POORNIMA

Experiment- 8

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

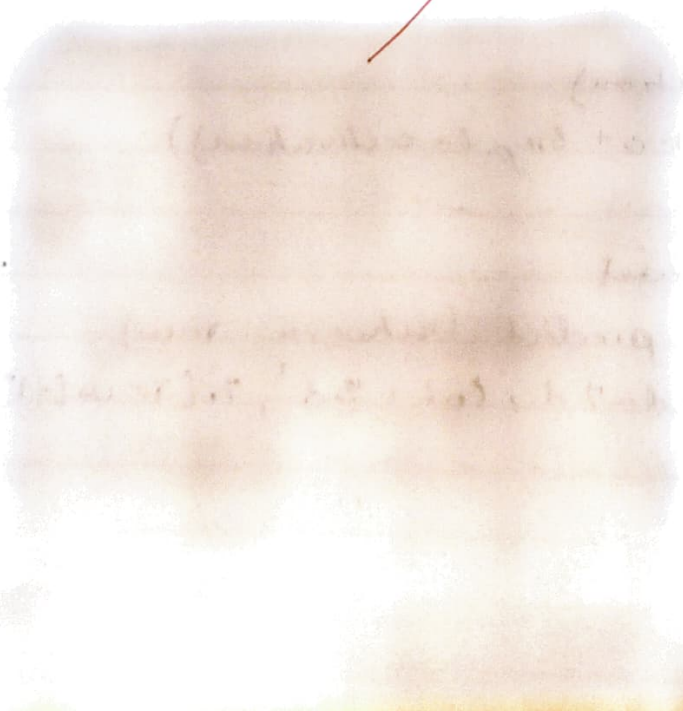
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
X, Y = load_iris(return_X_y = True)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size = 0.4, random_state = 0)
gnb = GaussianNB()
Y_pred = gnb.fit(X_train, 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()))

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, 1, 17]])

93. 333333



POORNIMA

Experiment-6

Assuming a set of documents that need to be classified use the naive bayerian classifier model to perform this 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))
print ('Recall and Precision')
print( metrics.Precision_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]]

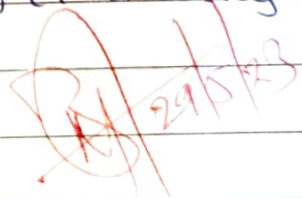
Recall and Precision
[1. 1. 0.80952381]
[1. 0.85185185 1.]

POORNIMA

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.metrics 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_matrix(y_test, y_pred)
print('confusion matrix is as follows (n, 'ne)
print('Accuracy metrics')
print('correct prediction,' accuracy_score(y_test, y_pred
print("wrong prediction", (1- accuracy_score(y_test,
                                    y_pred))
```



Output :-

Confusion matrix is as follows:

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

Accuracy matrix

	Precision	recall	f1 score	support
0	1.00	1.00	1.00	16
1	0.80	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 0.9333 333

wrong prediction 0.6666665

POORNIMA

Experiment - 8

WAP to construct a Bayesian network considering medical data case 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 Disease = pd.read_csv('heart.csv', names = names)
heart Disease = heart Disease.replace('?', np.nan)
model = Bayesian model([('age', 'trestbps'), ('age', 'fbs'), ('sex',
        'testbps'), ('exang', 'testbps'), ('trestbps', 'heartdisease'
        ('fbs', 'heartdisease'), ('heartdisease', 'restecg')])
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	phi (heart Disease)
heart Disease-0	0.5593
heart Disease-1	0.1107

POORNIMA

Experiment - 9

Apply EM algorithm to cluster a set of data stored in a CSV file. Use the same 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, 1000])
plt.ylim([0, 50])
plt.title('Dataset')
plt.scatter(x1, x2)
plt.show()
# code for EM
gmm = GaussianMixture(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
```