**القسم العاشر: مكتبة SKlearn**

1. **Data Preparation**
   1. Data files from SKlearn
   2. Data cleaning
   3. Metrics module
   4. Feature Selection
   5. Data Scaling
   6. Data Split
2. **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
3. **Algorithm Evaluation :**
   1. Model Check
   2. Grid Search
   3. Pipeline
   4. Model Save
4. **Time Series**

**2.11) KNN**

* و هي تطبيق لفكرة الجيران القريبة و تستخدم في التوقع او التصنيف او البينات بدون إشراف
* منها ثلاث أنواع رئيسية :
  + 1. neighbors.KNeighborsRegressor للتوقع
    2. neighbors.KNeighborsClassifier للتصنيف
    3. neighbors.NearestNeighbors البيانات دون إشراف

**2.11.1) K Neighbors Regressor**

* يتم استخدامها عبر الموديول . • neighbors.KNeighborsRegressor
* الـ parameters المستخدمة في الموديل :
* n\_neighbors عدد الجيران الذي سيتم اعتباره
* weights طريقة حساب التوقع , وتكون : uniform , distance, or defined function
* algorithm طريقة حساب المسافة للجيران , وتكون : auto , ball\_tree , kd\_tree , brute
* n\_jobs عدد المهام التي يتم تنفيذها بالتوازي
* الـ methods المستخدمة مع الموديل :
* fit(X,y) لعمل الفيت
* predict(X) لعمل التوقع
* kneighbors (X) لايجاد جيران نقطة معينة

**الصيغة العامة**

#Import Libraries

from sklearn.neighbors import KNeighborsRegressor

#----------------------------------------------------

#Applying KNeighborsRegressor Model

'''

#sklearn.neighbors.KNeighborsRegressor(n\_neighbors=5, weights=, algorithm=’auto’, leaf\_size=30,

# p=2, metric=’minkowski’, metric\_params=None,n\_jobs=None)

'''

KNeighborsRegressorModel = KNeighborsRegressor(n\_neighbors = 5, weights='uniform', #also can be : distance, or defined function

algorithm = 'auto') #also can be : ball\_tree , kd\_tree , brute

KNeighborsRegressorModel.fit(X\_train, y\_train)

#Calculating Details

print('KNeighborsRegressorModel Train Score is : ' , KNeighborsRegressorModel.score(X\_train, y\_train))

print('KNeighborsRegressorModel Test Score is : ' , KNeighborsRegressorModel.score(X\_test, y\_test))

print('----------------------------------------------------')

#Calculating Prediction

y\_pred = KNeighborsRegressorModel.predict(X\_test)

print('Predicted Value for KNeighborsRegressorModel is : ' , y\_pred[:10])

**مثال**

#Import Libraries

from sklearn.datasets import load\_boston

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import median\_absolute\_error

#----------------------------------------------------

#load boston data

BostonData = load\_boston()

#X Data

X = BostonData.data

#print('X Data is \n' , X[:10])

#print('X shape is ' , X.shape)

#print('X Features are \n' , BostonData.feature\_names)

#y Data

y = BostonData.target

#print('y Data is \n' , y[:10])

#print('y shape is ' , y.shape)

#----------------------------------------------------

#Splitting data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=44, shuffle =True)

#Splitted Data

#print('X\_train shape is ' , X\_train.shape)

#print('X\_test shape is ' , X\_test.shape)

#print('y\_train shape is ' , y\_train.shape)

#print('y\_test shape is ' , y\_test.shape)

#----------------------------------------------------

#Applying KNeighborsRegressor Model

'''

#sklearn.neighbors.KNeighborsRegressor(n\_neighbors=5, weights=, algorithm=’auto’, leaf\_size=30,

# p=2, metric=’minkowski’, metric\_params=None,n\_jobs=None)

'''

KNeighborsRegressorModel = KNeighborsRegressor(n\_neighbors = 5, weights='uniform', #also can be : distance, or defined function

algorithm = 'auto') #also can be : ball\_tree , kd\_tree , brute

KNeighborsRegressorModel.fit(X\_train, y\_train)

#Calculating Details

print('KNeighborsRegressorModel Train Score is : ' , KNeighborsRegressorModel.score(X\_train, y\_train))

print('KNeighborsRegressorModel Test Score is : ' , KNeighborsRegressorModel.score(X\_test, y\_test))

print('----------------------------------------------------')

#Calculating Prediction

y\_pred = KNeighborsRegressorModel.predict(X\_test)

print('Predicted Value for KNeighborsRegressorModel is : ' , y\_pred[:10])

#----------------------------------------------------

#Calculating Mean Absolute Error

MAEValue = mean\_absolute\_error(y\_test, y\_pred, multioutput='uniform\_average') # it can be raw\_values

print('Mean Absolute Error Value is : ', MAEValue)

#----------------------------------------------------

#Calculating Mean Squared Error

MSEValue = mean\_squared\_error(y\_test, y\_pred, multioutput='uniform\_average') # it can be raw\_values

print('Mean Squared Error Value is : ', MSEValue)

#----------------------------------------------------

#Calculating Median Squared Error

MdSEValue = median\_absolute\_error(y\_test, y\_pred)

print('Median Squared Error Value is : ', MdSEValue )

**مثال**

import numpy as np

import matplotlib.pyplot as plt

from sklearn import neighbors

np.random.seed(0)

X = np.sort(5 \* np.random.rand(40, 1), axis=0)

T = np.linspace(0, 5, 500)[:, np.newaxis]

y = np.sin(X).ravel()

# Add noise to targets

y[::5] += 1 \* (0.5 - np.random.rand(8))

# #############################################################################

# Fit regression model

n\_neighbors = 5

for i, weights in enumerate(['uniform', 'distance']):

knn = neighbors.KNeighborsRegressor(n\_neighbors, weights=weights)

y\_ = knn.fit(X, y).predict(T)

plt.subplot(2, 1, i + 1)

plt.scatter(X, y, c='b', label='data')

plt.plot(T, y\_, c='r', label='prediction')

plt.axis('tight')

plt.legend()

plt.title("KNeighborsRegressor (k = %i, weights = '%s')" % (n\_neighbors,

weights))

plt.tight\_layout()

plt.show()

**مثال**

import numpy as np

import pandas as pd

dataset = pd.read\_csv('houses.csv')

dataset.head(20)

from sklearn.impute import SimpleImputer

imp = SimpleImputer(missing\_values=np.nan, strategy='mean')

imp = imp.fit(dataset)

dataset = imp.transform(dataset)

X = dataset[:, :-1]

y = dataset[:, -1]

X

y

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X = sc.fit\_transform(X)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

X\_train

X\_test

y\_train

y\_test

from sklearn import neighbors

knn = neighbors.KNeighborsRegressor(n\_neighbors = 5)

y\_ = knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

y\_pred

from sklearn.metrics import mean\_absolute\_error

mean\_absolute\_error(y\_test, y\_pred)

from sklearn.metrics import mean\_squared\_error

mean\_squared\_error(y\_test, y\_pred)

**2.11.2) K Neighbors Classifier**

* يتم استخدامها عبر الموديول . • neighbors. KNeighborsClassifier
* الـ parameters المستخدمة في الموديل :
* n\_neighbors عدد الجيران الذي سيتم اعتباره
* weights طريقة حساب التوقع , وتكون : uniform , distance, or defined function
* algorithm طريقة حساب المسافة للجيران , وتكون : auto , ball\_tree , kd\_tree , brute
* n\_jobs عدد المهام التي يتم تنفيذها بالتوازي
* الـ methods المستخدمة مع الموديل :
* fit(X,y) لعمل الفيت
* predict(X) لعمل التوقع
* kneighbors (X) لايجاد جيران نقطة معينة
* score(X,y) لايجاد كفاءة الموديل

**الصيغة العامة**

#Import Libraries

from sklearn.neighbors import KNeighborsClassifier

#----------------------------------------------------

#Applying KNeighborsClassifier Model

'''

#sklearn.neighbors.KNeighborsClassifier(n\_neighbors=5, weights='uniform’, algorithm=’auto’, leaf\_size=30,

# p=2, metric='minkowski’, metric\_params=None,n\_jobs=None)

'''

