LAB EXERCISE – 4

Naive Bayes Classifier

1. Aim of the Experiment:

Implement and demonstrate the working of Naive Bayesian classifier using a sample data set. Build the model to classify a test sample.

Listing 1:

Sample Dataset Used: **Table 8.1**

Table 8.1 Training Dataset

S.No.	CGPA	Interactiveness	Practical Knowledge	Communication Skills	Job Offer
1.	≥9	Yes	Very good	Good	Yes
2.	≥8	No	Good	Moderate	Yes
3.	≥9	No	Average	Poor	No
4.	<8	No	Average	Good	No
5.	≥8	Yes	Good	Moderate	Yes
6.	≥9	Yes	Good	Moderate	Yes
7.	<8	Yes	Good	Poor	No
8.	≥9	No	Very good	Good	Yes
9.	≥8	Yes	Good	Good	Yes
10.	≥8	Yes	Average	Good	Yes

3. Python Program with Explanation:

1. Import LabelEncoder to normalize labels.

```
from sklearn.preprocessing import LabelEncoder
```

2. Import train_test_split function.

```
from sklearn.model_selection import train_test_split
```

3. Import Gaussian Classifier from sklearn.naive_bayes.

```
from sklearn.naive_bayes import GaussianNB
```

4. Import preprocessing package to use transformer classes.

```
from sklearn import preprocessing
```

5. Import classification_report and confusion_matrix from sklearn.metrics to measure the quality of predictions.

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

6. Create lists, CGPA, Inter, PK, CS and Job.

```
CGPA = ['g9','g8','g9','l8','g8','g9','l8','g9','g8','g8']
Inter = ['Y','N','N','N','Y','Y','Y','N','Y','Y']
PK = ['+++','+','==','==','+','+','+','+++','+','==']
CS = ['G','M','P','G','M','M','P','G','G','G']
Job = ['Y','Y','N','N','Y','Y','N','Y','Y','Y']
```

7. Create labelEncoder le to encode labels with value between 0 and no_of_classes-1.

```
le = preprocessing.LabelEncoder()
```

8. Convert non-numeric labels of all features into numbers. Then print and see the encoded labels.

```
CGPA_encoded = le.fit_transform(CGPA)
print("CGPA:", CGPA_encoded)
Inter_encoded = le.fit_transform(Inter)
PK_encoded = le.fit_transform(PK)
CS_encoded = le.fit_transform(CS)
label = le.fit_transform(Job)
print("Inter:",Inter_encoded)
print ("PK:",PK_encoded)
print ("CS:",CS_encoded)
print("Job:",label)
```

9. Create a list/dynamic array called ‘features’.

```
features = []
```

10. Append all the encoded features to the list created.

```
for i in range(len(CGPA_encoded)):
 features.append([CGPA_encoded[i], Inter_encoded[i],
PK_encoded[i], CS_encoded[i]])
```

11. Split the dataset into training dataset and test dataset by using the function train_test_split().

```
X_train,X_test,y_train,y_test=train_test_split(features,label,test_size
=0.30,random_state=2)
```

12. Create a Gaussian Classifier.

```
model = GaussianNB()
```

13. Train the model using the training sets.

```
model.fit(features, label)
```

14. After training, use the fitted model to predict a new instance.

```
y_pred = model.predict(X_test)
```

15. Generate classification report & confusion matrix to measure the quality of predictions.

```
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

16. Predict Output.

```
# The non-numerical equivalent of the new instance [2, 0, 2, 0] given is [2:‘l8’, 0:‘N’,
2:‘==’, 0:‘G’]
print([2,0,2,0])
if model.predict([[2,0,2,0]])==1:
   print("Predicted Value:Got JOB",predicted )
else:
```

```
    print("Didnt get JOB")

