2019103555 PRANAVA RAMAN B M S 01/11/2021

LAB - 09 - EXECUTION - NAIVE BAYES AND KNN

1. NAIVE BAYES ALGORITHM

CODE:-

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import math

df = pd.read_csv('computer.csv')

le = LabelEncoder()

dataset = df.apply(le.fit_transform)

X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

dataset
```

[1] ✓ 3.3s Python

	age	income	student	credit_rating	buys_computer
0	1	0	0	1	0
1	1	0	0	0	0
2	0	0	0	1	1
3	2	2	0	1	1
4	2	1	1	1	1
5	2	1	1	0	0
6	0	1	1	0	1
7	1	2	0	1	0
8	1	1	1	1	1
9	2	2	1	1	1
10	1	2	1	0	1
11	0	2	0	0	1
12	0	0	1	1	1
13	2	2	0	0	0

```
def groupUnderClass(mydata):
    dict = {}
    for i in range(len(mydata)):
        if (mydata.iloc[i, -1] not in dict):
            dict[mydata.iloc[i, -1]] = []
        dict[mydata.iloc[i, -1]].append(mydata.iloc[i, :])
    return dict

def mean(numbers):
    return sum(numbers) / float(len(numbers))

def std_dev(numbers):
    avg = mean(numbers)
    variance = sum([pow(x - avg, 2) for x in numbers]) / float(len(numbers) - 1)
    return math.sqrt(variance)

def MeanAndStdDev(mydata):
    info = [(mean(attribute), std_dev(attribute)) for attribute in zip(*mydata)]
    del info[-1]
    return info

def MeanAndStdDevForClass(mydata):
    info = {}
    dict = groupUnderClass(mydata)
    for classValue, instances in dict.items():
        info[classValue] = MeanAndStdDev(instances)
    return info
```

```
def calculateGaussianProbability(x, mean, stdev):
    expo = math.exp(-(math.pow(x - mean, 2) / (2 * math.pow(stdev, 2))))
    return (1 / (math.sqrt(2 * math.pi) * stdev)) * expo

def calculateClassProbabilities(info, test):
    probabilities = {}
    for classValue, classSummaries in info.items():
        probabilities[classValue] = 1
        for i in range(len(classSummaries)):
            mean, std_dev = classSummaries[i]
            x = test[i]
            probabilities[classValue] *= calculateGaussianProbability(x, mean, std_dev)
    return probabilities
```

```
def predict(info, test):
    probabilities = calculateClassProbabilities(info, test)
    bestLabel, bestProb = None, -1
    for classValue, probability in probabilities.items():
        if bestLabel is None or probability > bestProb:
            bestProb = probability
            bestLabel = classValue
    return bestLabel

def getPredictions(info, test):
    predictions = []
    for i in range(len(test)):
        result = predict(info, test.iloc[i, :])
        predictions.append(result)
    return predictions

def accuracy_rate(test, predictions):
    correct = 0
    for i in range(len(test)):
        if test.iloc[i] == predictions[i]:
            correct += 1
    return (correct / float(len(test))) * 100.0
```

[2] ✓ 0.3s Python

```
info = MeanAndStdDevForClass(dataset)
predictions = getPredictions(info, X)

accuracy = accuracy_rate(y, predictions)
print("Accuracy of Naive Bayes Model is: ", accuracy)
```

[33] ✓ 0.1s Python

```
Accuracy of Naive Bayes Model is:  85.71428571428571
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

y_true = y
y_pred = predictions
print('Confusion Matrix: \n', confusion_matrix(y_true, y_pred))

tp, fn, fp, tn = confusion_matrix(y_true,y_pred,labels=[0,1]).reshape(-1)
print('\nOutcome values : \n', tp, fn, fp, tn)

matrix = classification_report(y_true,y_pred,labels=[0,1])
print('\nClassification report : \n',matrix)
```

[39] ✓ 0.6s Python

```
Confusion Matrix:
 [[4 1]
 [1 8]]

