Experiment – 5

Student Name: Vivek Kumar **UID: 21BCS8129**
Branch: BE-CSE(LEET) **Section/Group: WM-20BCS-616/A**
Semester: 5th **Date of Performance: 12/10/2022**
Subject Name: Machine Learning Lab **Subject Code: 20CSP-317**

1. Aim/Overview of the practical:

Implement Naïve Bayes on any Dataset.

2. Task to be done/ Which logistics used:

Implement Naïve Bayes on any data set using sklearn.

3. Steps for experiment/practical/Code:

```
from google.colab import drive
drive.mount('/content/drive')

# importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# importing the dataset
dataset = pd.read_csv('/content/drive/MyDrive/Data/NaiveBayes.csv')

# split the data into inputs and outputs
X = dataset.iloc[:, [0,1]].values
y = dataset.iloc[:, 2].values

# training and testing data
from sklearn.model_selection import train_test_split

# assign test data size 25%
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size= 0.25, random_state=0)

# importing standard scaler
from sklearn.preprocessing import StandardScaler

# scalling the input data
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)

# importing classifier
from sklearn.naive_bayes import BernoulliNB
```

```
# initializaing the NB
classifer = BernoulliNB()

# training the model
classifer.fit(X_train, y_train)

# testing the model
y_pred = classifer.predict(X_test)

# importing accuracy score
from sklearn.metrics import accuracy_score

# printing the accuracy of the model
print(accuracy_score(y_pred, y_test))

# import Gaussian Naive Bayes classifier
from sklearn.naive_bayes import GaussianNB

# create a Gaussian Classifier
classifer1 = GaussianNB()

# training the model
classifer1.fit(X_train, y_train)

# testing the model
y_pred1 = classifer1.predict(X_test)

# importing accuracy score
from sklearn.metrics import accuracy_score

# printing the accuracy of the model
print(accuracy_score(y_test,y_pred1))

# importing the required modules
import seaborn as sns
from sklearn.metrics import confusion_matrix

# passing actual and predicted values
cm = confusion_matrix(y_test, y_pred)

# true write data values in each cell of the matrix
sns.heatmap(cm, annot=True)
plt.savefig('confusion.png')

# importing classification report
from sklearn.metrics import classification_report

# printing the report
print(classification_report(y_test, y_pred))

# importing the required modules
import seaborn as sns
```

