آحاد ومنها أجزاء العشرات لذلك نجري تقييسها بمساعدة المكتبات الإحصائية :

[15]:
```
from sklearn.preprocessing import StandardScaler
X = df.values[:,1:]
X = np.nan_to_num(X)
Clus_dataSet = StandardScaler().fit_transform(X)
Clus_dataSet
```

[15]:
```
array([[ 0.74291541,  0.31212243, -0.37878978, ..., -0.59048916,
        -0.52379654, -0.57652509],
       [ 1.48949049, -0.76634938,  2.5737211 , ...,  1.51296181,
        -0.52379654,  0.39138677],
       [-0.25251804,  0.31212243,  0.2117124 , ...,  0.80170393,
         1.90913822,  1.59755385],
       ...,
       [-1.24795149,  2.46906604, -1.26454304, ...,  0.03863257,
         1.90913822,  3.45892281],
       [-0.37694723, -0.76634938,  0.50696349, ..., -0.70147601,
        -0.52379654, -1.08281745],
       [ 2.1116364 , -0.76634938,  1.09746566, ...,  0.16463355,
        -0.52379654, -0.2340332 ]])
```

ندرب النموذج بعد فرض عدد القطاعات وبقية بارامترات k-means :

[16]:
```
clusterNum = 3
k_means = KMeans(init = "k-means++", n_clusters = clusterNum, n_init = 12)
k_means.fit(X)
labels = k_means.labels_
print(labels)
```

نعيد توزيع القيم للجدول :

[17]:
```
df["Clus_km"] = labels
df.head(5)
```

[17]:

	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Defaulted	DebtIncomeRatio	Clus_km
0	1	41	2	6	19	0.124	1.073	0.0	6.3	0
1	2	47	1	26	100	4.582	8.218	0.0	12.8	2
2	3	33	2	10	57	6.111	5.802	1.0	20.9	0
3	4	29	2	4	19	0.681	0.516	0.0	6.3	0
4	5	47	1	31	253	9.308	8.908	0.0	7.2	1

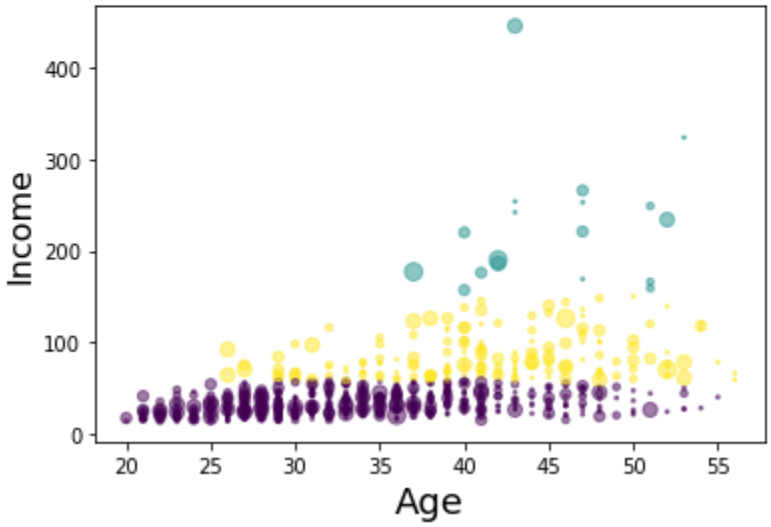
[18]:
```
df.groupby('Clus_km').mean()
```

[18]:

Clus_km	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Defaulted	DebtIncomeRatio
0	432.006154	32.967692	1.613846	6.389231	31.204615	1.032711	2.108345	0.284658	10.095385
1	410.166667	45.388889	2.666667	19.555556	227.166667	5.678444	10.907167	0.285714	7.322222
2	403.780220	41.368132	1.961538	15.252747	84.076923	3.114412	5.770352	0.172414	10.725824

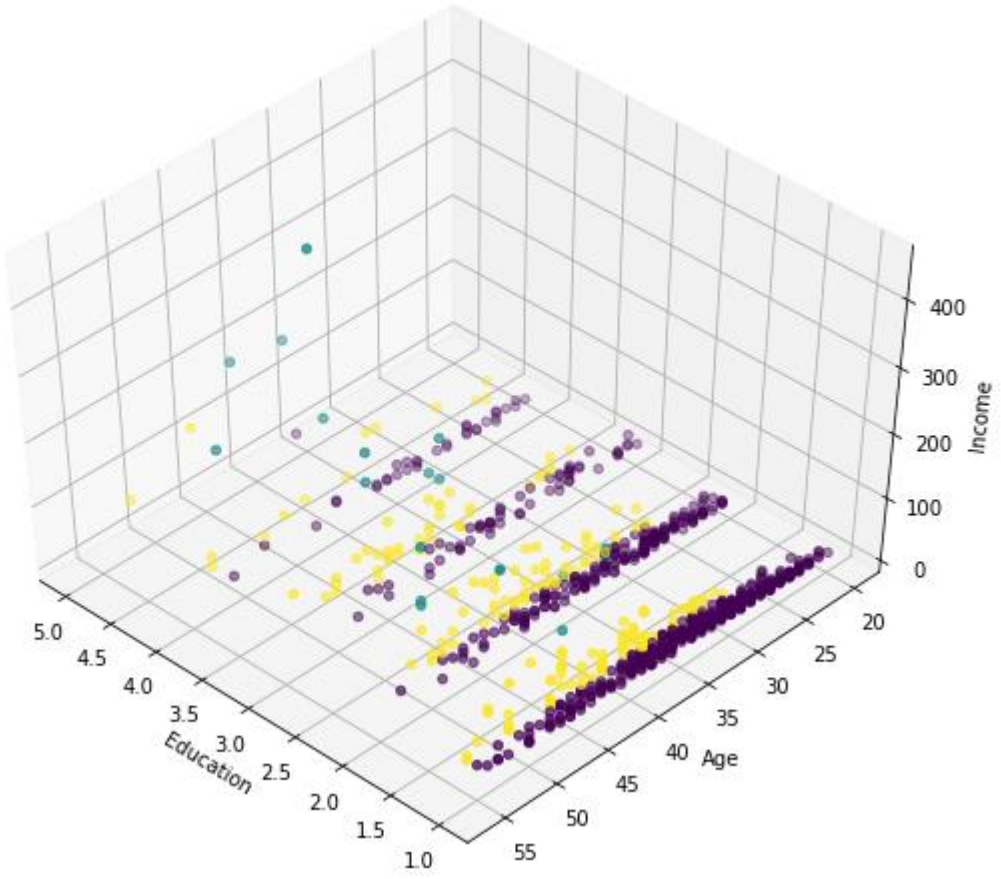
ثم نرسم البيانات :

[19]:
```
area = np.pi * ( X[:, 1])**2
plt.scatter(X[:, 0], X[:, 3], s=area, c=labels.astype(np.float), alpha=0.5)
plt.xlabel('Age', fontsize=18)
plt.ylabel('Income', fontsize=16)

plt.show()
```



وبشكل ثلاثي الأبعاد :

[20]:
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(1, figsize=(8, 6))
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)

plt.cla()
# plt.ylabel('Age', fontsize=18)
# plt.xlabel('Income', fontsize=16)
# plt.zlabel('Education', fontsize=16)
ax.set_xlabel('Education')
ax.set_ylabel('Age')
ax.set_zlabel('Income')