KNNClassifierModel = KNeighborsClassifier(n\_neighbors= 5,weights ='uniform', # it can be distance

algorithm='auto') # it can be ball\_tree, kd\_tree,brute

KNNClassifierModel.fit(X\_train, y\_train)

#Calculating Details

print('KNNClassifierModel Train Score is : ' , KNNClassifierModel.score(X\_train, y\_train))

print('KNNClassifierModel Test Score is : ' , KNNClassifierModel.score(X\_test, y\_test))

print('----------------------------------------------------')

#Calculating Prediction

y\_pred = KNNClassifierModel.predict(X\_test)

y\_pred\_prob = KNNClassifierModel.predict\_proba(X\_test)

print('Predicted Value for KNNClassifierModel is : ' , y\_pred[:10])

print('Prediction Probabilities Value for KNNClassifierModel is : ' , y\_pred\_prob[:10])

**مثال**

#Import Libraries

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

#----------------------------------------------------

#load breast cancer data

BreastData = load\_breast\_cancer()

#X Data

X = BreastData.data

#print('X Data is \n' , X[:10])

#print('X shape is ' , X.shape)

#print('X Features are \n' , BreastData.feature\_names)

#y Data

y = BreastData.target

#print('y Data is \n' , y[:10])

#print('y shape is ' , y.shape)

#print('y Columns are \n' , BreastData.target\_names)

#----------------------------------------------------

#Splitting data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=44, shuffle =True)

#Splitted Data

#print('X\_train shape is ' , X\_train.shape)

#print('X\_test shape is ' , X\_test.shape)

#print('y\_train shape is ' , y\_train.shape)

#print('y\_test shape is ' , y\_test.shape)

#----------------------------------------------------

#Applying KNeighborsClassifier Model

'''

#sklearn.neighbors.KNeighborsClassifier(n\_neighbors=5, weights='uniform’, algorithm=’auto’, leaf\_size=30,

# p=2, metric='minkowski’, metric\_params=None,n\_jobs=None)

'''

KNNClassifierModel = KNeighborsClassifier(n\_neighbors= 5,weights ='uniform', # it can be distance

algorithm='auto') # it can be ball\_tree, kd\_tree,brute

KNNClassifierModel.fit(X\_train, y\_train)

#Calculating Details

print('KNNClassifierModel Train Score is : ' , KNNClassifierModel.score(X\_train, y\_train))

print('KNNClassifierModel Test Score is : ' , KNNClassifierModel.score(X\_test, y\_test))

print('----------------------------------------------------')

#Calculating Prediction

y\_pred = KNNClassifierModel.predict(X\_test)

y\_pred\_prob = KNNClassifierModel.predict\_proba(X\_test)

print('Predicted Value for KNNClassifierModel is : ' , y\_pred[:10])

print('Prediction Probabilities Value for KNNClassifierModel is : ' , y\_pred\_prob[:10])

#----------------------------------------------------

#Calculating Confusion Matrix

CM = confusion\_matrix(y\_test, y\_pred)

print('Confusion Matrix is : \n', CM)

# drawing confusion matrix

sns.heatmap(CM, center = True)

plt.show()

**مثال**

from sklearn.datasets import load\_iris

iris = load\_iris()

X = iris.data

y = iris.target

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.4,random\_state = 0)

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

k\_range = range(1,26)

scores = []

for k in k\_range:

knn = KNeighborsClassifier(n\_neighbors= k)

knn.fit(X\_train , y\_train)

y\_pred = knn.predict(X\_test)

scores.append(metrics.accuracy\_score(y\_test , y\_pred) )

import matplotlib.pyplot as plt

plt.plot(k\_range , scores)

plt.xlabel('Values for k in KNN')

plt.ylabel('testing accuracy')

**مثال**

import pandas as pd

df = pd.read\_csv('diabetes.csv')

df.head()

df.shape

X = df.drop(columns=['Outcome'])

X.head()

y = df['Outcome'].values

y[0:5]

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=1, stratify=y)

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors = 3)

knn.fit(X\_train,y\_train)

knn.predict(X\_test)[0:5]

knn.score(X\_test, y\_test)

from sklearn.model\_selection import cross\_val\_score

import numpy as np

knn\_cv = KNeighborsClassifier(n\_neighbors=3)

cv\_scores = cross\_val\_score(knn\_cv, X, y, cv=5)

print(cv\_scores)

print('cv\_scores mean:{}'.format(np.mean(cv\_scores)))

**2.11.3) Nearest Neighbors**

* يتم استخدامها عبر الموديول . • neighbors.NearestNeighbors
* الـ parameters المستخدمة في الموديل :
* n\_neighbors عدد الجيران الذي سيتم اعتباره
* radius نصف القطر الذي سيتم الحساب علي اساسه
* algorithm طريقة حساب المسافة للجيران , وتكون : auto , ball\_tree , kd\_tree , brute
* n\_jobs عدد المهام التي يتم تنفيذها بالتوازي
* الـ methods المستخدمة مع الموديل :
* fit(X) لعمل الفيت
* kneighbors (X) لايجاد الجيران الخاصة بنقطة معينة
* radius\_neighbors (X) لايجاد جيران نقطة معينة بنصف قطر معين

**الصيغة العامة**

#Import Libraries

from sklearn.neighbors import NearestNeighbors

#----------------------------------------------------

#Applying NearestNeighborsModel Model

'''

#sklearn.neighbors.NearestNeighbors(n\_neighbors=5, radius=1.0, algorithm='auto’,leaf\_size=30,

# metric='minkowski’, p=2, metric\_params=None, n\_jobs=None)

'''

NearestNeighborsModel = NearestNeighbors(n\_neighbors=4,radius=1.0,algorithm='auto')#it can be:ball\_tree,kd\_tree,brute

NearestNeighborsModel.fit(X\_train)

#Calculating Details

print('NearestNeighborsModel Train kneighbors are : ' , NearestNeighborsModel.kneighbors(X\_train[: 5]))

print('NearestNeighborsModel Train radius kneighbors are : ' , NearestNeighborsModel.radius\_neighbors(X\_train[: 1]))

print('NearestNeighborsModel Test kneighbors are : ' , NearestNeighborsModel.kneighbors(X\_test[: 5]))

print('NearestNeighborsModel Test radius kneighbors are : ' , NearestNeighborsModel.radius\_neighbors(X\_test[: 1]))

print('----------------------------------------------------')

**مثال**

#Import Libraries

import pandas as pd

from sklearn.neighbors import NearestNeighbors

#----------------------------------------------------

#reading data

data = pd.read\_csv('data.csv')

#data.describe()

#X Data

X = data

#print('X Data is \n' , X.head())

#print('X shape is ' , X.shape())

#----------------------------------------------------

#Applying NearestNeighborsModel Model

'''

#sklearn.neighbors.NearestNeighbors(n\_neighbors=5, radius=1.0, algorithm='auto’,leaf\_size=30,

# metric='minkowski’, p=2, metric\_params=None, n\_jobs=None)

'''

NearestNeighborsModel = NearestNeighbors(n\_neighbors=4,radius=1.0,algorithm='auto')#it can be:ball\_tree,kd\_tree,brute

NearestNeighborsModel.fit(X)

#Calculating Details

print('NearestNeighborsModel Train kneighbors are : ' , NearestNeighborsModel.kneighbors(X\_train[: 5]))

print('NearestNeighborsModel Train radius kneighbors are : ' , NearestNeighborsModel.radius\_neighbors(X\_train[: 1]))

print('NearestNeighborsModel Test kneighbors are : ' , NearestNeighborsModel.kneighbors(X\_test[: 5]))

print('NearestNeighborsModel Test radius kneighbors are : ' , NearestNeighborsModel.radius\_neighbors(X\_test[: 1]))

print('----------------------------------------------------')

**مثال**

import pandas as pd

from sklearn.neighbors import NearestNeighbors

data = pd.read\_csv('data.csv')

neigh = NearestNeighbors(2, 0.4)

neigh.fit(data)

data

l =[-2,.5,-0.8,1.1,1.5,0.1,-1,2]

result = neigh.kneighbors([l],n\_neighbors= 2) # returns distance nd index