    # The non-numerical equivalent of the new instance [0, 1, 0, 1] given is [0:‘g8’, 1:‘Y’, 0:‘+’, 1:‘M’]
    print([0,1,0,1])
    if model.predict([[0,1,0,1]])==1:
        print("Predicted Value:Got JOB",predicted )
    else:
        print("Didnt get JOB")
```

Complete Program:

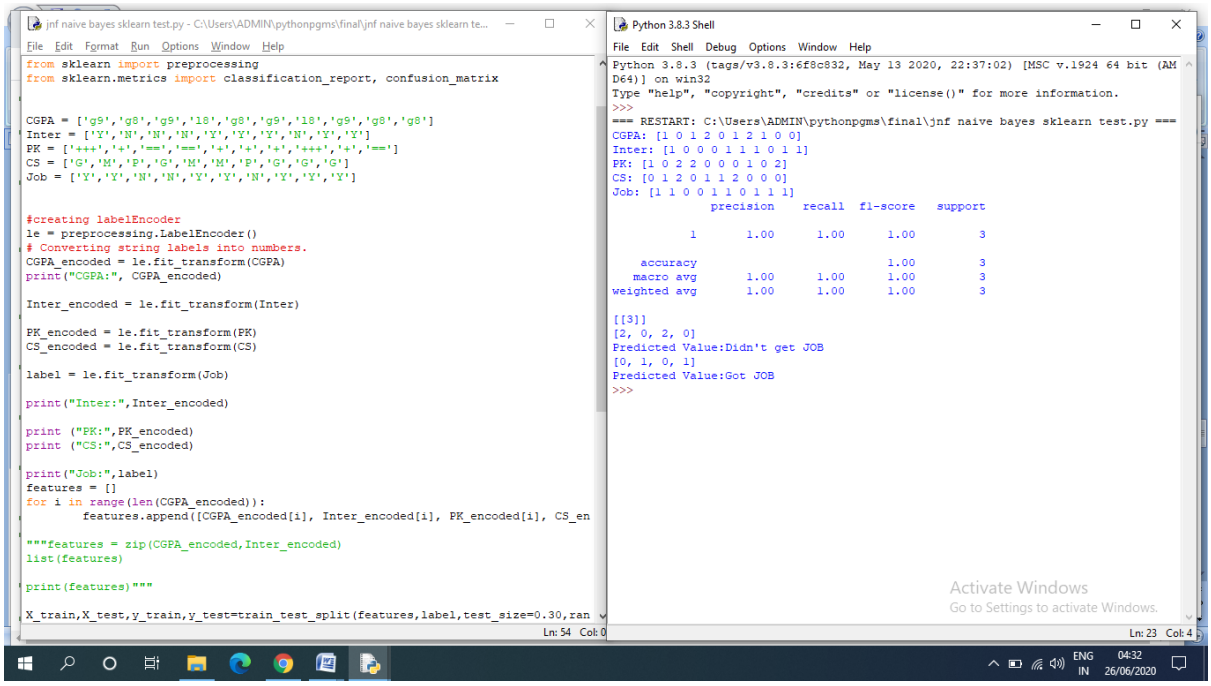
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import preprocessing
from sklearn.metrics import classification_report, confusion_matrix


CGPA = ['g9','g8','g9','l8','g8','g9','l8','g9','g8','g8']
Inter = ['Y','N','N','N','Y','Y','Y','N','Y','Y']
PK = ['+++','+','==','==','+','+','+','+++','+','==']
CS = ['G','M','P','G','M','M','P','G','G','G']
Job = ['Y','Y','N','N','Y','Y','N','Y','Y','Y']

#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
CGPA_encoded = le.fit_transform(CGPA)
print("CGPA:", CGPA_encoded)

Inter_encoded = le.fit_transform(Inter)
PK_encoded = le.fit_transform(PK)
CS_encoded = le.fit_transform(CS)
label = le.fit_transform(Job)
print("Inter:",Inter_encoded)
```

```
print ("PK:",PK_encoded)
print ("CS:",CS_encoded)
print("Job:",label)

features = []
for i in range(len(CGPA_encoded)):
        features.append([CGPA_encoded[i], Inter_encoded[i], PK_encoded[i],
CS_encoded[i]])

X_train,X_test,y_train,y_test=train_test_split(features,label,test_size=0.30,rando
m_state=2)
#Create a Gaussian Classifier
model = GaussianNB()

# Train the model using the training sets
model.fit(features,label)

#Predict Output
y_pred = model.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

print([2,0,2,0])
if model.predict([[2,0,2,0]])==1:
   print("Predicted Value:Got JOB")
else:
   print("Predicted Value:Didn't get JOB")

print([0,1,0,1])
if model.predict([[0,1,0,1]])==1:
   print("Predicted Value:Got JOB")
else:
   print("Predicted Value:Didn't get JOB")
```

Output:

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: C:\Users\ADMIN\pythonpgms\final\jnf naive bayes sklearn test.py
===
CGPA: [1 0 1 2 0 1 2 1 0 0]
Inter: [1 0 0 0 1 1 1 0 1 1]
PK: [1 0 2 2 0 0 0 1 0 2]
CS: [0 1 2 0 1 1 2 0 0 0]
Job: [1 1 0 0 1 1 0 1 1 1]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         3