ax.scatter(X[:, 1], X[:, 0], X[:, 3], c= labels.astype(np.float))
```

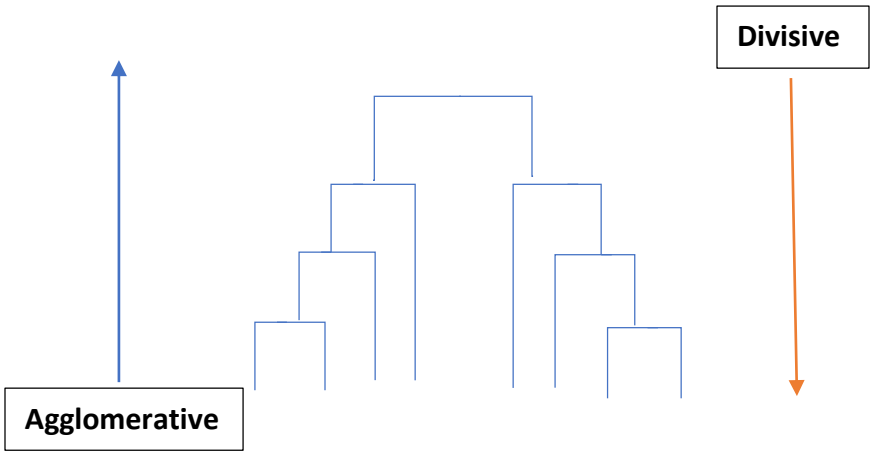


- التجميع الهرمي Hierarchical clustering :

ليكن لدينا خريطة أو مخطط chart وفريق عالمي من العلماء تابعين لمنظمة بيولوجية UCLA ويتم استخدام هذا المخطط لإعداد تقرير جيني لـ 900 حيوان الكلب منحدرة من 85 سلالة breeds وأكثر من 200 ذئب بري ضمن أمريكا الجنوبية و أوروبا و الشرق الأوسط و شرق آسيا (ربما تستغرب من هذا المثال لكنه واقع في جهة ما من الكرة الأرضية) .

يتم استخدام التقنيات الوراثية الجزئية molecular genetic techniques لتحليل أكثر من 48000 علامة جينية genetic markets ليتبين لنا مخطط يفرز هذه الحيوانات بالاعتماد على التشابه في بنياتها الجينية ، حيث يتم بناؤه بشكل هرمي و عقد وكل عقدة هي قطاع بحيث تتألف العقدة من عدة قطاعات أخرى تدعى أبنائها ويتم بناء التجميع الهرمي بالاعتماد على نوعين من الطرق :

- التقسيم Divisive له اتجاه top-down أي نبدأ بالتسلسل من الرأس للقاعدة أي نبدأ بالقسم الأكبر ثم نجزئه لأقسام أصغر منه وهكذا حتى ننتهي من التقسيم dividing the clusters .
- التجميع Agglomerative يعمل بشكل معاكس لسابقه bottom-up حيث كل قطاع يبدأ لوحده وكلما اجتمع مع قطاع آخر يشكلا قطاعا واحدا حتى نصل للقمة فهو على مبدأ التجميع أو التكديس amass وهي الأساس في علم البيانات ومحمور هذا الفصل.



لنأخذ مثالا عن التجميع حيث هذه الطريقة تبني الهرم من العناصر بإجراء الدمج التدريجي progressively merging للقطاعات .

لنقول نريد تجميع 6 مدن في كندا بالاعتماد على بعد (المسافة) كل منها عن الأخرى والمدن هي : Toronto,Ottawa,Vancouver,Motreal,Winnipeg,and Edmonton .

نسميها مصفوفة المسافات dis(i*j) :

	TO	OT	VA	MO	WI	ED
TO		351	3363	505	1510	2699
OT			3543	167	1676	2840
VA				3690	1867	819
MO					1824	2976
WI						1195
ED						

بعد حساب مصفوفة المسافات سنبدأ بتوجيه كل مدينة إلى قطاعها فعندنا 6 مدن أي 6 قطاعات وكل قطاع يحوي مدينة واحدة فقط ، والتجميع يعتمد على القرب بين المدن .

من المصفوفة نجد أن المسافة بين MO,OT هي 167 وهي أصغر مسافة بين مدينتين فنجمعهما في قطاع واحد .

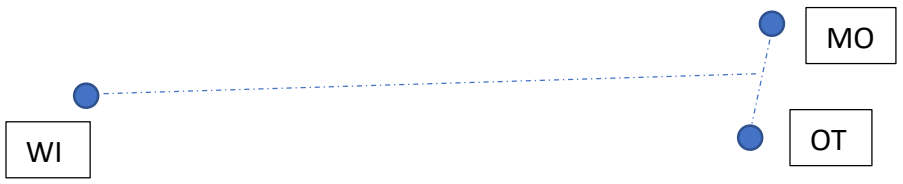
نلاحظ أنه نعتمد على ميزة feature واحدة وهي المسافة أي بعد واحد مع أنه يمكننا الاعتماد على أكثر من ميزة باستخدام خوارزميات عديدة منها Pearson,Eucluiden,Average Distance وذلك بحسب نوع البيانات ومجال المعرفة لدينا .

بعد دمج المدينتين OT,MO في قطاع واحد نعيد ترتيب المدن من جديد .

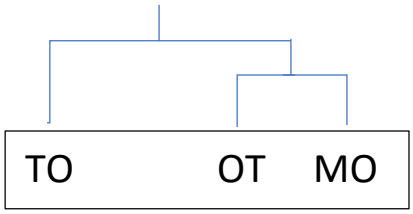
	TO	OT/MO	VA	WI	ED
TO		351	3363	1510	2699
OT/MO			3543	1676	2840
VA				1867	819
WI					1195
ED					

ولكن بعد دمج OT/MO بقطاع واحد كيف يتم حساب مسافتهما سوية عن المدن الأخر مثل WI وهي 1676 ؟

هنالك عدة طرق لذلك منها : المسافة بين مركز المسافة بين OT&MO وبين مركز WI وهكذا نحدث كافة بيانات المصفوفة :



وجدنا أن أصغر مسافة تالية بين القطاع الجديد OT/MO و TO وهي 351 وبالتالي نضعهما في قطاع واحد .



ونعيد تشكيل المصفوفة :

	TO/OT/MO	VA	WI	ED
TO/OT/MO		3363	1510	2699
VA			1867	819
WI				1195
ED				

نجد الآن أقرب مدينتين لبعضهما هما ED,VA وهي 819 ليتشكل لدينا قطاع جديد بينهما :

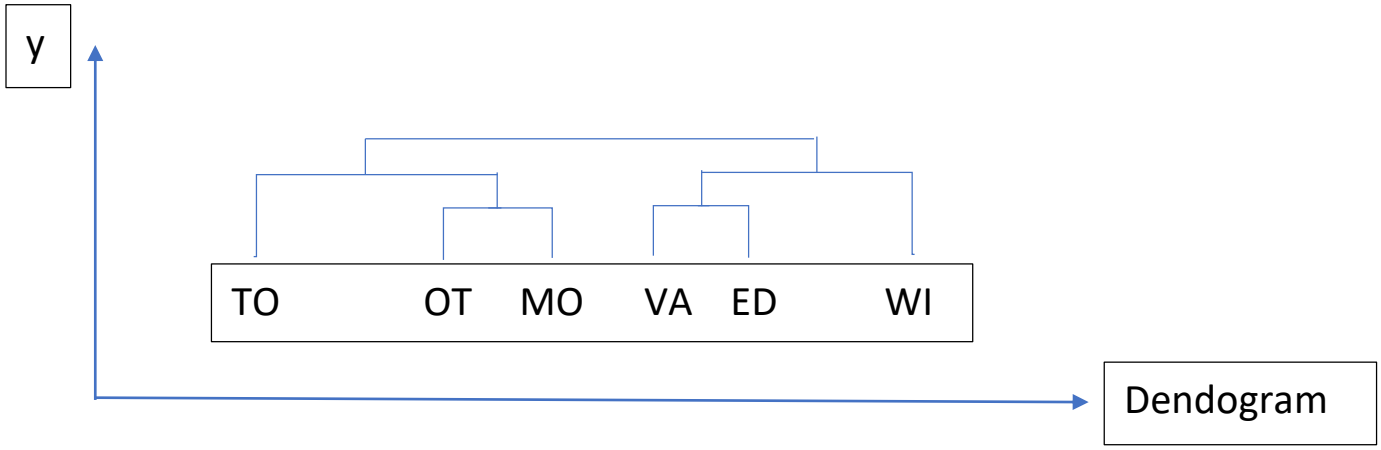
	TO/OT/MO	VA/ED	WI
TO/OT/MO		2840	1676
VA/ED			1667
WI			

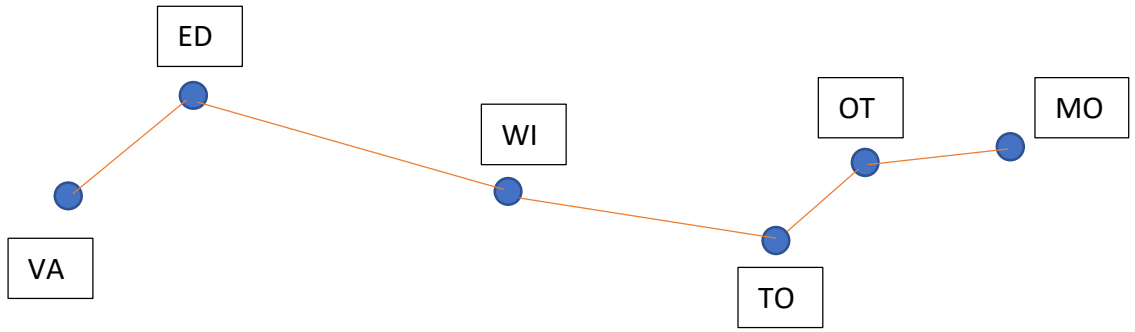
ثم نجد أقرب قطاعين هما WI و VA/ED :

	TO/OT/MO	VA/ED/WI
TO/OT/MO		1676
VA/ED/WI		

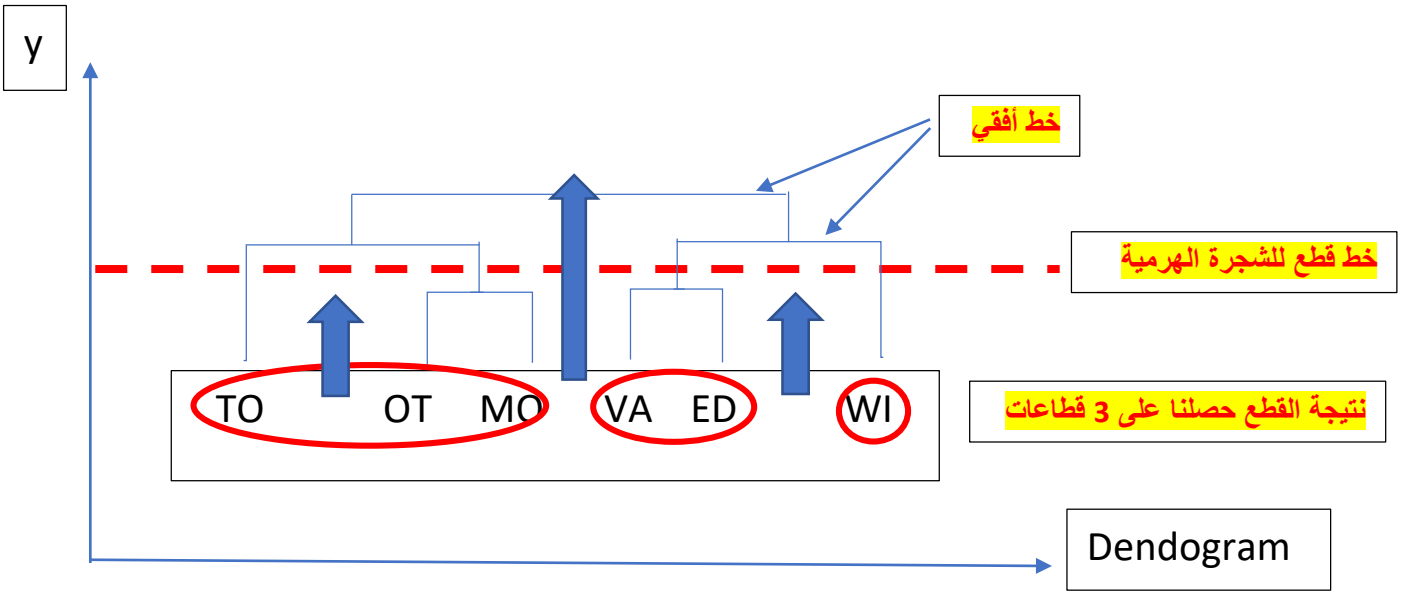
نلاحظ أنه تم دمج جميع القطاعات بالتسلسل حتى حصلنا على قطاع واحد حجمه 6 .

ويتم تمثيل الشجرة الهرمية بمخطط نسميه Dendogram :





نعود للمخطط لدراسته بتفصيل أكثر :



يتم تمثيل كل اندماج merge يحصل بين القطاعات بخط أفقي حيث تعتبر الإحداثيات y مقدار التشابه بين القطاعات التي تم دمجها سوية .

ففي مثالنا بداية تم اعتبار كل مدينة قطاع بحد ذاته وبالانتقال في المخطط من الأسف باتجاه الأعلى يتم إعادة بناء reconstruct عمليات الدمج الذي أنتج هذا التجمع العنقودي depicted clustering .

في هذه الطريقة لا يوجد افتراض أولي لعدد القطاعات و مع ذلك في بعض التطبيقات نحتاج إلى تجزئة القطاعات المنفصلة disjoint clusters كما هو متبع في التجميع المسطح flat clustering .

عند هذه الحالات يتم قطع الشجرة الهرمية عند بعض النقاط من التشابه في مستوى معين لتشكيل قطاعات متشابهة .

