Experiment- 5

Write a program to implement the naive Bayer's classifier for a sample training data set stored as csv file, compute the accuracy of the classifier considering few data sets.

```
from sklearn.datasets import load_iris
 from sklearn.model_selection import train_test_split
 from sklearn.naive_bayes import Gaussian NB
 x, Y = load_iris(return_x_y = True)
 x_train, X_test, Y_train, Y_test = train_test_split(x, y,
  test_size = 0.4, random_state = 0)
  gnb = Gaussian NB()
  Y_pred = gnb.fit(x_train, y_train).predict(X_test)
  print("Number of mislabelled point out of total %d points: %d"
        (x_test.shape(0), (y_test != y_pred).sum(1))

 from sklearn.metrics import confusion_matrix
  (confusion_matrix(y_test, y_pred)
  accuracy = (56/60) * 100
  accuracy.
```

Output:-

```
Number of mislabelled points out of a total 60 points
4
array ([[16, 0, 0],
        [ 0, 23, 0],
        [ 0, 4, 17]])

 93. 333333
```

Experiment-6

Assuming a set of documents that need to be classified use the naive bayerian classifier model to perform the task · calculate the accuracy precision and recall for your dataset.

```
from sklearn import metrics
 print ('Accuracy metrics')
 print (' Accuracy of the classifier is ' metrics. accuracy')
 score (y-test, y-pred)*100)
  print ('confusion matrix')
  print (metrics. confusion -matrix (y-test, y-pred)
  point ('Recall and Precission')
  print( metrices. Precission -score (y-test, y-pred,
                                     avg = Naive)
```

Output:

```
Accuracy metrics
 Accuracy of the classifier is 93.33333
  confusion metrics
     [[16, 0, 0]
      [0, 23, 0]
      [0, 4, 17]]

 Recall and Precision
    [1.        1.        0.80952381]
    [1.        0.85185705        1.]
```

Experiment- 7

WAP to implement KNN algorithm to classify the Iris dataset. print both correct and wrong prediction.

```
from sklearn . neighbours import KNeighbours classifier
from sklearn. matrics import classification report
from sklearn. metrics import accuracy_score
classifier = KNeighbours classifier (n_neighbours = 2,
 p=3, metric =' euliedian')

classifier . fit (x_train, y_train)
y_pred = classifier . predict (x_test)
cm = confusion_matrixe (y_test, y_pred)
print (' confusion matrix is as follows (n, 'ine)
print ('Accuracy matrics')
print ('correct prediction," accuracy_score (y_test, y_pred
print ("wrong prediction", (1- accuracy_score (y_test,
                                                y_pred))
```

29/5/23

Output:-

Confusion matrix is as follows:

```
[[16 0 0]
 [0 12 0]
 [0 3 18]]
```

Accuracy metris

```
             Precission   recall   f1 score   support
0               1.00       1.00     1.00        16
1               0.88       0.96     0.92        23
2               0.95       0.86     0.90        21

accuracy                            0.93        60
macro avg       0.94       0.94     0.94        60
weight avg      0.94       0.93     0.93        60
```

```
correct prediction    0.9333333
wrong prediction      0.66666665
```

Experiment - 8

WAP to construct a Bayesian network considering medical data use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data set.

```
import numpy as np
from urllib.request import urlopen
 import urllib
 import pandas as pd
from pgmpy import variable Elimination
 names =['age','sex','cp','test bps', 'chol', 'fbs','restecg','thalach
          'exang','old peak','slope','ca', 'thal', heartdisease']
heart Diseas = pd.read.csv('heart.csv', names = names)
heart Diseas = heart Diseas. replace ('?'. np.nan)
 model = Bayesian model(('age','trest bps'), ('age', 'fbs'), ('sex',
          'testbps'), ('exang','testbps'), ('resotphs','heartdisease
          ('sbs','heart disease'), ('heartdiseas','rest ecg')])
model.fit (heart Disease, estimate = maximum likelihood Estimator)
from pgmpy. inference import variable Elimination
 Heart Disease _infer = variable Elimination (model)
 q = Heart Disease_infer. query (variables = ['heart Disease'],
          evidence = {'age': 37,'sex':0})
 print (q ['heart disease'])
```

heart Disease	phil (heart Disease)
heart Diseases-0	0·5593
heart Diseases-1	0·1107

Experiment-9

Apply EM algorithm to cluster a set of data stored in a CSV file. Use the some data set for clustering using k-means algorithm compare the results of these two algorithms and comment on the quality of clustering. ~~You can a~~

```
import numpy as np
from sklearn.cluster import Kmeans.
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture.
import pandas as pd
x = pd.read_csv("Kmeans data.csv")
x1 = x['Distance_Feature'].values
x2 = x['Speeding_Feature'].values
x = np.array(list(zip(x1, x2))).reshape(len(x1),2)
plt.plot()
plt.xlim([0,100])
plt.ylim([0,50])
plt.title('Dataset')
plt.scatter(x1, x2)
plt.show()
# code for EM
gmm = Gaussian mixture(n_components=3)
gmm.fit(x)
em_predictions = gmm.predict(x)
print("\nEM predictions)
```

POORNIMA

```
print ("mean:\n", gmm.means_)
print ('\n')
print ("covariances \n", gmm.covariances_)
print (x)
plt.title ('Expectation maximum')
plt.scatter (x[:,0], x[:,1], c=em_predictions, s=50
plt.show()
# code for kmeans.
import matplotlib.pyplot as plt1
kmeans= kmeans (n_cluster=3)
kmeans.fit(x)
print (kmeans.cluster_centers_)
print (kmeans.labels_)
plt.title ('KMEANS')
plt1.scatter (X[:,0], X[:,1], c=kmeans.labels_, cmap=
                                                'rainbow')
plt1.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster
                                                [:,1], color='black
```