ERCİYES ÜNİVERSİTESİ
COMPUTER ENGINEERING

INTRODUCTION TO PATTERN RECOGNITION

MIDTERM PROJECT

STUDENT: Gaye Armut - 1030510037

LECTURER :Dr. Öğr. Üyesi Özkan Ufuk Nalbantoğlu

PROJECT :Forensics investigation from fingerprint microbes:

- In this project i used the Gaussian Naive Bayes Algorithm

Plugins that we need to use:

```
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, auc
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import auc
from sklearn import metrics
import sklearn
```

We need to read to the file and load the data:

```
#load the dataset
data = pd.read_csv('otu.csv')
data = data.T
```

We converted left / right (string) to 0/1 (binary):

```
#convert left/right to 0/1
le = LabelEncoder()
data[0] = le.fit_transform(data[0])
print(data.head(5))
```

Split the data using «iloc»:

```
#split dataset into feature target
X = data.iloc[:, 1:3033]
y = data.iloc[:, 0]
```

We need to training the data:

```
#training dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=0)
```

We need to training the model (Gaussian Naive Bayes):

```
#train Gaussian naive bayes
model = GaussianNB()
model.fit(X_train, y_train)
```

Cross- validation:

```
#use cross validation
scores = cross_val_score(model, X_test, y_test, scoring='accuracy', cv=2)
print('Cross-validated scores:', scores)
```

Calculate the accuracy:

```
#accuracy
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print('Accuracy:', accuracy)
```

Calculate the AUC (Area under the ROC curve):

```
#Area under the ROC curve.
roc_auc = roc_auc_score(y_test, predictions)
print('Auc:', roc_auc)
```

Confusion matrix:

```
#confusion matrix
confusion = confusion_matrix(y_test, predictions)
print('Confusion matrix: ')
print(confusion)
```

Calculate the sensivity and specificity:

```
#sensivity and specificity
sensitivity = confusion[1, 1] / (confusion[1, 0] + confusion[1, 1])
specificity = confusion[0, 0] / (confusion[0, 0] + confusion[0, 1])
print('   Sensitivity:', sensitivity)
print('   Specificity:', specificity)
```

Here is the all code:

```
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, auc
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import auc
from sklearn import metrics
import sklearn

# load the dataset
data = pd.read_csv('otu.csv')
data = data.T

# convert left/right to 0/1
le = LabelEncoder()
data[0] = le.fit_transform(data[0])
print(data.head(5))

# split dataset into feature target
X = data.iloc[:, 1:3033]
y = data.iloc[:, 0]

# training dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=0)

# train Gaussian naive bayes
model = GaussianNB()
model.fit(X_train, y_train)

# use cross validation
scores = cross_val_score(model, X_test, y_test, scoring='accuracy', cv=2)
print('Cross-validated scores:', scores)

# accuracy
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print('Accuracy:', accuracy)

# Area under the ROC curve.
roc_auc = roc_auc_score(y_test, predictions)
print('Auc:', roc_auc)

# confusion matrix
confusion = confusion_matrix(y_test, predictions)
print('Confusion matrix: ')
print(confusion)

#sensivity and specificity
sensitivity = confusion[1, 1] / (confusion[1, 0] + confusion[1, 1])
specificity = confusion[0, 0] / (confusion[0, 0] + confusion[0, 1])
print('   Sensitivity:', sensitivity)
print('   Specificity:', specificity)