كما حدث في مثالنا السابق نجد عند خط القطع أصبح لدينا ثلاثة قطاعات بدل أن كانت 6 قطاعات .

إذا التسلسل في عملية التجميع من القاعدة للقمة bottom-up بفرض لدينا n نقطة في مجموعة البيانات :

1- Create n clustres ,one for each data point
2- Compute the proximity/Distance matrix : n*n
3- Repeat :
   3-1- Merge the two closest clusters
   3-2- Update the distance matrix
4- Until only a single cluster remains

وللعلم يكون شكل مصفوفة التباعد بين القطاعات :

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(n,1) & d(n,2) & .. & 0 \end{bmatrix}$$

قطرية و قطرها الرئيسي 0 لأنه يمثل بعد كل قطاع عن نفسه .

ينتج لدينا بعض التساؤلات :

- كيف يتم قياس المسافات بين هذه القطاعات وكيف نعرف الأقرب فيما بينها من هذه القطاعات ؟
- ما هي النقاط التي نستخدمها في مجموعة البيانات لدينا ؟

لنرى كيف نحسب المسافة بين قطاعين (مريضين) ولكل منهما نقطة واحدة :

	Age	BMI	BP
$Patient_1$	54	190	120
$Patient_2$	50	200	125

بطريقة Euclidean نحسب الاختلاف بين النقاط كالتالي :

$$Dis(P_1, P_2) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$$

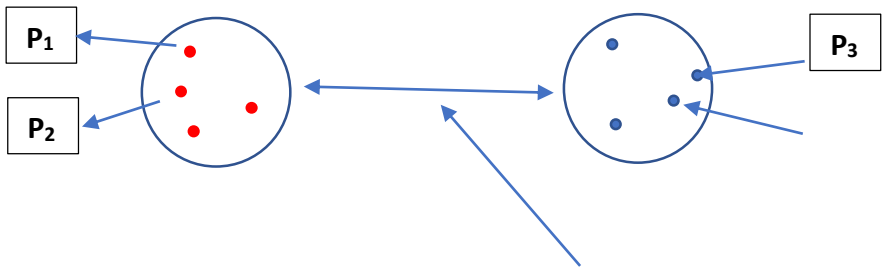
$$= \sqrt{(54-50)^2 + (190-200)^2 + (120-125)^2} = 11.87$$

فبفرض لدينا n نقطة (مريض) ينتج لدينا مصفوفة الاختلاف dissimilarly-distance :

$$\begin{matrix} P_1 \leftarrow \\ P_2 \leftarrow \end{matrix} \begin{bmatrix} x_{II} & \dots x_{IF} & \dots x_{IP} \\ \vdots & \ddots & \vdots \\ x_{iI} & \dots x_{iF} & x_{iP} \\ x_{n1} & x_{iF} & x_{nP} \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(n,1) & d(n,2) & .. & 0 \end{bmatrix}$$

Data set — Dissimilarity matrix

وكما رأينا أنه تم الدمج بين القطاعات المتقاربة بخوارزمية التجميع Agglomerative كما في الشكل التمثيلي :

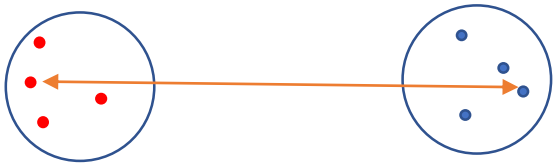


ولكن كيف يتم حساب **المسافة بين القطاعات** التي تجميع النقاط (المرضى) المتشابهة ضمنها؟

يوجد عدة تعريفات لذلك حيث نجد القطاعات المتقاربة و دمجها :

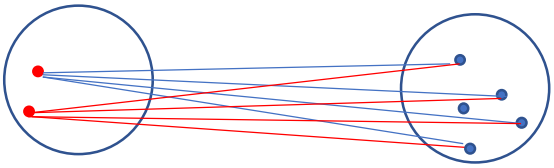
- Single-Linkage Clustering :
  Minimum distance between clusters :
  حيث نوجد أصغر مسافة بين نقطتين كل منهما في قطاع مثل (a,b)

- Complete-Linkage Clustering :
  Maximum distance between clusters :
  إيجاد أطول مسافة بين نقطتين كل منهما في قطاع :



- Average Linkage Clustering :
  Average distance between clusters :

نحسب متوسط المسافات بين كل نقطة من كل قطاع عن البقية mean distance



- Centroid Linkage Clustering :
  Distance between clusters centroids .
  نوجد المسافة بين مركزي القطاعين .

تعتمد الطريقة المتبعة لحساب هذه المسافات على نوع البيانات و أبعادها و أهم شيء مجال البيانات التي نتعامل معها .

- **سلبيات و إيجابيات طريقة التجميع الهرمي :**

Disadvantages	Advantages
لا يمكن التراجع عن أي خطوة خلال العمل حين يتم دمج القطاعات	لا تحتاج تحديد عدد معين من القطاعات
تأخذ وقتا طويلا للحسابات	سهلة التطبيق
أحيانا يصعب تحديد عدد القطاعات في المخطط عندما يكون لدينا مجموعة بيانات ضخمة	ينشئ لنا مخطط dendogram مما يسهل فهم عمل الخوارزمية

- **مقارنة بين التجميع الهرمي و k-means :**

Hierarchical	K-means
بطيئة للبيانات الضخمة	أكثر فعالية للبيانات الضخمة
لا تحتاج تحديد عدد القطاعات لتبدأ العمل	نحدد عدد القطاعات قبل البدء
تُعطي أكثر من تقسيم بالاعتماد على الدقة المطلوبة	تعطي فقط قسم واحد للبيانات بالاعتماد على عدد القطاعات المقترحة
دوما تعطي نفس عدد القطاعات	كل مرة تعطي عدد قطاعات مختلفة بحسب عدد المراكز المفترضة بشكل عشوائي حين تبدأ الخوارزمية بالعمل

نأتي لمثال مشابه لما قمنا به عند استخدام k-means بتوليد مجموعة بيانات عشوائية وتجميعها باستخدام التجميع الهرمي وتحديدا Agglomerative لأنه الأكثر شيوعا في الاستخدام :

بداية نقوم باستيراد المكتبات اللازمة :

[1]:
```
import numpy as np
import pandas as pd
from scipy import ndimage
from scipy.cluster import hierarchy
from scipy.spatial import distance_matrix
from matplotlib import pyplot as plt
from sklearn import manifold, datasets
from sklearn.cluster import AgglomerativeClustering
from sklearn.datasets.samples_generator import make_blobs
%matplotlib inline
```

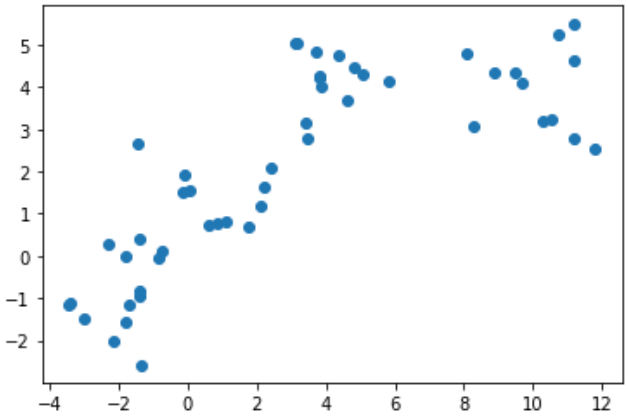
ثم نولد مجموعة بيانات عشوائية و نرسمها :

```
[2]: X1, y1 = make_blobs(n_samples=50, centers=[[4,4], [-2, -1], [1, 1], [10,4]], cluster_std=0.9)
```

Plot the scatter plot of the randomly generated data

```
[3]: plt.scatter(X1[:, 0], X1[:, 1], marker='o')
```

```
[3]: <matplotlib.collections.PathCollection at 0x7fdc7409b438>
```



نقوم باستخدام خوارزمية التجميع الهرمي وتدريب النموذج :

Save the result to a variable called **agglom**

```
[4]: agglom = AgglomerativeClustering(n_clusters = 4, linkage = 'average')
```

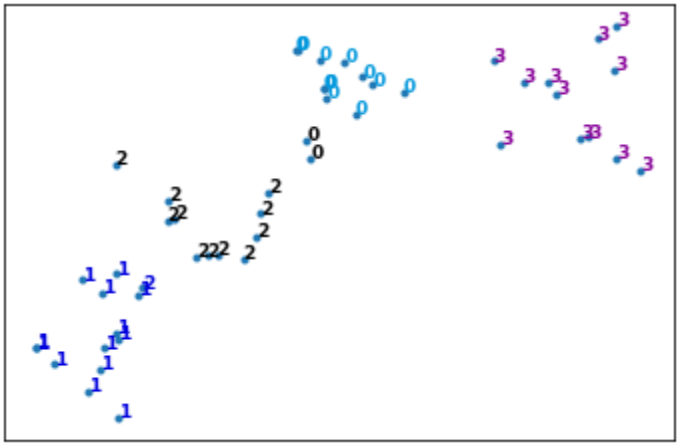
Fit the model with **X2** and **y2** from the generated data above.

```
[5]: agglom.fit(X1,y1)
```

```
[5]: AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
                             connectivity=None, linkage='average', memory=None,
                             n_clusters=4, pooling_func='deprecated')
```

ونرسم النتيجة التي حصلنا عليها :

[6]:
```
# Create a figure of size 6 inches by 4 inches.
plt.figure(figsize=(6,4))

# These two lines of code are used to scale the data points down,
# Or else the data points will be scattered very far apart.

# Create a minimum and maximum range of X1.
x_min, x_max = np.min(X1, axis=0), np.max(X1, axis=0)

# Get the average distance for X1.
X1 = (X1 - x_min) / (x_max - x_min)

# This loop displays all of the datapoints.
for i in range(X1.shape[0]):
    # Replace the data points with their respective cluster value
    # (ex. 0) and is color coded with a colormap (plt.cm.spectral)
    plt.text(X1[i, 0], X1[i, 1], str(y1[i]),
             color=plt.cm.nipy_spectral(agglom.labels_[i] / 10.),
             fontdict={'weight': 'bold', 'size': 9})

# Remove the x ticks, y ticks, x and y axis
plt.xticks([])
plt.yticks([])
#plt.axis('off')



# Display the plot of the original data before clustering
plt.scatter(X1[:, 0], X1[:, 1], marker='.')
# Display the plot
plt.show()
```



نحسب مصفوفة التقارب :

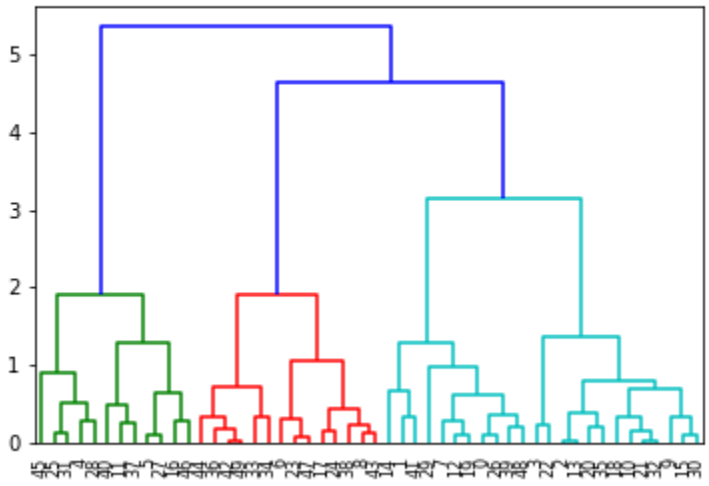
```
[7]: dist_matrix = distance_matrix(X1,X1)
     print(dist_matrix)
```

```
[[0.         0.11231913 0.55467479 ... 0.23558676 0.0766225  0.3502003 ]
 [0.11231913 0.         0.48205276 ... 0.3390299  0.0615597  0.46030415]
 [0.55467479 0.48205276 0.         ... 0.7826964  0.54192885 0.87156257]
 ...
 [0.23558676 0.3390299  0.7826964  ... 0.         0.28264915 0.13610882]
 [0.0766225  0.0615597  0.54192885 ... 0.28264915 0.         0.40843322]
 [0.3502003  0.46030415 0.87156257 ... 0.13610882 0.40843322 0.        ]]
```

ونرسم مخطط dendrogram الذي يعبر عن التجميع الهرمي :

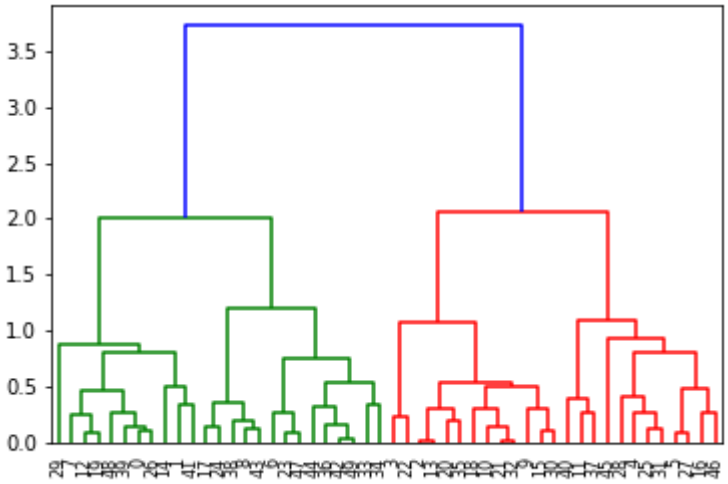
```
[8]: Z = hierarchy.linkage(dist_matrix, 'complete')
```

```
[9]: dendro = hierarchy.dendrogram(Z)
```



نلاحظ أننا استخدمنا طريقة complete عند حساب Z إذ يمكننا اختيار طرق أخرى مثل average :