Outcome values :
 4 1 1 8

Classification report :
              precision    recall  f1-score   support

           0       0.80      0.80      0.80         5
           1       0.89      0.89      0.89         9

    accuracy                           0.86        14
   macro avg       0.84      0.84      0.84        14
weighted avg       0.86      0.86      0.86        14
```

```
print('For the Data Instance X = (age ⩽30,Income = medium,Student = yes,Credit_rating = fair)\n')
X_test = pd.DataFrame([[0, 2, 1, 1]])

predictions = getPredictions(info, X_test)

if predictions[0] == 1:
    print('Prediction is: Yes the student will buy computer')
else:
    print('Prediction is: No the student will not buy computer')
```

[45] ✓ 0.5s Python

```
For the Data Instance X = (age ⩽30,Income = medium,Student = yes,Credit_rating = fair)

Prediction is: Yes the student will buy computer
```

+ Code + Markdown

2. KNN - CLASSIFIER

```
import numpy as nm
import matplotlib.pyplot as mtp
from sklearn.preprocessing import LabelEncoder
import math
import pandas as pd

le = LabelEncoder()
df= pd.read_csv('computer.csv')

data_set = df.apply(le.fit_transform)
x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 4].values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.3)

from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

[85] ✓ 0.9s Python

```
x_train
```

[86] ✓ 0.9s Python

```
array([[ 1.01600102,  1.22474487, -1.11803399,  0.89442719],
       [ 1.01600102,  0.        ,  0.89442719,  0.89442719],
       [-0.12700013, -1.22474487, -1.11803399,  0.89442719],
       [-1.27000127, -1.22474487, -1.11803399,  0.89442719],
       [-0.12700013,  1.22474487,  0.89442719, -1.11803399],
       [-1.27000127, -1.22474487,  0.89442719,  0.89442719],
       [ 1.01600102,  1.22474487, -1.11803399, -1.11803399],
       [ 1.01600102,  0.        ,  0.89442719, -1.11803399],
       [-1.27000127,  0.        ,  0.89442719, -1.11803399]])
```

```
x_test
```

[87] ✓ 0.1s Python

```
array([[-0.12700013,  0.        ,  0.89442719,  0.89442719],
       [ 1.01600102,  1.22474487,  0.89442719,  0.89442719],
       [-0.12700013, -1.22474487, -1.11803399, -1.11803399],
       [-1.27000127,  1.22474487, -1.11803399, -1.11803399],
       [-0.12700013,  1.22474487, -1.11803399,  0.89442719]])
```

```
y_train
```

[88] ✓ 0.1s Python

```
array([1, 1, 0, 1, 1, 1, 0, 0, 1])
```

```
y_test
```

[89] ✓ 0.1s Python

```
array([1, 1, 0, 1, 0])
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=4, metric='minkowski', p=2 )
classifier.fit(x_train, y_train)
```

[90] ✓ 0.1s Python

```
KNeighborsClassifier(n_neighbors=4)
```

```
y_pred= classifier.predict(x_test)
```

[91] ✓ 0.2s Python

```
y_pred
```

[92] ✓ 0.2s Python

```
array([1, 1, 0, 1, 0])
```

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
```
[93] ✓ 0.1s Python

```
cm
```
[94] ✓ 0.5s Python

```
array([[2, 0],
       [0, 3]], dtype=int64)
```

```
pred = classifier.predict([[0, 2, 1, 1]])
pred
```
[95] ✓ 0.1s Python

```
array([1])
```

```
if pred[0] == 1:
    print('Prediction is: Yes the student will buy computer')
else:
    print('Prediction is: No the student will not buy computer')
```
[97] ✓ 0.5s Python

```
Prediction is: Yes the student will buy computer
```

```
from sklearn.metrics import classification_report
matrix = classification_report(y_test,y_pred,labels=[0,1])
print('\nClassification report : \n',matrix)
```
[96] ✓ 0.1s Python

```
Classification report :
               precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         3

    accuracy                           1.00         5
   macro avg       1.00      1.00      1.00         5
weighted avg       1.00      1.00      1.00         5
```