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3

[[3]]
[2, 0, 2, 0]
Predicted Value:Didn't get JOB
[0, 1, 0, 1]
Predicted Value:Got JOB
>>>
```

Screenshot of the Output:



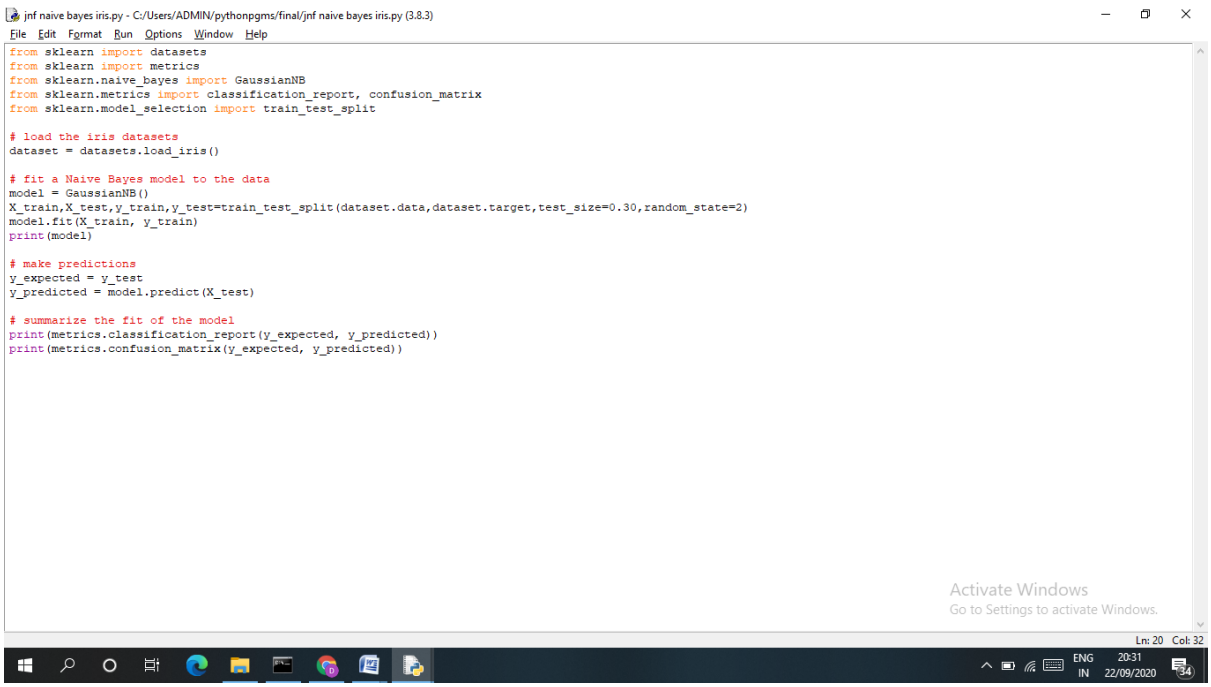
Listing 2:

Program Code:

```
from sklearn import datasets

from sklearn import metrics

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import classification_report, confusion_matrix

from sklearn.model_selection import train_test_split


# Load the Iris dataset

dataset = datasets.load_iris()


# Fit a Naive Bayes model to the data

model = GaussianNB()

X_train,X_test,y_train,y_test=train_test_split(dataset.data,dataset.target,test_size=0.30,random_state=2)

model.fit(X_train, y_train)

print(model)
```

```
# Make predictions
y_expected = y_test
y_predicted = model.predict(X_test)

# Evaluate the model and print the classification report, Confusion Matrix
print(metrics.classification_report(y_expected, y_predicted))
print(metrics.confusion_matrix(y_expected, y_predicted))
```

Screen Shot of the Program:



Output:

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

======= RESTART: C:\Users\ADMIN\pythonpgms\Review\jnf naive bayes iris.py ======
```

GaussianNB()

	precision	recall	f1-score	support
0	1.00	1.00	1.00	17
1	1.00	0.93	0.97	15
2	0.93	1.00	0.96	13
accuracy			0.98	45
macro avg	0.98	0.98	0.98	45
weighted avg	0.98	0.98	0.98	45

```
[[17  0  0]
 [ 0 14  1]
 [ 0  0 13]]
>>>
```

Screen Shot of the Output: