القسم العاشر: مكتبة SKlearn

A. Data Preparation

1. Data files from SKlearn
2. Data cleaning
3. Metrics module
4. Feature Selection
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

1.6) Data Split

وهي خاصة بعمل تقسيم للبيانات قبل تدريبها , وهي تتم عبر استخدام model_selection من مكتبة sklearn
وهي بها أكثر من أداة :

1.6.1 model_selection.train_test_split
1.6.2 model_selection.Kfold
1.6.3 model_selection.RepeatedKFold
1.6.4 model_selection.StratifiedKFold
1.6.5 model_selection.RepeatedStratifiedKFold
1.6.6 model_selection.LeaveOneOut
1.6.7 model_selection.LeavePOut
1.6.8 model_selection.ShuffleSplit
1.6.9 model_selection.StratifiedShuffleSplit
1.6.10 model_selection.TimeSeriesSplit

1.6.1) train_test_split

تقسيم البيانات لتدريب و اختبار , يتم تحديد النسبة للاختبار , و عبر قيمة random_state يتم تحديد نظام العشوائية

الصيغة :

```
#Import Libraries
from sklearn.model_selection import train_test_split
#-------------------------------------------------

#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)
```

مثال

```
import numpy as np
from sklearn.model_selection import train_test_split
X, y = np.arange(10).reshape((5, 2)), range(5)

X
list(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

X_train
y_train
X_test
y_test
```

يمكن ان يتم عمل تقسيم داخلي فيها بحيث يكون عشوائي او غير عشوائي

```
train_test_split(y, shuffle=True)
```

1.6.2) KFold

الصيغة :

```
#Import Libraries
from sklearn.model_selection import KFold
#-----------------------------------------------

#KFold Splitting data

kf = KFold(n_splits=4, random_state=44, shuffle =True)

#KFold Data
for train_index, test_index in kf.split(X):
    print('Train Data is : \n', train_index)
    print('Test Data is  : \n', test_index)
    print('------------------------------')
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    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)
    print('==========================================')
```

مثال

```
import numpy as np
from sklearn.model_selection import KFold
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([11, 22, 33, 44])
```

يتم عمل اكثر من تجربة للتقسيم يساوي 2 او 3 او 4

```
nn = 2
#nn=3
#nn=4

kf = KFold(n_splits=nn)
kf.get_n_splits(X)
KFold(n_splits=nn, random_state=None, shuffle=False)

for train_index, test_index in kf.split(X):
```

```
print("TRAIN:", train_index, "TEST:", test_index)
X_train, X_test = X[train_index], X[test_index]
y_train, y_test = y[train_index], y[test_index]
print('X_train \n' , X_train)
print('X_test \n' , X_test)
print('y_train \n' ,y_train)
print('y_test \n' , y_test)
```

1.6.3) Repeated KFold

و هي تقوم بعمل kfold اكثر من مرة, بتنويعات مختلفة

```
import numpy as np
from sklearn.model_selection import RepeatedKFold
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([11, 22, 33, 44])
```

هنا يتم تحديد عدد الاقسام و عدد التكرارات

```
rkf = RepeatedKFold(n_splits=4, n_repeats=4, random_state=44)
for train_index, test_index in rkf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    print('X_train \n' , X_train)
    print('X_test \n' , X_test)
    print('y_train \n' ,y_train)
    print('y_test \n' , y_test)
    print('**********************')
```

مثال اخر

```
import numpy as np
from sklearn.model_selection import RepeatedKFold
X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
random_state = 12883823
rkf = RepeatedKFold(n_splits=2, n_repeats=2, random_state=random_state)
for train, test in rkf.split(X):
    print("%s %s" % (train, test))
```

1.6.4) Stratified KFold

و هي لعمل تقسيم الطبقات بشكل متوازن مع انواع الاصناف

```
import numpy as np
from sklearn.model_selection import StratifiedKFold
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([0,0,0,1])
skf = StratifiedKFold(n_splits=2)
skf.get_n_splits(X, y)

print(skf)

for train_index, test_index in skf.split(X, y):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    print('X_train \n' , X_train)
    print('X_test \n' , X_test)
```

```
print('y_train \n' ,y_train)
print('y_test \n' , y_test)
print('**********')
```