```
Z = hierarchy.linkage(dist_matrix, 'average')
dendro = hierarchy.dendrogram(Z)
```

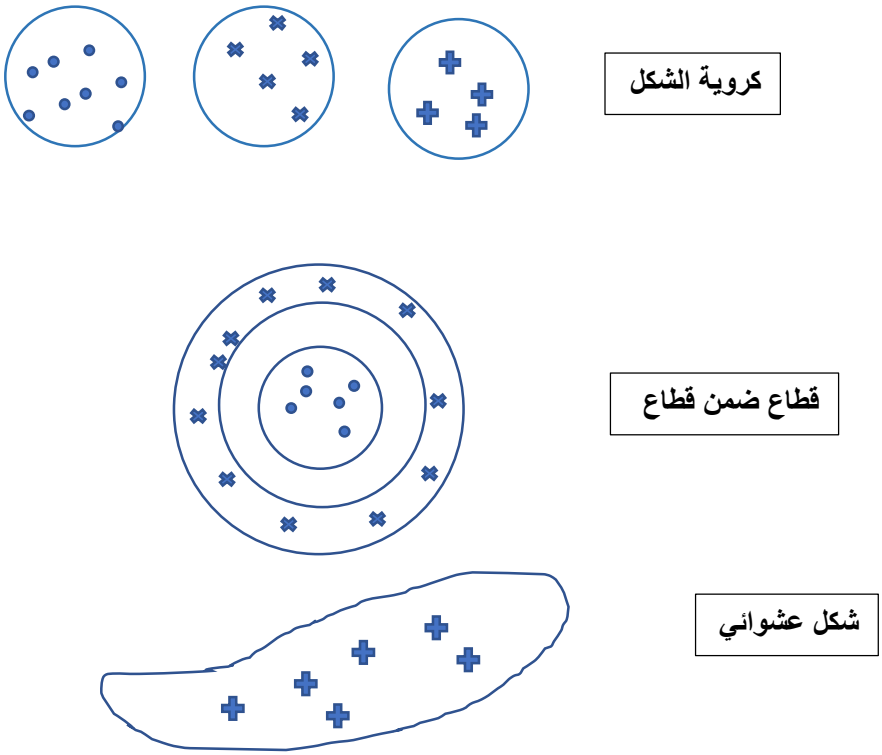


هذه الطرق الإحصائية التي تعطي نتائج مختلفة في التجميع والتي تتعلق بطبيعة المسألة ونوع البيانات والنتيجة الأفضل التي نرجوها .

- **التجميع Density-Based Spatial Clustering of Applications with Noise DBSAN :**

وهي مناسبة appropriate للاستخدام في اختبار البيانات أي يكون لها دراية في البيانات الشاذة فمعظم طرق التجميع الاعتيادية السابقة k-means,hierarchical,fuzzy clustering,.... تقوم بتجميع البيانات بطريقة غير إشرافية unsupervised فهي خوارزميات قائمة على التقسيم partitioning-based فهي عملية و سهلة الفهم ولكن ليس لها دراية notion بالقيم الشاذة outlies أي كل النقاط ستنتمي لقطاع ما حتى ولو لم تكن منه فهذه الأمور قد تتسبب ببعض المشاكل فوجود النقاط الشاذة سيسبب في تحديد مركز القطاع الموجودة ضمنه .

فعند تجميع النقاط يظهر عندنا عدة انماط مثل : spherical-shape clusters كروية الشكل أو كيفية الشكل arbitrary-shape clusters (اعتباطية)



تقنيات التجميع السابقة لن تكون مجدية عندما تكون العناصر في نفس القطاع غير متشاركة بشكل كاف بالتشابه فيما بينها وسيكون الأداء ضعيفا .

بالمقابل in contrast فإن خوارزمية DBSCAN تقوم بتجميع النقاط من حيث كثافتها فعندما تكون كثيفة كفاية تجمعها سوية وتفصلها عن المجموعات الأقل كثافة وتعرف هنا الكثافة على أنها عدد النقاط في نصف قطر محدد radius وهي فعالة جدا في التجميع الذي له علاقة بالمكان أو الحيز (مثل التجمعات السكنية و غيرها) وتعتبر الميزة الأساسية لها هي اكتشاف حيز تجمع النقاط بأي شكل عشوائي موجودة ضمنه دون التأثر بالضجيج أو بالقيم الشاذة .

مثال عن ذلك خريطة تحدد مواقع محطات الأرصاد الجوية the location of weather station في كندا هنا يمكن لخوارزمية DBSCAN أن تجمع المراكز (المراصد) التي تعطي نفس الأخبار الجوية .

حيث ليس فقط تقوم هذه الخوارزمية أشكال القطاعات المتعددة بل تحدد كذلك الجزء الأكثر كثافة بالبيانات وذلك بتجاهل المناطق الأقل كثافة أو التي فيها ضجيج أو قيم شاذة .

لنرى كيف تعمل هذه الخوارزمية :

تعتمد هذه الخوارزمية كما أسلفنا علة كثافة النقاط وذلك بقياس مدى قرب هذه النقاط من بعضها البعض في نفس القطاع لتقوم بتجميعها وتعتمد هذه العملية الحسابية على أمرين أولهما نصف القطر (radius of neighborhood) R ما هي النقاط الموجودة على بعد R من نقطة ما حيث بالاعتماد على عدد هذه النقاط يمكننا تسميتها منطقة كثيفة dense-area أو لا .

والأمر الثاني هو (minimum number of neighborhood) M أقل عدد من النقاط المجاورة لنقطة ما لنقوم بتجميعهم مع بعض وتشكيل قطاع مستقل .

لنفرض مثلا R = 2 cm و M = 6 أي لدينا ضمن دائرة نصف قطرها 2 سم عدد من النقاط وهو 6 نقاط مع النقطة المدروسة .

يجب بداية تفحص نوع هذه النقاط حيث لدينا ثلاثة احتمالات :

نقطة مركزية core point – نقطة محيطية border point – نقطة (خارجية) شاذة outlier point .

حيث نختار نقطة عشوائية ونبدأ بتفحصها هل هي مركزية حيث تعتبر مركزية عندما تكون ضمن دائرة نصف قطرها R وضمن الدائرة 6 نقاط على الأقل (بالاعتماد على مثالنا)

فعندما لا تكون مركزية ننتقل لنقطة أخرى و نختبرها هل هي مركزية أم لا حتى ننتهي من جميع النقاط ويتحدد لنا النقطة المركزية .

فعلى فرض وجدنا عنجد اختبار نقطة ما وجود 5 نقاط بما فيها النقطة المختبرة ضمن دائرة نصف قطرها R=2 عندها تسمى نقطة محيطية وهي النقطة التي تحتوي دائرتها أقل من العدد M أو يمكن أن نصل إليها من خلال نقطة مركزية أخرى أي تكون ضمن دائرة مركزها نقطة مركزية .

وعندما لا يوجد حول النقطة ضمن دائرة نصف قطرها R أي نقاط مركزية أو عدد نقاط M فهي تكون نقطة شاذة .

عندما ننتهي من فرز النقاط نقوم عندها بوصل النقاط المركزية القريبة من بعضها لتشكل قطاع مشترك .

فالقطاع يتشكل على الأقل من نقطة مركزية و النقاط التي تم الوصول إليها ابتداء من النقطة المركزية بدائرة نصف قطرها R مع النقاط المحيطية ، وبهذا يتم عزل النقاط الشاذة بشكل جيد

يمكننا تلخيص خصائص DBSCAN :

1- Arbitrarily shaped clusters .
2- Robust to outlier points .
3- Doesn’t require specification of the number of the clusters such as k-means .

سيتم تبيان ميزة DBSCAN في انتقاء النقاط الشاذة وذلك بنفس مثال k-means بتوليد مجموعة بيانات عشوائية وتجميعها :

نبدأ باستيراد المكتبات اللازمة :

[1]:
```
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
%matplotlib inline
```

نولد مجموعة البيانات العشوائية :

Data generation

The function below will generate the data points and requires these inputs:

- **centroidLocation**: Coordinates of the centroids that will generate the random data.
  - Example: input: [[4,3], [2,-1], [-1,4]]
- **numSamples**: The number of data points we want generated, split over the number of centroids (# of centroids defined in centroidLocation)
  - Example: 1500
- **clusterDeviation**: The standard deviation between the clusters. The larger the number, the further the spacing.
  - Example: 0.5

2]:
```
def createDataPoints(centroidLocation, numSamples, clusterDeviation):
    # Create random data and store in feature matrix X and response vector y.
    X, y = make_blobs(n_samples=numSamples, centers=centroidLocation,
                                cluster_std=clusterDeviation)

