القسم العاشر: مكتبة SKlearn

A. Data Preparation

1. Data files from SKlearn
2. Data cleaning
3. Metrics module
4. Feature Selction
5. Data Scaling
6. Data Split

B. ML Algorithms

1. Linear Regression
2. Logistic Regression
3. Neural Network
4. SVR
5. SVC
6. K-means
7. PCA
8. Decision Tree
9. Ensemble Regression
10. Ensemble Classifier
11. K Nearest Neighbors
12. Naïve Bayes
13. LDA , QDA
14. Hierarchical Clusters
15. DbScan
16. NLP
17. Apriori

C. Algorithm Evaluation :

1. Model Check
2. Grid Search
3. Pipeline
4. Model Save

D. Time Series

1.5) Data Scaling

و هي خاصة بعملية تدريج البيانات Scaling بأنواعها و هي تأتي من موديول preprocessing .

1.5.1 preprocessing.StandardScaler
1.5.2 preprocessing.MinMaxScaler
1.5.3 preprocessing.Normalizer
1.5.4 preprocessing.MaxAbsScaler
1.5.5 preprocessing.FunctionTransformer
1.5.6 preprocessing.Binarizer
1.5.7 preprocessing.PolynomialFeatures

خطوات تفيذ العملية :

- عمل كائن object باسم الكلاس المطلوب
- إعطاء أمر fit
- عمل متغير جديد يساوي أمر transform

1.5.1) Standard Scaler (Standardization)

وهي العملية الأكثر شهرة , وفيها يتم طرح القيمة ناقص المتوسط (ميو) مقسومة علي الإنحراف المعياري (سيجما)

الصيغة :

```
#Import Libraries
from sklearn.preprocessing import StandardScaler
#------------------------------------------------

#Standard Scaler for Data
scaler = StandardScaler(copy=True, with_mean=True, with_std=True)
X = scaler.fit_transform(X)

#showing data
print('X \n' , X[:10])
print('y \n' , y[:10])
```

مثال

```
from sklearn.preprocessing import StandardScaler
data = [[0, 0], [0, 0], [1, 1], [1, 1]]
scaler = StandardScaler()
scaler.fit(data)
print(scaler.mean_)
newdata = scaler.transform(data)
print(newdata)
```

أو قد نستخدم أمر fit_transform مرة واحدة

```
newdata = scaler.fit_transform(data)
print(newdata)
```

و نستخدمها في كلا من بيانات التدريب و الإختبار

```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)
y_test = sc_y.fit_transform(y_test)
```

1.5.2) MinMaxScaler (Normalization)

و فيها يتم طرح القيمة ناقص المتوسط علي المدي ,و هو الفرق بين أكبر و أصغر قيمة , ويكون السكيل للارقام بحيث تكون من صفر لواحد

الصيغة :

```
#Import Libraries
from sklearn.preprocessing import MinMaxScaler
#------------------------------------------------

#MinMaxScaler for Data

scaler = MinMaxScaler(copy=True, feature_range=(0, 1))
X = scaler.fit_transform(X)

#showing data
print('X \n' , X[:10])
print('y \n' , y[:10])
```

مثال

```
from sklearn.preprocessing import MinMaxScaler
data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
scaler = MinMaxScaler()
scaler.fit(data)
print(scaler.data_range_)
print(scaler.data_min_)
print(scaler.data_max_)
newdata = scaler.transform(data)
print(newdata)
```

أو قد نستخدم أمر fit_transform مرة واحدة

```
newdata = scaler.fit_transform(data)
print(newdata)
```

و ممكن تغير رينج الارقام بحيث تكون من كذا لكذا

```
scaler = MinMaxScaler(feature_range = (1,5))
```

1.5.3) Normalizer

و هي مخصصة لتناول كل صف علي حدة في المصفوفات ثنائية الأبعاد

الصيغة :

```
#Import Libraries
from sklearn.preprocessing import Normalizer
#----------------------------------------------

#Normalizing Data

scaler = Normalizer(copy=True, norm='l2') # you can change the norm to 'l1' or 'max'
X = scaler.fit_transform(X)

#showing data
print('X \n' , X[:10])
print('y \n' , y[:10])
```

مثال

```
from sklearn.preprocessing import Normalizer
X = [[4, 1, 2, 2], [1, 3, 9, 3], [5, 7, 5, 1]]

#transformer = Normalizer(norm='l1' )

#transformer = Normalizer(norm='l2' )

transformer = Normalizer(norm='max' )

transformer.fit(X)
transformer.transform(X)
```