1.6.5) Repeated Stratified KFold

يجمع بين الصفتين في المثالين السابقين

```
import numpy as np
from sklearn.model_selection import RepeatedStratifiedKFold

X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([0,0,1,1])
rskf = RepeatedStratifiedKFold(n_splits=2, n_repeats=2,random_state=36851234)
for train_index, test_index in rskf.split(X, y):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    print('X_train \n' , X_train)
    print('X_test \n' , X_test)
    print('y_train \n' ,y_train)
    print('y_test \n' , y_test)
    print('**********************')
```

1.6.6) Leave One Out

يترك عنصر واحد للاختبار و الباقي للتدريب

```
import numpy as np
from sklearn.model_selection import LeaveOneOut
X = np.array([1, 2, 3, 4])
y = np.array([5,6,7,8])
loo = LeaveOneOut()
loo.get_n_splits(X)
print(loo)
LeaveOneOut()
for train_index, test_index in loo.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    print('X_train \n' , X_train)
    print('X_test \n' , X_test)
    print('y_train \n' ,y_train)
    print('y_test \n' , y_test)
```

```
    print('**********')
```

مثال اخر

```
from sklearn.model_selection import LeaveOneOut
X = [1, 2, 3, 4]
loo = LeaveOneOut()
for train, test in loo.split(X):
    print("%s %s" % (train, test))
```

1.6.7) Leave P Out

يترك عدد من العناصر للاختبار و الباقي للتدريب

```
import numpy as np
from sklearn.model_selection import LeavePOut
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([1, 2, 3, 4])
#lpo = LeavePOut(1)
#lpo = LeavePOut(2)
lpo = LeavePOut(3)

lpo.get_n_splits(X)

print(lpo)
LeavePOut(p=2)
for train_index, test_index in lpo.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
```

```
    y_train, y_test = y[train_index], y[test_index]
    print('X_train \n' , X_train)
    print('X_test \n' , X_test)
    print('y_train \n' ,y_train)
    print('y_test \n' , y_test)
    print('**********************')
```

مثال اخر

```
import numpy as np
from sklearn.model_selection import LeavePOut
X = np.ones(10)
lpo = LeavePOut(p=3)
for train, test in lpo.split(X):
    print("%s %s" % (train, test))
```

1.6.8) Shuffle Split

يقوم بعمل اختيار عشوائي للتدريب و الاختبار حسب النسبة المعطاة

```
import numpy as np
from sklearn.model_selection import ShuffleSplit
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [3, 4], [5, 6]])
y = np.array([1, 2, 1, 2, 1, 2])
rs = ShuffleSplit(n_splits=5, test_size=.1, random_state=0)
rs.get_n_splits(X)

print(rs)

for train_index, test_index in rs.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)

rs = ShuffleSplit(n_splits=5, train_size=0.5, test_size=.25,random_state=0)
for train_index, test_index in rs.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
```

مثال اخر

```
import numpy as np
from sklearn.model_selection import ShuffleSplit
X = np.arange(10)

n=0.1
#n=0.3
#n=0.5
#n=0.7
#n=0.9

ss = ShuffleSplit(n_splits=5, test_size=n,random_state=0)
for train_index, test_index in ss.split(X):
    print("%s %s" % (train_index, test_index))
```

1.6.9) Stratified Shuffle Split

نفس الفكرة السابقة لكن مع مراعاة توزيع النسب

```
 import numpy as np
from sklearn.model_selection import StratifiedShuffleSplit
X = np.array([[1, 2], [3, 4], [1, 2], [3, 4], [1, 2], [3, 4]])
y = np.array([0, 0, 0, 1, 1, 1])
sss = StratifiedShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
sss.get_n_splits(X, y)
print(sss)
for train_index, test_index in sss.split(X, y):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    print('X_train \n' , X_train)
    print('X_test \n' , X_test)
    print('y_train \n' ,y_train)
    print('y_test \n' , y_test)
    print('**********************')
```

1.6.10) Time Series Split

تعتمد فكرة عنصر تدريب و اخر اختبار , ثم عنصرين تدريب وواحد اختبار , ثم ثلاثة تدريب وواحد اختبار و هكذا

```
 import numpy as np
from sklearn.model_selection import TimeSeriesSplit
X = np.array([[1, 2], [3, 4], [1, 2], [3, 4], [1, 2], [3, 4]])
y = np.array([1, 2, 3, 4, 5, 6])
tscv = TimeSeriesSplit(n_splits=5)
print(tscv)
for train_index, test_index in tscv.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    print('X_train \n' , X_train)
    print('X_test \n' , X_test)
    print('y_train \n' ,y_train)
    print('y_test \n' , y_test)
    print('**********************')
```