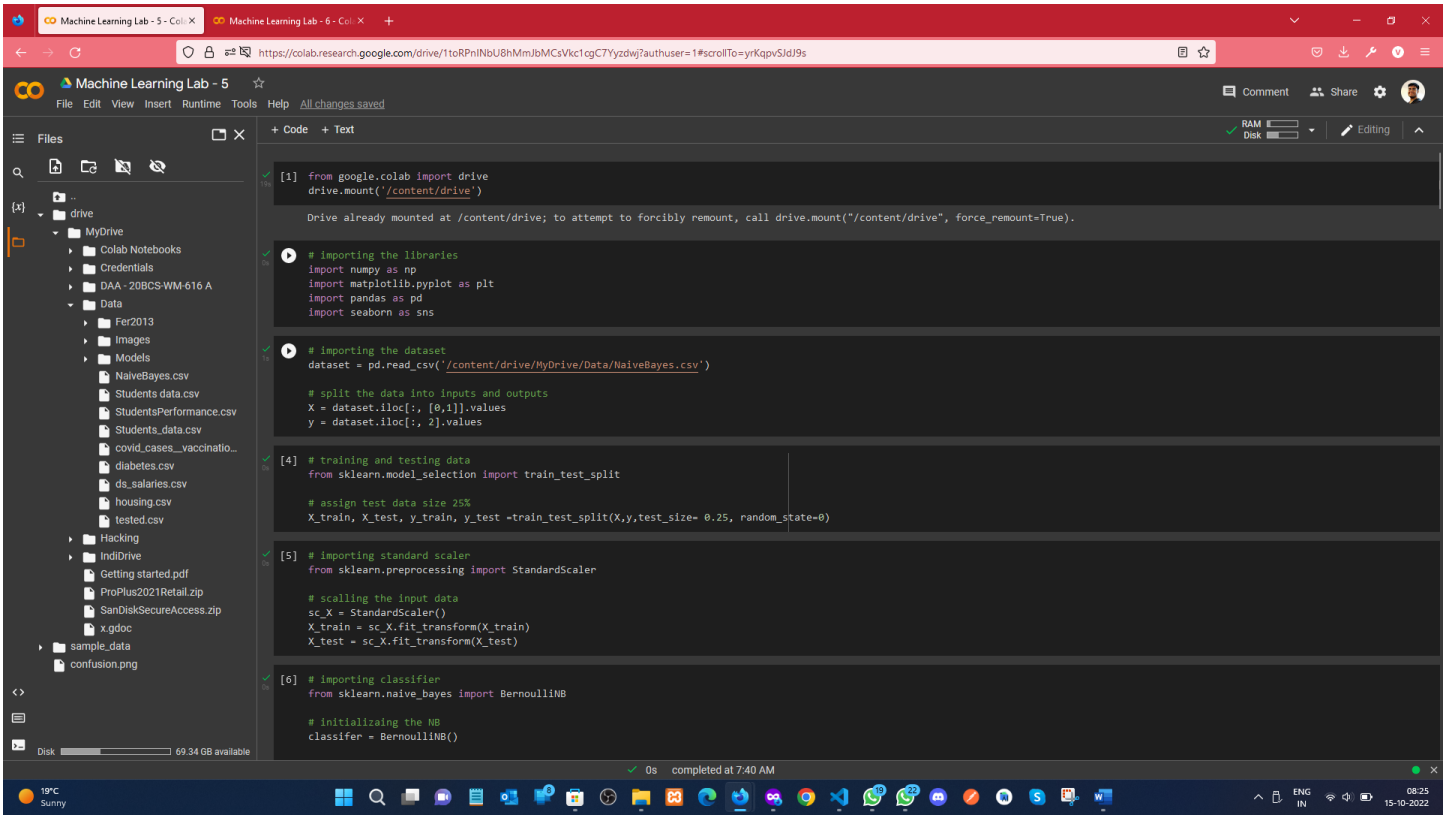
```
from sklearn.metrics import confusion_matrix

# passing actual and predicted values
cm = confusion_matrix(y_test, y_pred1)

# true write data values in each cell of the matrix
sns.heatmap(cm,annot=True)
plt.savefig('confusion.png')

# importing classification report
from sklearn.metrics import classification_report

# printing the report
print(classification_report(y_test, y_pred1))

# assigning features and label variables
weather = ['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny',
'Rainy','Sunny','Overcast','Overcast','Rainy']

# output class
play = ['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']

# Import LabelEncoder
from sklearn import preprocessing

# creating LabelEncoder
labelCode = preprocessing.LabelEncoder()

# Converting string labels into numbers.
wheather_encoded=labelCode.fit_transform(weather)

print(wheather_encoded)

# import LabelEncoder
from sklearn import preprocessing

# creating LabelEncoder
labelCode = preprocessing.LabelEncoder()

# converting string labels into numbers.
label=labelCode.fit_transform(play)

# import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

# create a Gaussian Classifier
model = GaussianNB()

# train the model using the training sets
model.fit(wheather_encoded, label)

# importing numpy module
```

```
import numpy as np

# converting 1D array to 2D
weather_2d = np.reshape(wheather_encoded, (-1, 1))

# import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

# create a Gaussian Classifier
model = GaussianNB()

# train the model using the training sets
model.fit(weather_2d, label)

# predicting the odel
predicted= model.predict([[0]]) # 0:Overcast

# printing predicted value
print(predicted)

# import scikit-learn dataset library
from sklearn import datasets

# load dataset
dataset = datasets.load_wine()

# print the names of the 13 features
print ("Inputs: ", dataset.feature_names)

# print the label type of wine
print ("Outputs: ", dataset.target_names)

# print the wine data features
print(dataset.data[0:3])

# print the wine labels
print(dataset.target)

# import train_test_split function
from sklearn.model_selection import train_test_split

# input and outputs
inputs = dataset.data
outputs = dataset.target

# split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(inputs, outputs, test_size=0.3,
random_state=1)

# import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
```

```
# create a Gaussian Classifier
classifer = GaussianNB()

# train the model using the training sets
classifer.fit(X_train, y_train)

# predict the response for test dataset
y_pred = classifer.predict(X_test)

# import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# printing accuracy
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

# importing the required modules
import seaborn as sns
from sklearn.metrics import confusion_matrix

# passing actual and predicted values
cm = confusion_matrix(y_test, y_pred)

# true Write data values in each cell of the matrix
sns.heatmap(cm, annot=True)
plt.savefig('confusion.png')

# Importing classification report
from sklearn.metrics import classification_report

# printing the report
print(classification_report(y_test, y_pred))

# importring modules
import matplotlib.pyplot as plt
import pandas as pd

# importing the dataset
dataset = pd.read_csv('/content/drive/MyDrive/Data/NaiveBayes.csv')

# split the data into inputs and outputs
X = dataset.iloc[:, [0,1]].values
y = dataset.iloc[:, 2].values

# training and testing data
from sklearn.model_selection import train_test_split

# assign test data size 25%
X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=0.25, random_state=0)

# importing StandardScaler
from sklearn.preprocessing import StandardScaler
```

```
# scalling the input data
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)

# importing bernoulli NB
from sklearn.naive_bayes import BernoulliNB

# initializaing the NB
classifer=BernoulliNB()

# training the model
classifer.fit(X_train, y_train)

# testing the model
y_pred = classifer.predict(X_test)

# importing accuracy score
from sklearn.metrics import accuracy_score

# printing the accuracy of the model
print(accuracy_score(y_test, y_pred))
```

