project 03

April 2, 2020

[47]:
```
#Suja Basnet
#project 03

from sklearn.datasets import load_iris
from sklearn.model_selection import KFold
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

[48]:
```
iris_data = load_iris()
```

[49]:
```
data_input = iris_data.data
data_output = iris_data.target
print(data_output)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

[50]:
```
vec = np.arange(0,15)
kf = KFold(n_splits =10, shuffle=True)
print("Train Set                                Test Set ")
for train_set,test_set in kf.split(vec):
    print(train_set, test_set)
```

```
Train Set                                Test Set
[ 0  1  3  4  5  6  7  8 10 11 12 13 14] [2 9]
[ 0  1  2  3  4  5  6  7  8  9 10 12 14] [11 13]
[ 0  1  2  4  5  6  7  8  9 11 12 13 14] [ 3 10]
[ 1  2  3  4  5  6  7  9 10 11 12 13 14] [0 8]
[ 0  1  2  3  5  6  7  8  9 10 11 12 13] [ 4 14]
```

```
[ 0  1  2  3  4  5  7  8  9 10 11 12 13 14] [6]
[ 0  1  2  3  4  6  7  8  9 10 11 12 13 14] [5]
[ 0  1  2  3  4  5  6  7  8  9 10 11 13 14] [12]
[ 0  1  2  3  4  5  6  8  9 10 11 12 13 14] [7]
[ 0  2  3  4  5  6  7  8  9 10 11 12 13 14] [1]
```

[51]:
```
#Split the data into 80% training and 20% testing
x_train, x_test, y_train, y_test = train_test_split(data_input, data_output,
test_size=0.2, random_state=40)
```

[52]:
```
#Train the model
model = LogisticRegression()
model.fit(x_train, y_train) #Training the model
#Test the model
predictions = model.predict(x_test)
print(predictions)# printing predictions
print()# Printing new line
#Check precision, recall, f1-score
print( classification_report(y_test, predictions) )
print( accuracy_score(y_test, predictions))
# import warnings filter
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
import warnings
warnings.filterwarnings('ignore')
```

```
[0 1 2 2 1 2 1 1 1 0 1 0 0 2 1 2 2 2 1 1 2 2 1 0 1 0 0 2 0 1]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00        12
           2       1.00      1.00      1.00        10

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

1.0
```

[53]:
```
from sklearn.metrics import make_scorer, accuracy_score,␣
↪precision_score,recall_score, f1_score
2
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import cross_validate
```

```
kf2 = KFold(n_splits=10, shuffle=True, random_state=52)
scoring = {'acc': 'accuracy',
'prec_macro': 'precision_macro',
'rec_micro': 'recall_macro',
}
scores = cross_validate(model, data_input, data_output, scoring=scoring,cv=kf2,␣
 ↪return_train_score=True)
print(scores['test_acc'])
print("Mean accuracy: ", cross_val_score(model, data_input, data_output,cv=kf2,␣
 ↪scoring='accuracy').mean())
print(scores['test_prec_macro'])
print("Mean precision: ", cross_val_score(model, data_input,␣
 ↪data_output,cv=kf2, scoring='precision_macro').mean())
print(scores['test_rec_micro'])
print("Mean Recall: ", cross_val_score(model, data_input, data_output,␣
 ↪cv=kf2,scoring='recall_macro').mean())
print("F1score:
 ↪",cross_val_score(model,data_input,data_output,scoring=make_scorer(f1_score,average='weight
 ↪labels=[2]),cv=kf2))
#precision = cross_val_score(model,data_input, data_output,scoring='precision',␣
 ↪cv=kf2)
#print ("Precision: " + str(round(100*precision.mean(), 2)) + "%")
#print("Precision: ", cross_val_score(model, data_input,␣
 ↪data_output,cv=kf2,scoring='precision'))
```

```
[0.93333333 1.         0.93333333 1.         0.93333333 0.86666667
 1.         1.         0.93333333 1.        ]
Mean accuracy:  0.96
[0.96296296 1.         0.91666667 1.         0.88888889 0.88571429
 1.         1.         0.94444444 1.        ]
Mean precision:  0.9598677248677248
[0.88888889 1.         0.95833333 1.         0.95833333 0.88571429
 1.         1.         0.93333333 1.        ]
Mean Recall:  0.9624603174603175
F1score: [0.94117647 1.         0.85714286 1.         0.93333333 0.8
 1.         1.         0.88888889 1.        ]
```