تستخدم l1 لجعل مجموع كل صف هو القيمة العظمي

تستخدم l2 لجعل جذر مجموع مربعات كل صف هو القيمة العظمي

تستخدم max لجعل القيمة العظمي في كل صف هي القيمة العظمي

1.5.4) MaxAbsScaler

مشابهة لـ normalizer , لكن بالنسبة للعمود و ليس الصف حيث تجعل أكبر قيمة في كل عمود هي القيمة العظمي و تغير الباقيين علي اساسه

الصيغة :

```
#Import Libraries
from sklearn.preprocessing import MaxAbsScaler
#------------------------------------------------

#MaxAbsScaler Data

scaler = MaxAbsScaler(copy=True)
X = scaler.fit_transform(X)

#showing data
print('X \n' , X[:10])
print('y \n' , y[:10])
```

مثال

```
from sklearn.preprocessing import MaxAbsScaler
X = [[ 1., 10., 2.],
     [ 2., 0., 0.],
     [ 5., 1., -1.]]
transformer = MaxAbsScaler().fit(X)
transformer
transformer.transform(X)
```

1.5.5) FunctionTransformer

وهي تستخدم لعمل سكيل بدالة اقوم كتابتها بنفسي

الصيغة :

```
#Import Libraries
from sklearn.preprocessing import FunctionTransformer
#------------------------------------------------

#Function Transforming Data
'''
FunctionTransformer(func=None, inverse_func=None, validate= None,
                    accept_sparse=False,pass_y='deprecated', check_inverse=True,
                    kw_args=None,inv_kw_args=None)
'''

scaler = FunctionTransformer(func = lambda x: x**2,validate = True) # or func = function1
X = scaler.fit_transform(X)
```

```
#showing data
print('X \n' , X[:10])
print('y \n' , y[:10])
```

مثال

```
import numpy as np
from sklearn.preprocessing import FunctionTransformer

X = [[4, 1, 2, 2], [1, 3, 9, 3], [5, 7, 5, 1]]

def function1(z):
    return np.sqrt(z)

FT = FunctionTransformer(func = function1)
FT.fit(X)
newdata = FT.transform(X)
newdata
```

1.5.6) Binarizer

و هي تقوم بتحويل كل الأرقام إلي صفر او واحد , حسب قيمة العتبة threshold

الصيغة :

```
#Import Libraries
from sklearn.preprocessing import Binarizer
#----------------------------------------------

#Binarizing Data

scaler = Binarizer(threshold = 1.0)
X = scaler.fit_transform(X)

#showing data
print('X \n' , X[:10])
print('y \n' , y[:10])
```

مثال

```
from sklearn.preprocessing import Binarizer
X = [[ 1., -1., -2.],[ 2., 0., -1.], [ 0., 1., -1.]]

transformer = Binarizer(threshold=1.5 )
transformer.fit(X)

transformer

transformer.transform(X)
```

1.5.7) PolynomialFeatures

وهي خاصة بعمل فيتشرز جديدة , هي عبارة عن حاصل ضرب الفيتشرز الحالية بطريقة البولونوميال , فلو كانت الدرجة 2 مثلا , و كان لدينا اصلا عمودين فقط اي 2 فيتشرز , فسيعطي لنا علي الترتيب :

1, a, b, a^2, ab, b^2.

حيث رقم 1 في الاول لضربها في ثيتا صفر

الصيغة :

```
#Import Libraries
from sklearn.preprocessing import PolynomialFeatures
#-----------------------------------------------

#Polynomial the Data
scaler = PolynomialFeatures(degree=3, include_bias=True, interaction_only=False)
X = scaler.fit_transform(X)

#showing data
print('X \n' , X[:10])
print('y \n' , y[:10])
```

مثال

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
X = np.arange(6).reshape(3, 2)
```

يتم كتابة الدرجة , و هل تحتوي علي قيمة بياس (رقم 1) ام لا

```
poly = PolynomialFeatures(degree=2 , include_bias = True)
poly.fit_transform(X)
```

ولو تم اختيار interaction_only كقيمة True سيعرض فقط قيم a مضروبة في b و يحذف الاسس للقيم الوحيدة

```
poly = PolynomialFeatures(interaction_only=True)
poly.fit_transform(X)
```