4. Result/Output/Writing Summary:



Machine Learning Lab - 5

```
[6] # training the model
    classifer.fit(X_train, y_train)

    # testing the model
    y_pred = classifer.predict(X_test)
```

```
[7] # importing accuracy score
    from sklearn.metrics import accuracy_score

    # printing the accuracy of the model
    print(accuracy_score(y_pred, y_test))
```

```
0.8
```

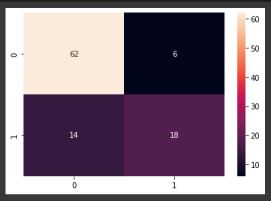
```
# import Gaussian Naive Bayes classifier
from sklearn.naive_bayes import GaussianNB

# create a Gaussian Classifier
classifer1 = GaussianNB()

# training the model
classifer1.fit(X_train, y_train)

# testing the model
y_pred1 = classifer1.predict(X_test)
```

```
[9] # importing accuracy score
    from sklearn.metrics import accuracy_score

    # printing the accuracy of the model
    print(accuracy_score(y_test,y_pred1))
```

```
0.91
```

```
# importing the required modules
import seaborn as sns
from sklearn.metrics import confusion_matrix

# passing actual and predicted values
cm = confusion_matrix(y_test, y_pred)
```

```
[10] # true write data values in each cell of the matrix
     sns.heatmap(cm, annot=True)
     plt.savefig('confusion.png')
```



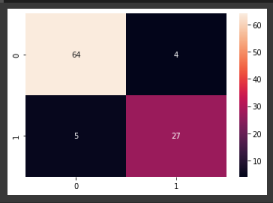

```
[11] # importing classification report
     from sklearn.metrics import classification_report

     # printing the report
     print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.91      0.86        68
           1       0.75      0.56      0.64        32

    accuracy                           0.80       100
   macro avg       0.78      0.74      0.75       100
weighted avg       0.79      0.80      0.79       100
```

```
# importing the required modules
import seaborn as sns
from sklearn.metrics import confusion_matrix

# passing actual and predicted values
cm = confusion_matrix(y_test, y_pred1)

# true write data values in each cell of the matrix
```

Machine Learning Lab - 5

```
cm = confusion_matrix(y_test, y_pred1)

# true write data values in each cell of the matrix
sns.heatmap(cm,annot=True)
plt.savefig('confusion.png')
```




```
[13] # importing classification report
from sklearn.metrics import classification_report

# printing the report
print(classification_report(y_test, y_pred1))
```

```
              precision    recall  f1-score   support

           0       0.93      0.94      0.93        68
           1       0.87      0.84      0.86        32

    accuracy                           0.91       100
   macro avg       0.90      0.89      0.90       100
weighted avg       0.91      0.91      0.91       100
```

```
# assigning features and label variables
weather = ['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny', 'Rainy','Sunny','Overcast','Overcast','Rainy']

# output class
play = ['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']
```