    # Standardize features by removing the mean and scaling to unit variance
    X = StandardScaler().fit_transform(X)
    return X, y
```

Use **createDataPoints** with the **3 inputs** and store the output into variables **X** and **y**.

[3]:
```
X, y = createDataPoints([[4,3], [2,-1], [-1,4]] , 1500, 0.5)
```

ندرب النموذج لدينا :

Modeling

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. This technique is one of the most common clustering algorithms which works based on density of object. The whole idea is that if a particular point belongs to a cluster, it should be near to lots of other points in that cluster.

It works based on two parameters: Epsilon and Minimum Points
Epsilon determine a specified radius that if includes enough number of points within, we call it dense area
minimumSamples determine the minimum number of data points we want in a neighborhood to define a cluster.

[4]:
```
epsilon = 0.3
minimumSamples = 7
db = DBSCAN(eps=epsilon, min_samples=minimumSamples).fit(X)
labels = db.labels_
labels
```

[4]:
```
array([0, 1, 0, ..., 2, 2, 0])
```

نوجد النقاط الشاذة حتى لا تدخل ضمن القطاعات :

Distinguish outliers

Lets Replace all elements with 'True' in core_samples_mask that are in the cluster, 'False' if the points are outliers.

[5]:
```
# Firts, create an array of booleans using the labels from db.
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
core_samples_mask
```

[5]:
```
array([ True,  True,  True, ...,  True,  True,  True])
```

[6]:
```
# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_clusters_
```

[6]:
```
3
```

[7]:
```
# Remove repetition in labels by turning it into a set.
unique_labels = set(labels)
unique_labels
```

[7]:
```
{-1, 0, 1, 2}
```

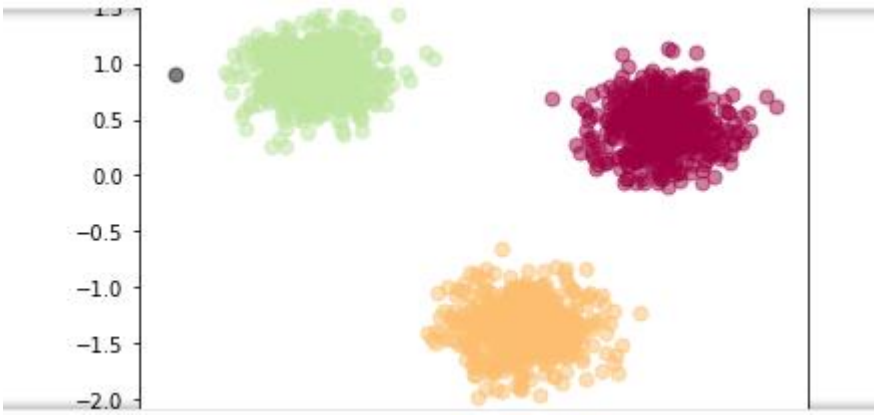
نرسم البيانات :

Data visualization

[8]:
```
# Create colors for the clusters.
colors = plt.cm.Spectral(np.linspace(0, 1, len(unique_labels)))
colors
```

[8]:
```
array([[0.61960784, 0.00392157, 0.25882353, 1.        ],
       [0.99346405, 0.74771242, 0.43529412, 1.        ],
       [0.74771242, 0.89803922, 0.62745098, 1.        ],
       [0.36862745, 0.30980392, 0.63529412, 1.        ]])
```

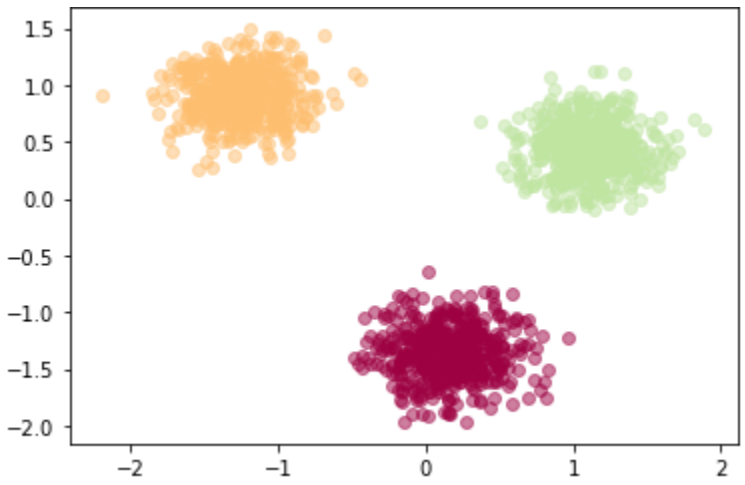
[9]:
```
# Plot the points with colors
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = 'k'

    class_member_mask = (labels == k)

    # Plot the datapoints that are clustered
    xy = X[class_member_mask & core_samples_mask]
    plt.scatter(xy[:, 0], xy[:, 1],s=50, c=col, marker=u'o', alpha=0.5)

    # Plot the outliers
    xy = X[class_member_mask & ~core_samples_mask]
    plt.scatter(xy[:, 0], xy[:, 1],s=50, c=col, marker=u'o', alpha=0.5)
```



نلاحظ وجود نقطة شاذة في الجانب الأيسر تم اكتشافها بخوارزمية DBSCAN .

سنعيد نفس المثال باستخدام k-means لنرى الفرق في التعامل مع النقاط الشاذة :

[10]:
```
# write your code here
from sklearn.cluster import KMeans
k = 3
k_means3 = KMeans(init = "k-means++", n_clusters = k, n_init = 12)
k_means3.fit(X)
fig = plt.figure(figsize=(6, 4))
ax = fig.add_subplot(1, 1, 1)
for k, col in zip(range(k), colors):
    my_members = (k_means3.labels_ == k)
    plt.scatter(X[my_members, 0], X[my_members, 1],  c=col, marker=u'o', alpha=0.5)
plt.show()
```



نلاحظ أن النقطة الشاذة التي تم إيجادها باستخدام DBSCAN تم ضمها للقطاع الثالث في خوارزمية k-means .

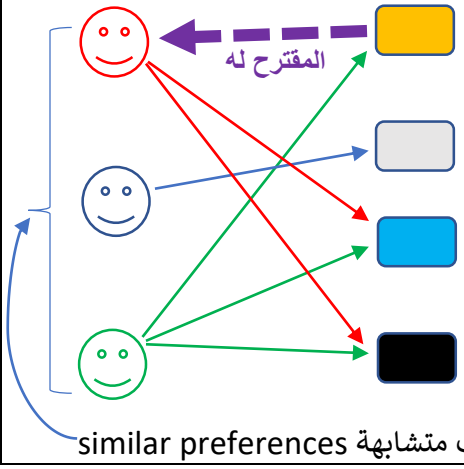
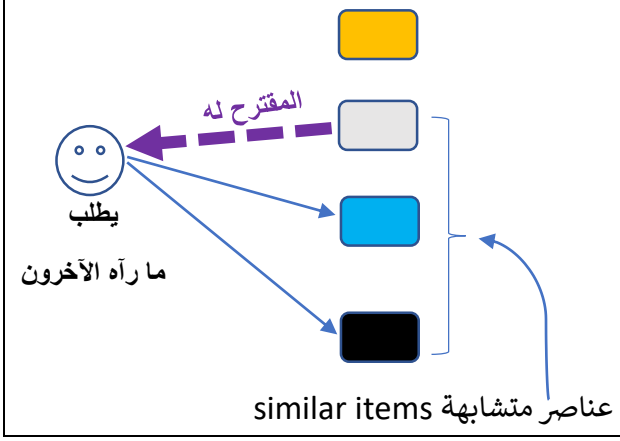
17- أنظمة التوصية Recommended Systems :

تلتقط هذه الأنظمة الأنماط التي يحددها الناس في سلوكهم و يستخدمها ليتوقع ماذا يمكن أن يختاروا أو ماذا يفعلون ، فمهما كان الناس مختلفين في خياراتهم إلا أنهم يتبعون انماطا patterns محددة أثناء حياتهم وبالتالي فهناك تشابه في الأشياء التي يرغبها الناس .

فالناس يميلون trend لاختيار أشياء في نفس الصنف same category أو الأشياء التي تشترك في الصفات same characteristics ، فمثلا لو أنك اشتريت purchased كتابا عن لغة البايثون فإنك ستحب قراءة كتاب تحليل البيانات ، كذلك الأمر عند انتقاء المأكولات والألبسة.....حيث يتم استخدام مثل هذه الأنظمة في مواقع ويب عديدة مثل أمازون amazon و نتفليكس Netflix وذلك بالاعتماد على طلبات الزبائن ، وأيضا في بعض تطبيقات الموبايل على الويب وأيضا الفيس بوك الذي يقترح عليك إضافة أصدقاء ممكن ان يكونوا قد أعجبوا بنفس الصفحات التي أعجبت بها أنت أثناء تصفحك .

بعض محاسن هذه الأنظمة هي border exposure العرض الشامل أو الواسع للسلع أو للخدمات بحيث يعطي منتجات أفضل وخدمات أحسن للزبون ويحدد خصوصية أكبر له.

يوجد بشكل رئيسي نوعان لهذه الأنظمة :

Collaborative Filtering التصفية بالاعتماد على التفضيل	Content-Based التصفية بالاعتماد على المحتوى
Tell me what's popular among my neighbors, I also might like it. هنا يبحث عن مجموعة الأشخاص القريبين منه ماذا يفضلون ويقترحها للزبون .  	Show me the sane of what I've liked before . هنا يبحث عن العناصر المشابهة لما يحبه الزبون (اختاره سابقا) ويقترحها . 

كذلك يوجد نوع آخر Hybrid recommender Sys أنظمة هجينة تحتوي على عدة تقنيات مجتمعة .

وهنالك أسلوبين لتطبيق أنظمة التوصية :

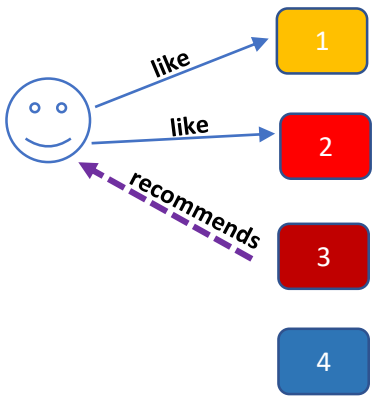
- **Memory-based :** حيث يتم استخدام جميع بيانات العناصر و الأشخاص entire user-item dataset لتقديم الاقتراحات ويتم استخدام عدة تقنيات إحصائية : Pearson Correlation – Cosine Similarity – Euclidean Dis…..

- **Model-based :** يتم هنا تطوير نموذج للمستخدمين فقط لمحاولة معرفة رغباتهم ويتم استخدام تقنيات تعلم الآلة مثل : Regression – Clustering – Classification .

- أنظمة التوصية بالاعتماد على المحتوى content-based :

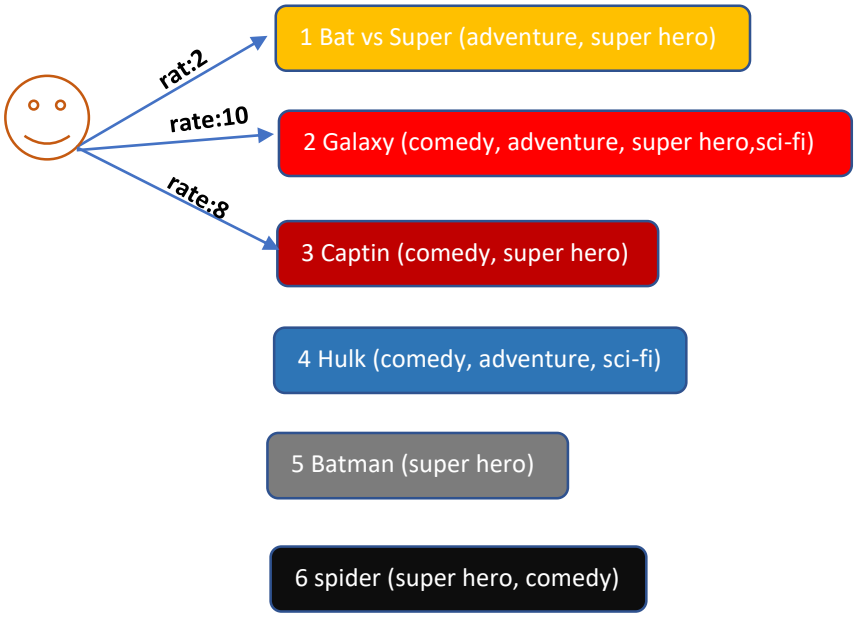
تقوم هذه الطريقة على تقديم العنصر الأفضل للزبون بالاعتماد على بروفايل المستخدم نفسه حيث يركز البروفايل على ما يحبه المستخدم وهذا يعتمد على عدد المرات التي اختار فيها المستخدم عنصر ما وما يدل على اهتمام المستخدم بهذا العنصر كوضع like له .

حيث يعتمد نظام التوصية على مقدار التشابه بين العناصر و هذا التشابه أو التقارب يُقاس بالاعتماد على تشابه المحتوى ضمن هذه العناصر والمقصود بالمحتوى content صنف العنصر category و علامته tag ونوعه gender وهكذا ...



فلو قام المستخدم بالإعجاب بالفلمين 1 و 2 و كان الفلم رقم 3 من نفس النوع أو المحتوى سيقدمه نظام التوصية للمستخدم كاقتراح له وذلك معتمدا على التشابه في المحتوى .

ليكن لدينا مجموعة من 6 أفلام رأى منها المستخدم ثلاثة وأعطاها تفضيلا rating ونريد من نظام التوصية أن يقدم الفلم الأفضل من الثلاثة المتبقية ليراها المستخدم .



كما نلاحظ لدينا تصنيف كل فلم ونريد من النظام أن يتوقع ماذا سيقيم المستخدم الأفلام الثلاثة المتبقية ، بالاعتماد على المحتوى سنحتاج التعرف على بروفايل المستخدم :

أولا ننشئ متجه (شعاع) يحوي الأفلام التي رآها المستخدم مع العلامات التي اختاراها لهم ويسمى هذا المتجه input user ratings .

mov	rating
1	2
2	10
3	8

ثم نرمز هذه الأفلام التي رآها بطريقة One Hot Encoding وتسمى movies matrix:

movie	Comedy	adventure	Super hero	Sci-fi
1 (adventure, super hero)	0	1	1	0
2 (comedy, adventure, super hero,sci-fi)	1	1	1	1
3 (comedy, super hero)	1	0	1	0

حيث يتم استخدامها كمجموعة ميزات :

feature set (comedy,adventure,super hero,sci-fi)

ثم نجري جداء بين المصفوفتين السابقتين لنحصل على أوزان هذه السمات weighted feature set نحصل على المصفوفة weighted matrix :

movie	Comedy	adventure	Super hero	Sci-fi
1	0	2	2	0
2	10	10	10	10
3	8	0	8	0

نجمع كل عمود على حدا لنحصل على بروفايل المستخدم :

	Comedy	adventure	Super hero	Sci-fi
User profile	18	12	20	10

مجموع القيم 60 نقسم كل خانة على 60 فنحصل على التقييم :

	Comedy	adventure	Super hero	Sci-fi
User profile	0.3	0.2	0.33	0.16

نلاحظ أخيرا أنه يفضل نوع super hero ثم comedy ثم adventure ثم sci-fi

لدينا ثلاثة أفلام لم يشاهدها المستخدم سنجري لها مصفوفة ترميز كما فعلنا بالثلاثة السابقة:

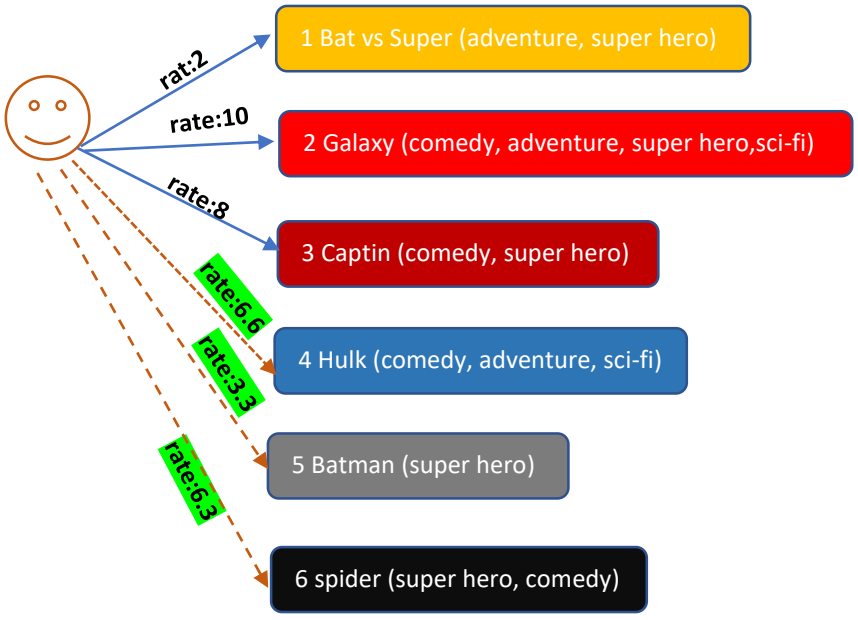
movie	Comedy	adventure	Super hero	Sci-fi
4 (comedy, adventure, sci-fi)	1	1	0	1
5 (super hero)	0	0	1	0
6 (comedy, super hero)	1	0	1	0

ونجري جداء بينها و بين بروفايل المستخدم فنحصل على weighted matrix :

movie	Comedy	adventure	Super hero	Sci-fi
4	0.3	0.2	0	0.16
5	0	0	0.33	0
6	0.3	0	0.33	0

ثم نجمع كل سطر ونضربه بـ 10 لأن التقييم لدينا من 10 فنحصل على recommendation matrix :

4	0.66 * 10 = 6.6
5	0.33 * 10 = 3.3
6	0.63 * 10 = 6.3



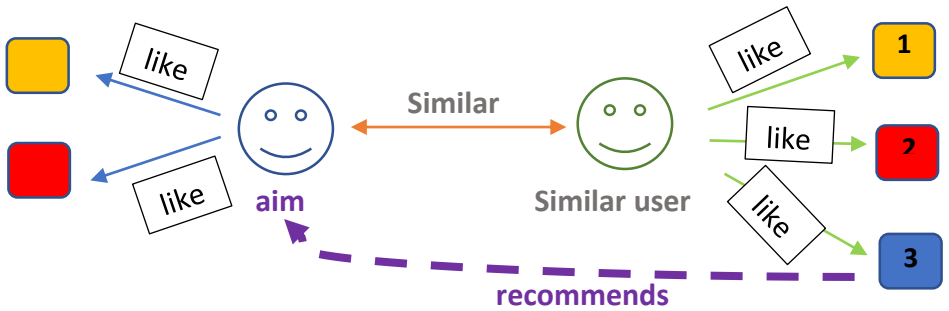
نجد أن نظام التوصية سيقدم الفلم رقم 4 ليشاهده المستخدم ومع ذلك ربما لا تعطي هذه الطريقة نتيجة مرضية لأنه ربما يوجد فلم آخر ليس موجود ضمن بروفايل المستخدم ومن نوع آخر ليكن drama فهذه الطريقة تعاملت فقط مع الأفلام التي ضمن بروفايل المستخدم دون الأخذ بعين الاعتبار الأفلام الأخرى المتواجدة في الموقع هنا يأتي دور الطريقة التالية وهي collaborative filtering .

- أنظمة التوصية بالاعتماد على الأفضلية Collaborative Filtering :

تعتمد هذه الطريقة على وجود علاقة بين المنتجات و اهتمام الناس بها فأنظمة التوصية تعتمد على إيجاد هذه العلاقة و إعطاء توصية دقيقة للمنتج الذي يحبه المستخدم أو يكون ممتعا له و لهذا النمط طريقتان :

- التجميع المفلتر بالاعتماد على الأشخاص user-based collaborative filtering تعتمد على التشابه بين الأشخاص أو على التقارب فيما بينهم .
- الاعتماد على المنتجات (العناصر) item-based collaborative filtering تعتمد على التشابه بين المنتجات .

لفهم مقصد الطريقة الأولى user-based لدينا أشخاص فعالين (نشيطين) يكونون هدف نظام التوصية فيبحث النظام عن الأشخاص المشابهين بنشاطهم للهدف ويعتمد التشابه فيما بينهم على : الاختيارات choices – السجلات history – الميزات preferences و من ثم يتم توقع المنتجات التي لم يتطرق لها الهدف ليتم اقتراحها له (لأن الذي يشبهه يكون قد تطرق لها فيتم احتمال أن تعجبه للهدف) .



ليكن لدينا المعلومات التي تضم معدلات تفضيل ratings لأربع زبائن لخمس أفلام :

	Movie1	Movie2	Movie3	Movie4	Movie5
User1	9	6	8	4	-
User2	2	10	6	-	8
User3	5	9	-	10	7
User4	?	10	7	8	?

هذه تدعى مصفوفة Ratings matrix .

المستخدم الرابع هو الهدف و النشط حاليا active user و سنحاول اكتشاف أي الأفلام التي لم يشاهدها (1,5) ينبغي أن تعرض له .

المرحلة الأولى نجد المستخدمين المشابهين للهدف ويتم ذلك بعد طرق إحصائية statistical و شعاعية victor مثل : distance or similarity measurements , Euclidean distance, Pearson correlation, Cosine similarity .....

فلمعرفة مدى التشابه بين عنصرين يتم ذلك بالاعتماد على الأفلام التي شاهدها كلا المستخدمين وبغض النظر عن الطريقة المتبعة في ذلك حاليا لنفترض أن مقدار التشابه بين العنصر الهدف وهو في مثالانا الرابع و البقية كالتالي :

	U1	U2	U3
U4	0.4	0.9	0.7

تعبر هذه القيم عن proximity or weight بين العنصر الهدف وبقية عناصر المجموعة .

الخطوة التالية هي إيجاد مصفوفة weighted matrix :

	Mov1	Mov5
U1	9	
U2	2	8
U3	5	7

Rating matrix subset

	Similarity index
U1	0.4
U2	0.9
U3	0.7

Similarity matrix

بالجداء بين كل قيمتين نحصل على المصفوفة

Mov1	Mov5
3.6	
1.8	7.2
3.5	4.9

Weighted matrix

تعبر هذه المصفوفة الأخيرة عن تفضيلات المستخدمين (المشابهين لهدفنا) للأفلام التي نريد معرفة أي منها ممكن أن يختار زبوننا الرابع لنقدمها له .

الآن نجمع نتائج كل عمود بشكل مستقل لنجد rating weighted matrix :

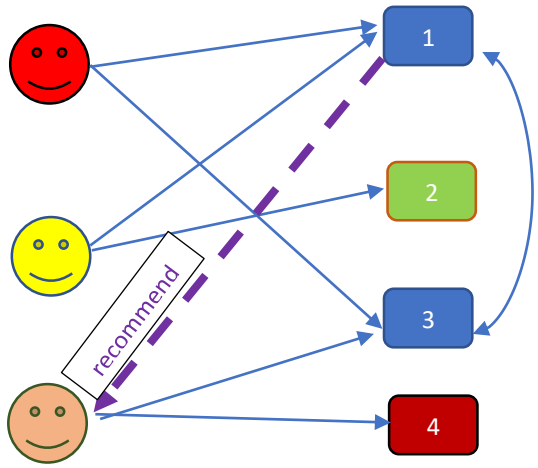
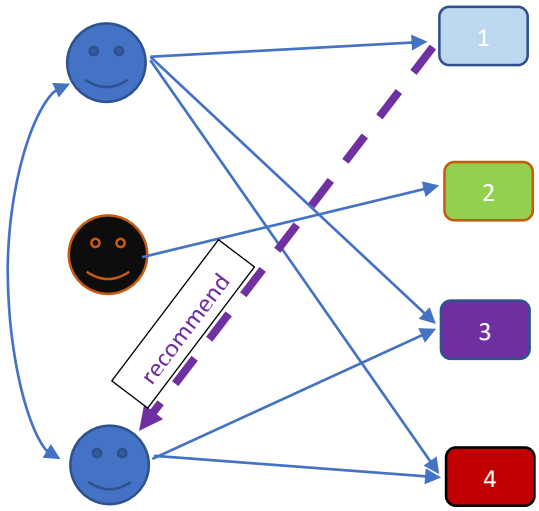
	Mov1	Mov5
U2+U3		7.2+4.9=12.1
U1+U2+U3	3.6+1.8+3.5=8.9	

نقسم نتيجة جمع كل عمود على مجموع تشابه العناصر أي :

	Mov1	Mov5
User4	8.9/(0.4+0.9+0.7) = 4.4	12.1/(0.9+0.7)= 7.5

هذه تدعى مصفوفة recommendation matrix نلاحظ أن الفلم رقم 5 هو الذي سنقدمه للمستخدم 4 والذي نتوقع أن يختاره أكثر من الفلم 1 .

قلنا سابقا أن هنالك طريقتين للفلترة item-based & user-based لنرى سريعا الفرق بينهما:

Item-based	User-based
	
هنا تشابه بين العنصرين 1,3 واختارهما كل من المستخدمين 1,2 وبالتالي يمكن أن يقدم المنتج 1 للمستخدم 3 لأنه استخدم من قبل المنتج 3 الذي يشبه المنتج 1 .	**هنا التشابه بين المستخدمين 1,3 بناء على مشاركتهم للمنتجين 3,4 وبالتالي سنقدم المنتج 1 للمستخدم 3**

مع أن طريقة collaborative filtering فعالة ولكن توجد بعض التحديات التي تواجهها مثل تباين البيانات data sparsity وهذا يحدث عند وجود عدد كبير من المستخدمين يختارون عدد محدد من المنتجات وبالتالي لن يكون لدينا تقييمات ratings كافية لكل من user, item فلن نستطيع تقديم خيار مفضل للمستخدم وهذه الحالة تعبر عن الصعوبة التي تواجه نظام التوصية عند وجود مستخدمين جدد cold start ليس لديهم بروفايل سابق لديه يمكنه الاعتماد عليه وكذلك عند وجود منتجات جديدة لم يختارها بعد أي مستخدم .

هنا عندما يصبح المستخدمين أو حتى العناصر لديهم قابلية التوسعة scalability وفي ازدياد دائم سيضع خوارزمية التصفية هذه تقع في أخطاء بسبب التشابه الذي يحصل بين المستخدمين و المنتجات على حد سواء مما يسبب صعوبة التفضيل فيما بينهم .

يمكننا في مثل هذه الحالات أن ننتقل لخوارزمية أخرى hybrid-based recommendation .

خاتمة :

في النهاية أعتذر عن كل خطأ لغوي أو برمجي أو علمي والملخص قابل للتعديل في محتواه وإعادة نشره مع ذكر مصدره وذلك لتعم الفائدة العلمية الدقيقة في مجتمعنا .

لا تنسوني من صالح دعائكم

والحمد لله رب العالمين