Section3: Sklearn

SCIKIT LEARN

Metrics Module : Regression

- **Metrics Module** : quantifying the quality of predictions

1. metrics.mean_absolute_error
2. metrics.mean_squared_error

1. mean_absolute_error

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

```
from sklearn.metrics import mean_absolute_error

y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]


MAE=mean_absolute_error(y_true, y_pred, multioutput='uniform_average') # 0.5
# uniform_average is the default value
print(MAE)
```

1. mean_absolute_error

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

```
from sklearn.metrics import mean_absolute_error

y_true = [[0.5, 1], [-1, 1], [7, -6]]
y_pred = [[0, 2], [-1, 2], [8, -5]]


MAE=mean_absolute_error(y_true, y_pred, multioutput='uniform_average') # 0.75
# uniform_average is the default value
print(MAE)
```

1. mean_absolute_error

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

```
from sklearn.metrics import mean_absolute_error

y_true = [[0.5, 1], [-1, 1], [7, -6]]
y_pred = [[0, 2], [-1, 2], [8, -5]]


MAE=mean_absolute_error(y_true, y_pred, multioutput='raw_values') # [0.5 1.0] each column
# uniform_average is the default value
print(MAE)
```

2. Mean Squared Error

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

```
from sklearn.metrics import mean_squared_error
y_true1 = [3, -0.5, 2, 7]
y_pred1 = [2.5, 0.0, 2, 8]
MAE1=mean_squared_error(y_true1, y_pred1, multioutput='uniform_average')

y_true2 = [[0.5, 1],[-1, 1],[7, -6]]
y_pred2 = [[0, 2],[-1, 2],[8, -5]]

MAE2=mean_squared_error(y_true2, y_pred2, multioutput='raw_values')

print(MAE1)
print(MAE2)
```

Metrics Module : classification
1. Confusion Matrix

- **Confusion Matrix:** A confusion matrix is a table that is used to define the performance of a classification algorithm.

TN	FP
FN	**TP**

- **Example**

positive	120	750	20
negative	880	130	100

Metrics Module : classification
1. Confusion Matrix

```
from sklearn.metrics import confusion_matrix
import numpy as np
y_true  =['1','0','0','1','0','1','1','0','1','0']
y_pred = ['1','1','0','0','1','0','1','1','1','1']

cm=confusion_matrix(y_true, y_pred)
print(cm)
```

Metrics Module : classification
1. Confusion Matrix

```
from sklearn.metrics import confusion_matrix

y_true  = ['a','b','b','a','b','a','a','b','a','b']
y_pred = ['a','a','b','b','a','b','a','a','a','a']

cm=confusion_matrix(y_true, y_pred)
print(cm)
```

-	pred a	pred b
actual a	3 (**TN**)	2(FP)
actual b	4(FN)	1 (**TP**)

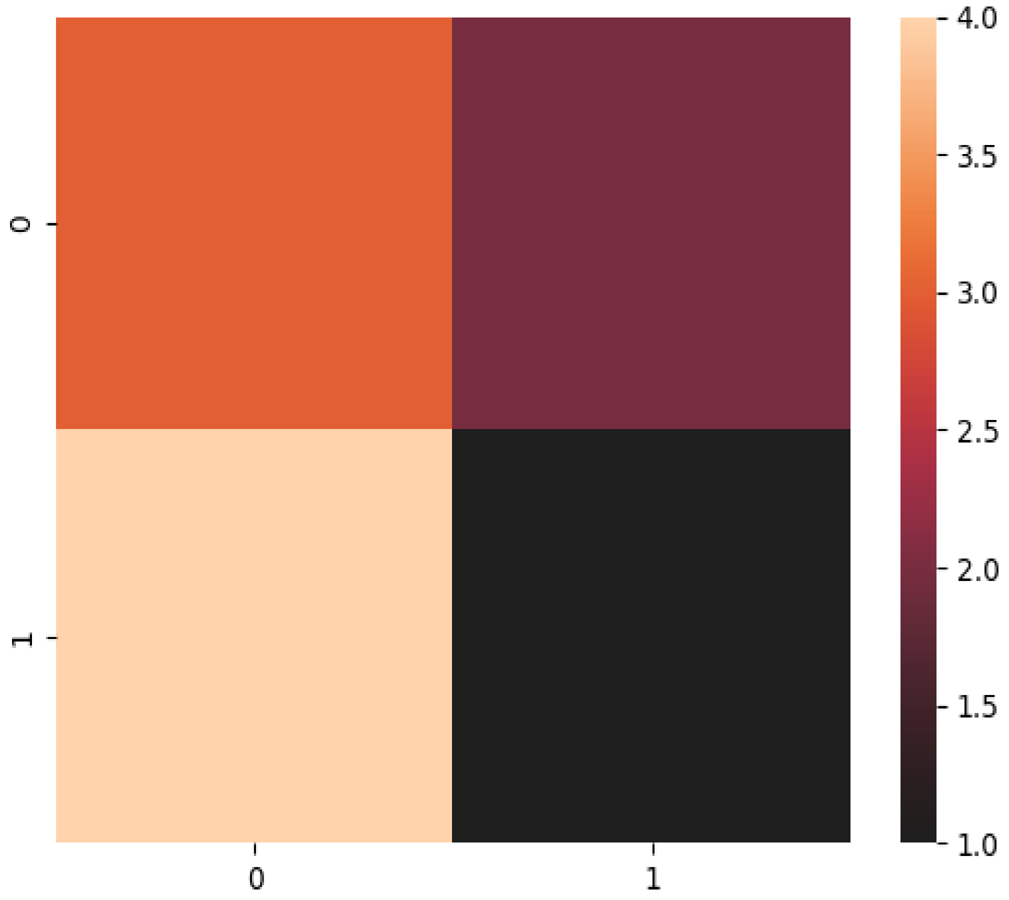
Metrics Module : classification
1. Confusion Matrix # Drawing