0s completed at 7:40 AM

Machine Learning Lab - 5

```
[15] # Import LabelEncoder
from sklearn import preprocessing

# creating LabelEncoder
labelCode = preprocessing.LabelEncoder()

# Converting string labels into numbers.
wheather_encoded=labelCode.fit_transform(weather)
```

```
[16] print(wheather_encoded)
```

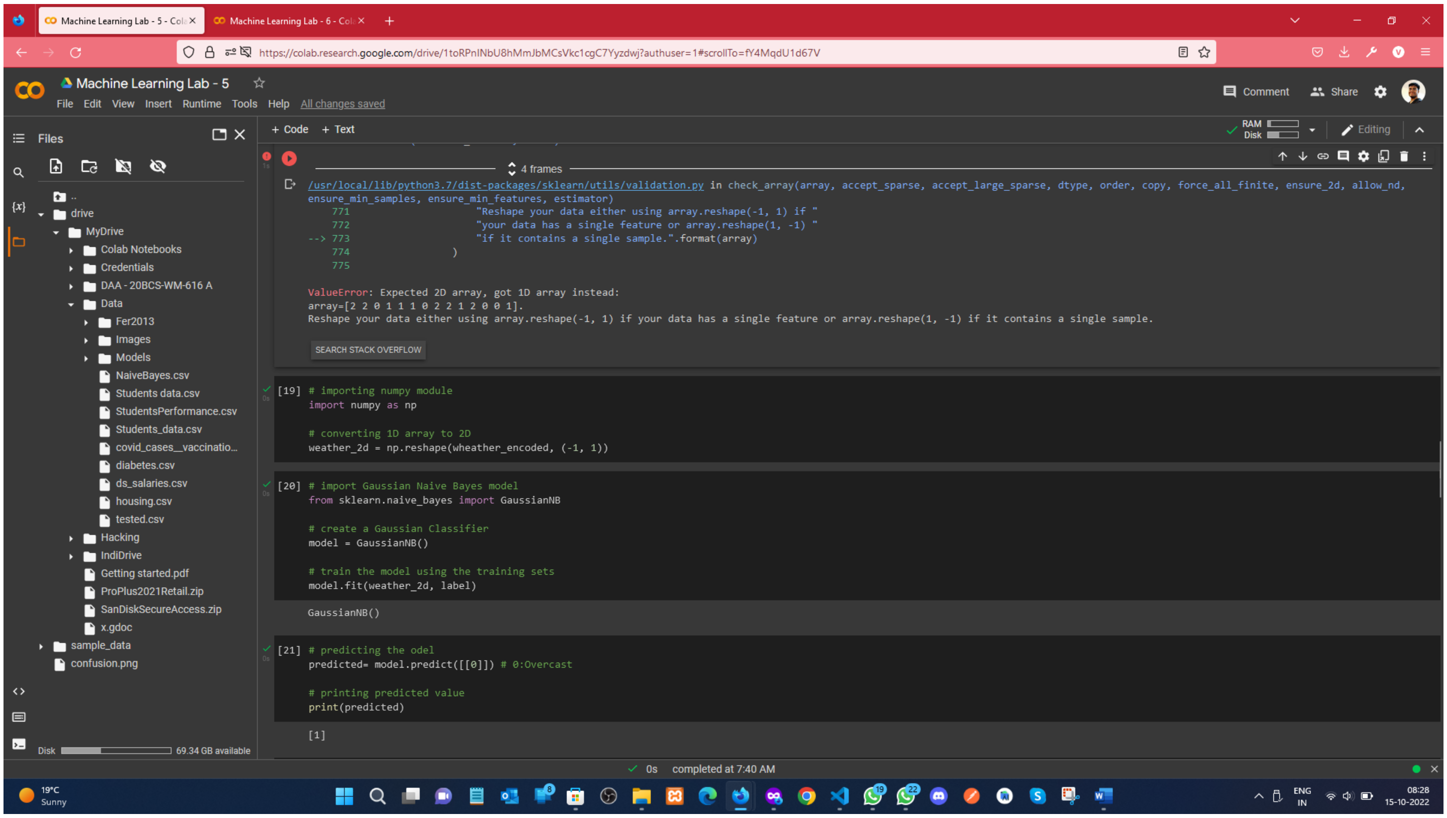
```
[2 2 0 1 1 1 0 2 2 1 2 0 0 1]
```

```
[17] # import LabelEncoder
from sklearn import preprocessing

# creating LabelEncoder
labelCode = preprocessing.LabelEncoder()

# converting string labels into numbers.
label=labelCode.fit_transform(play)
```

```
# import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

# create a Gaussian Classifier
model = GaussianNB()

# train the model using the training sets
model.fit(wheather_encoded, label)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-18-92b593933a77> in <module>
      6
      7 # train the model using the training sets
----> 8 model.fit(wheather_encoded, label)

                                  4 frames
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
```