```

Here is the output:

```
PS C:\Users\Gaye\Desktop\pattern> & C:/Users/Gaye/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Gaye/Desktop/pattern/Pattern_Rec.py
c:\Users\Gaye\Desktop\pattern\Pattern_Rec.py:11: DtypeWarning: Columns (0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,4
3,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119
5,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,182
8,199,200,201,202,203,204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245
1,262,263,264,265,266,267,268,269,270) have mixed types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv('otu.csv')
         0       1       2       3       4       5       6       7       8       9    10   11   12        13   14   15   16   17   18    ... 3284 3285 3286 3287 3288 3289 3290 3291
Sample1     0  0.33364  0.33364  0.33364  0.33364  0.33364  0.33364  0.33364  0.33364       0    0    0    0  99.66636    0    0    0    0    0  ...    0    0    0    0    0    0    0    0
Sample2     0  0.62579  0.62579  0.62579  0.62579  0.62579  0.62579  0.49776  0.49776  0.12802    0    0    0  99.37421    0    0    0    0    0  ...    0    0    0    0    0    0    0    0
Sample3     0        0        0        0        0        0        0        0        0        0    0    0    0       100    0    0    0    0    0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
Sample4     0        0        0        0        0        0        0        0        0        0    0    0    0       100    0    0    0    0    0  ...    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0
Sample5     0  0.56233  0.56233  0.56233  0.56233  0.56233  0.56233  0.49446  0.49446  0.06786    0    0    0  99.43767    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

[5 rows x 3303 columns]
Cross-validated scores: [0.60714286 0.59259259]
Accuracy: 0.6545454545454545
Auc: 0.6439393939393939
Confusion matrix:
[[13  9]
 [10 23]]
   Sensitivity: 0.696969696969697
   Specificity: 0.5909090909090909
PS C:\Users\Gaye\Desktop\pattern>
```

Here is the SVC(Support Vector Machines) Model’s result :

```
# train svc model
model = SVC()
model.fit(X_train, y_train)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

9,260,261,262,263,264,265,266,267,268,269,270) have mixed types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv('otu.csv')
         0       1       2       3       4       5       6       7       8       9   10   11   12       13   14   ...
3302
Sample1     0  0.33364  0.33364  0.33364  0.33364  0.33364  0.33364  0.33364  0.33364       0    0    0    0  99.66636    0  ...
   0
Sample2     0  0.62579  0.62579  0.62579  0.62579  0.62579  0.62579  0.49776  0.49776  0.12802    0    0    0  99.37421    0  ...
   0
Sample3     0        0        0        0        0        0        0        0        0        0    0    0    0       100    0  ...
 0.0
Sample4     0        0        0        0        0        0        0        0        0        0    0    0    0       100    0  ...
   0
Sample5     0  0.56233  0.56233  0.56233  0.56233  0.56233  0.56233  0.49446  0.49446  0.06786    0    0    0  99.43767    0  ...
 0.0

[5 rows x 3303 columns]
Cross-validated scores: [0.60714286 0.62962963]
Accuracy: 0.6181818181818182
Auc: 0.6439393939393939
Confusion matrix:
[[17  5]
 [16 17]]
   Sensitivity: 0.5151515151515151
   Specificity: 0.7727272727272727
PS C:\Users\Gaye\Desktop\pattern>
```

Here is the KNN(K-Nearest Neighbors) Model’s result :

```
# train K-Nearest Neighbors model
model = KNeighborsClassifier()
model.fit(X_train, y_train)
```

```
  data = pd.read_csv('otu.csv')
         0       1       2       3       4       5       6       7       8       9    10   11   12       13   14    ... 3288 3289 3290 3291 3292 3293 3294 329
3302
Sample1     0  0.33364  0.33364  0.33364  0.33364  0.33364  0.33364  0.33364  0.33364       0    0    0    0  99.66636    0  ...    0    0    0    0    0    0    0
   0
Sample2     0  0.62579  0.62579  0.62579  0.62579  0.62579  0.62579  0.49776  0.49776  0.12802    0    0    0  99.37421    0  ...    0    0    0    0    0    0    0
   0
Sample3     0        0        0        0        0        0        0        0        0        0    0    0    0       100    0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.
 0.0
Sample4     0        0        0        0        0        0        0        0        0        0    0    0    0       100    0  ...    0    0    0    0    0    0    0
   0
Sample5     0  0.56233  0.56233  0.56233  0.56233  0.56233  0.56233  0.49446  0.49446  0.06786    0    0    0  99.43767    0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.
 0.0

[5 rows x 3303 columns]
Cross-validated scores: [0.64285714 0.62962963]
Accuracy: 0.5818181818181818
Auc: 0.5984848484848485
Confusion matrix:
[[15  7]
 [16 17]]
   Sensitivity: 0.5151515151515151
   Specificity: 0.6818181818181818
PS C:\Users\Gaye\Desktop\pattern>
```