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

y_true  = ['a','b','b','a','b','a','a','b','a','b']
y_pred = ['a','a','b','b','a','b','a','a','a','a']

cm=confusion_matrix(y_true, y_pred)
print(cm)

# drawing
sns.heatmap(cm,center=True)
plt.show()
```



Metrics Module : classification
1. Confusion Matrix

```
from sklearn.metrics import confusion_matrix

y_true =  ['a','a','b','b','a','b','c','c','b','b']
y_pred = ['a','b','c','a','b','c','a','b','c','a']

cm=confusion_matrix(y_true, y_pred)
print(cm)
```

-	pred a	pred b	Pred c
actual a	1	2	0
actual b	2	0	3
actual c	1	1	0

Metrics Module : classification
2. Accuracy Score

```
from sklearn.metrics import accuracy_score

y_true  =['1','0','0','1','0','1','1','0','1','0']
y_pred = ['1','1','0','0','1','0','1','1','1','1']

#Calculating Accuracy Score  : ((TP + TN) / float(TP + TN + FP + FN))
AccScore = accuracy_score(y_true, y_pred, normalize=True)
#normalize=false will give only (TP + TN)
#normalize=true will give ((TP + TN) / float(TP + TN + FP + FN))
print('Accuracy Score is : ', AccScore)
```

-	pred 0	pred 1
actual 0	1 (**TN**)	4(FP)
actual 1	2(FN)	3(**TP**)

Metrics Module : classification
3. Recall Score (Sensitivity)

```python
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
#---------------------------------------------------------
y_true  =['1','0','0','1','0','1','1','0','1','0']
y_pred = ['1','1','0','0','1','0','1','1','1','1']
#---------------------------------------------------------
CM = confusion_matrix(y_true, y_pred)
print('Confusion Matrix is : \n', CM)
#------------------------------------------------------------
#Calculating Recall Score : (Sensitivity) (TP / float(TP + FN))
RecallScore = recall_score(y_true, y_pred,pos_label='1', average='binary')
#it can be : binary,macro,weighted,samples
print('Recall Score is : ', RecallScore)
```

Recall score is used to measure the model performance in terms of measuring the count of true positives in a correct manner out of all the actual positive values.

-	pred 0	pred 1
actual 0	1 (**TN**)	4(FP)
actual 1	2(FN)	3(**TP**)

Metrics Module : classification
4. Precision Score (Specificity)

```
from sklearn.metrics import precision_score
from sklearn.metrics import confusion_matrix
#----------------------------------------------------
y_true  =['1','0','0','1','0','1','1','0','1','0']
y_pred = ['1','1','0','0','1','0','1','1','1','1']
#----------------------------------------------------
CM = confusion_matrix(y_true, y_pred)
print('Confusion Matrix is : \n', CM)
#-------------------------------------------------------
#Calculating Precision Score : (Specificity) #(TP / float(TP + FP))
PrecisionScore = precision_score(y_true, y_pred,pos_label='1', average='binary')
#it can be : binary,macro,weighted,samples
print('Precision Score is : ', PrecisionScore)
```

Precision score is used to measure the model performance in terms of measuring the count of true positives in a correct manner out of all positive predictions.

-	pred 0	pred 1
actual 0	1 (**TN**)	4(FP)
actual 1	2(FN)	3(**TP**)

Metrics Module : classification
5. F1 Score

```
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
#----------------------------------------------------
y_true  =['1','0','0','1','0','1','1','0','1','0']
y_pred = ['1','1','0','0','1','0','1','1','1','1']
#----------------------------------------------------
CM = confusion_matrix(y_true, y_pred)
print('Confusion Matrix is : \n', CM)
#------------------------------------------------------
#Calculating F1 Score  : 2 * (precision * recall) / (precision + recall)
F1Score = f1_score(y_true, y_pred,pos_label='1', average='binary')
#it can be : binary,macro,weighted,samples
print('F1 Score is : ', F1Score)
```

F-score is the harmonic mean of precision and recall

-	pred 0	pred 1
actual 0	1 (**TN**)	4(FP)
actual 1	2(FN)	3(**TP**)