0s completed at 7:40 AM

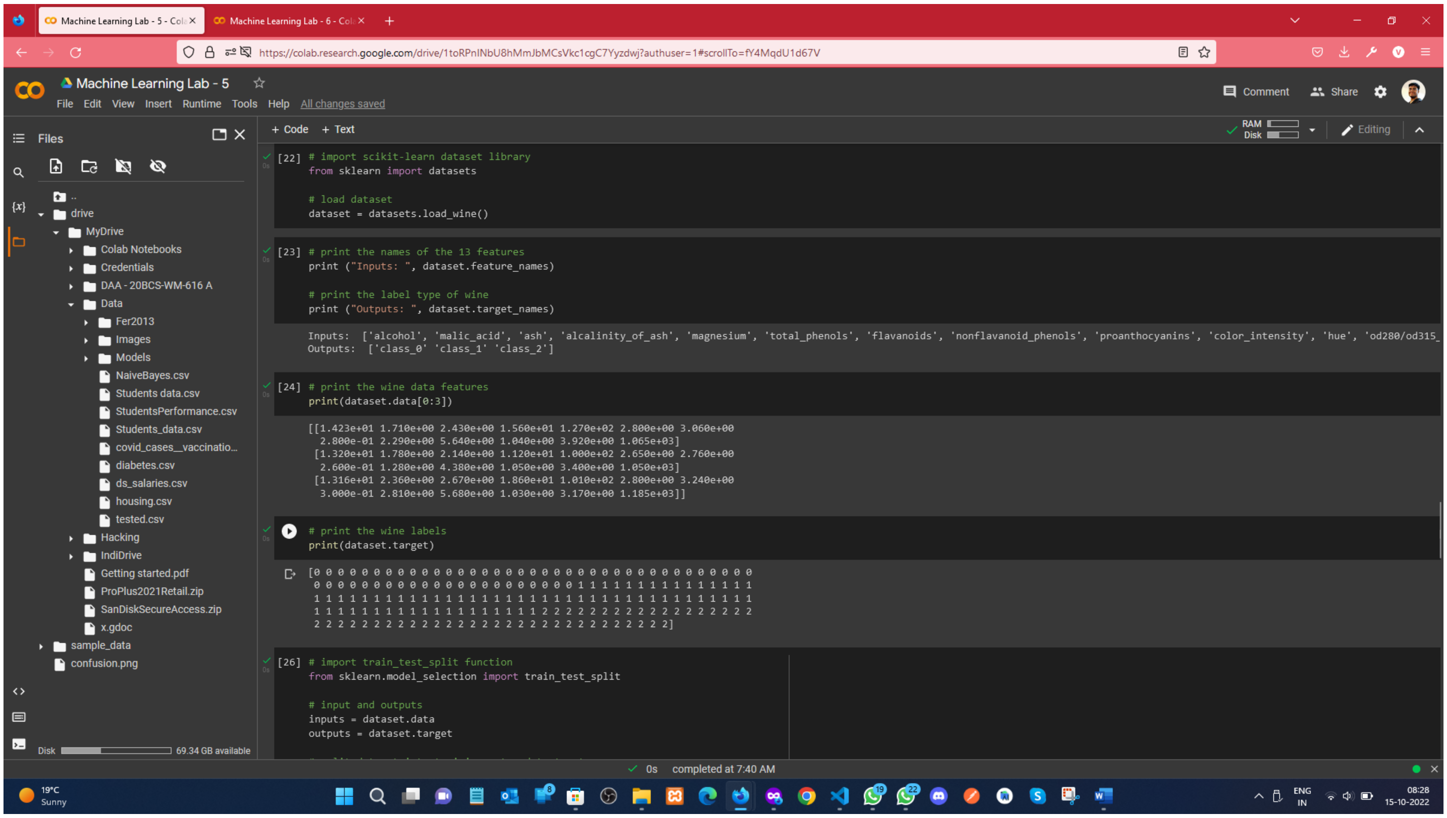


```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    771                 "Reshape your data either using array.reshape(-1, 1) if "
    772                 "your data has a single feature or array.reshape(1, -1) "
--> 773                 "if it contains a single sample.".format(array)
    774             )
    775

ValueError: Expected 2D array, got 1D array instead:
array=[2 2 0 1 1 1 0 2 2 1 2 0 0 1].
Reshape your data either using array.reshape(-1, 1) if your data has a single feature or array.reshape(1, -1) if it contains a single sample.
```

```
[19] # importing numpy module
     import numpy as np

     # converting 1D array to 2D
     weather_2d = np.reshape(wheather_encoded, (-1, 1))
```

```
[20] # import Gaussian Naive Bayes model
     from sklearn.naive_bayes import GaussianNB

     # create a Gaussian Classifier
     model = GaussianNB()

     # train the model using the training sets
     model.fit(weather_2d, label)

     GaussianNB()
```

```
[21] # predicting the odel
     predicted= model.predict([[0]]) # 0:Overcast

     # printing predicted value
     print(predicted)

     [1]
```



```
[22] # import scikit-learn dataset library
     from sklearn import datasets

     # load dataset
     dataset = datasets.load_wine()
```

```
[23] # print the names of the 13 features
     print ("Inputs: ", dataset.feature_names)

     # print the label type of wine
     print ("Outputs: ", dataset.target_names)

Inputs:  ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'od280/od315_
Outputs:  ['class_0' 'class_1' 'class_2']
```

```
[24] # print the wine data features
     print(dataset.data[0:3])

[[1.423e+01 1.710e+00 2.430e+00 1.560e+01 1.270e+02 2.800e+00 3.060e+00
  2.800e-01 2.290e+00 5.640e+00 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 1.120e+01 1.000e+02 2.650e+00 2.760e+00
  2.600e-01 1.280e+00 4.380e+00 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 1.860e+01 1.010e+02 2.800e+00 3.240e+00
  3.000e-01 2.810e+00 5.680e+00 1.030e+00 3.170e+00 1.185e+03]]
```

```
# print the wine labels
print(dataset.target)

[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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

```
[26] # import train_test_split function
     from sklearn.model_selection import train_test_split

     # input and outputs
     inputs = dataset.data
     outputs = dataset.target
```

Machine Learning Lab - 5

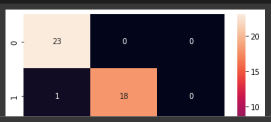
```
# split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(inputs, outputs, test_size=0.3, random_state=1)
```

```
# import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

# create a Gaussian Classifier
classifer = GaussianNB()

# train the model using the training sets
classifer.fit(X_train, y_train)

# predict the response for test dataset
y_pred = classifer.predict(X_test)
```

```
# import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# printing accuracy
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

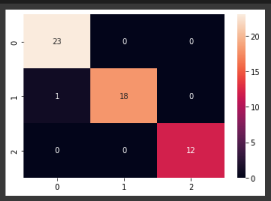
```
Accuracy: 0.9814814814814815
```

```
# importing the required modules
import seaborn as sns
from sklearn.metrics import confusion_matrix

# passing actual and predicted values
cm = confusion_matrix(y_test, y_pred)

# true Write data values in each cell of the matrix
sns.heatmap(cm, annot=True)
plt.savefig('confusion.png')
```



Machine Learning Lab - 5

```
# importing the required modules
import seaborn as sns
from sklearn.metrics import confusion_matrix

# passing actual and predicted values
cm = confusion_matrix(y_test, y_pred)

# true Write data values in each cell of the matrix
sns.heatmap(cm, annot=True)
plt.savefig('confusion.png')
```




```
# Importing classification report
from sklearn.metrics import classification_report

# printing the report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        23
           1       1.00      0.95      0.97        19
           2       1.00      1.00      1.00        12

    accuracy                           0.98        54
   macro avg       0.99      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

Learning outcomes (What I have learnt):

1. Understood the concept of Naïve Bayes (NB)
2. Learnt how to split the data into training and testing parts and perform operation on it.
3. Understood the concept of GaussianNB, BernoulliNB, and confusion matrix.
4. Finally plotted the classification report.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			