Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19MN4C2) AY: 2021-22

Experiment 5

(Naïve Bayes Classifier)

Name : Ayush Jain **SAP-ID :**60004200132 **Branch :** Computer Engineering

Aim: Implement Naïve Bayes Classifier on a given Dataset.

Theory:

Naïve Bayes Classifier Algorithm ○ Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.

- It is mainly used in *text classification* that includes a high-dimensional training dataset. ○ Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Types of Naïve Bayes Model:
There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.
  The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

Naïve Bayes Classifier

```
!pip install -U scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packages (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.22.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (3.1.0)
```

Data Preprocessing

```
import numpy as np
import pandas as pd
#importing dataset
dataset1 = pd.read_csv("/content/Breast_cancer_data.csv")
print('Dataset1: Breast Cancer\n',dataset1) #get the view od dataset
```

```
Dataset1: Breast Cancer
      mean_radius  mean_texture  mean_perimeter  mean_area  mean_smoothness  \
0           17.99         10.38          122.80     1001.0          0.11840
1           20.57         17.77          132.90     1326.0          0.08474
2           19.69         21.25          130.00     1203.0          0.10960
3           11.42         20.38           77.58      386.1          0.14250
4           20.29         14.34          135.10     1297.0          0.10030
..            ...           ...             ...        ...              ...
564         21.56         22.39          142.00     1479.0          0.11100
565         20.13         28.25          131.20     1261.0          0.09780
566         16.60         28.08          108.30      858.1          0.08455
567         20.60         29.33          140.10     1265.0          0.11780
568          7.76         24.54           47.92      181.0          0.05263

     diagnosis
0            0
1            0
2            0
3            0
4            0
..         ...
564          0
565          0
566          0
567          0
568          1

[569 rows x 6 columns]
```

Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution

```
#Extracting Independent X and Dependent Y Value
#abc.iloc[all rows, (all column except first and last)or range from ]
Features_BC_X = dataset1.iloc[:,:4].values #:Index gives array in 1D, :-1 gives array in 2D
#abc.iloc[all rows, only last column]
labels_diagnosis = dataset1.iloc[:,-1].values
print('\nValues of Breast Cancer Features :\n', Features_BC_X)
print('\nDiagnosis of Breast Cancer :\n', labels_diagnosis)
```

```
    Values of Breast Cancer Features :
     [[  17.99    10.38  122.8  1001.  ]
     [  20.57    17.77  132.9  1326.  ]
     [  19.69    21.25  130.   1203.  ]
     ...
     [  16.6     28.08  108.3   858.1 ]
     [  20.6     29.33  140.1  1265.  ]
     [   7.76    24.54   47.92  181.  ]]

    Diagnosis of Breast Cancer :
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 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 0 0 0 0 0 0 1]
```

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
Features_BC_train, Features_BC_test, labels_diagnosis_train, labels_diagnosis_test = train_test_split(Features_BC_X, labels_diagnosis, test_s
print('\n Features_BC_train =\n',Features_BC_train)
print('\n Features_BC_test =\n',Features_BC_test)
print('\n labels_diagnosis_train =\n',labels_diagnosis_train)
print('\n labels_diagnosis_test =\n',labels_diagnosis_test)
```

```
 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0]
```

```
# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier_1 = GaussianNB()
classifier_1.fit(Features_BC_train,labels_diagnosis_train) #for Dataset1
```

```
▾ GaussianNB
GaussianNB()
```

```
# Predicting the Test set results
labels_diagnosis_predicted = classifier_1.predict(Features_BC_test)  #for dataset1
print('\nlabels_diagnosis_predicted =\n',labels_diagnosis_predicted)
```

```
labels_diagnosis_predicted =
 [1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 1 1
 0 1 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 1 0 1 1
 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 0 0 0 1 1 0 1 1 1 1 1 0 1 0 1 0 1 0 0 1
 0 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 1 0]
```

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
CM_BC = confusion_matrix(labels_diagnosis_test,labels_diagnosis_predicted)  #for dataset1
print('\nConfusion Matrix of Breast Cancer =\n',CM_BC)
```

```
Confusion Matrix of Breast Cancer =
 [[46  7]
 [ 6 84]]
```

```
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Dataset1 Model Accuracy:", metrics.accuracy_score(labels_diagnosis_test,labels_diagnosis_predicted)) #for dataset1
```

```
Dataset1 Model Accuracy: 0.9090909090909091
```

```
import numpy as np
import pandas as pd
#importing dataset
dataset2 = pd.read_csv("/content/Social_Network_Ads.csv")
print('Dataset2: Social Networking Ads\n',dataset2)
```

```
Dataset2: Social Networking Ads
      User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510    Male   19            19000          0
1    15810944    Male   35            20000          0
2    15668575  Female   26            43000          0
3    15603246  Female   27            57000          0
4    15804002    Male   19            76000          0
..        ...     ...  ...              ...        ...
395  15691863  Female   46            41000          1
396  15706071    Male   51            23000          1
397  15654296  Female   50            20000          1
398  15755018    Male   36            33000          0
399  15594041  Female   49            36000          1

[400 rows x 5 columns]
```

```
# Import label encoder
from sklearn import preprocessing
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
# Encode labels in column 'Gender'.
dataset2['Gender']= label_encoder.fit_transform(dataset2['Gender'])
```

```
       User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510       1   19            19000          0
1    15810944       1   35            20000          0
2    15668575       0   26            43000          0
3    15603246       0   27            57000          0
4    15804002       1   19            76000          0
..        ...     ...  ...              ...        ...
395  15691863       0   46            41000          1
396  15706071       1   51            23000          1
397  15654296       0   50            20000          1
398  15755018       1   36            33000          0
399  15594041       0   49            36000          1

[400 rows x 5 columns]
```

```
#Extracting Independent X and Dependent Y Value
#abc.iloc[all rows, (all column except first and last)or range from ]
Features_SNA_X = dataset2.iloc[:,1:-1].values #:Index gives array in 1D, :-1 gives array in 2D
#abc.iloc[all rows, only last column]
labels_Purchased = dataset2.iloc[:,-1].values
print('\nValues of Social Networking Ads Features :\n', Features_SNA_X)
print('\nPurchased Social Networking Ads :\n', labels_Purchased)
```

```
Values of Social Networking Ads Features :
 [[    1    19 19000]
 [    1    35 20000]
 [    0    26 43000]
 ...
 [    0    50 20000]
 [    1    36 33000]
 [    0    49 36000]]

Purchased Social Networking Ads :
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 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 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 1
 1 1 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1
 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0
 1 1 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0
 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1
 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1]
```

```
# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier_2 = GaussianNB()
classifier_2.fit(Features_SNA_train,labels_Purchased_train) #for Dataset2
```

```
▾ GaussianNB
GaussianNB()
```

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
Features_SNA_train, Features_SNA_test, labels_Purchased_train, labels_Purchased_test = train_test_split(Features_SNA_X, labels_Purchased, tes
print('\n Features_SNA_train =\n',Features_SNA_train)
print('\n Features_SNA_test =\n',Features_SNA_test)
print('\n labels_Purchased_train =\n',labels_Purchased_train)
print('\n labels_Purchased_test =\n',labels_Purchased_test)
```

```
[      0     35  77000]
[      0     22  63000]
[      1     45  22000]
[      1     27  89000]
[      1     18  82000]
[      0     42  79000]
[      0     40  60000]
[      0     53  34000]
[      0     47 107000]
[      1     58 144000]
[      0     59  83000]
[      0     24  55000]
[      0     26  35000]
[      0     58  38000]
[      0     42  80000]
[      0     40  75000]
[      1     59 130000]
[      0     46  41000]
[      0     41  60000]
[      1     42  64000]
[      0     37 146000]
[      0     23  48000]
[      1     25  33000]
[      1     24  84000]
[      0     27  96000]
[      1     23  63000]
[      1     48  33000]
[      1     48  90000]
[      1     42 104000]]

labels_Purchased_train =
[0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0
0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
0 0 0 0]

labels_Purchased_test =
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

```
# Predicting the Test set results
labels_Purchased_predicted = classifier_2.predict(Features_SNA_test) #for dataset2
print('\nlabels_Purchased_predicted =\n',labels_Purchased_predicted)
```

```
labels_Purchased_predicted =
 [0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0
 0 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 1 1]
```

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
CM_SNA = confusion_matrix(labels_Purchased_test,labels_Purchased_predicted)  #for dataset2
print('\nConfusion Matrix of Social Networking Ads =\n',CM_SNA)
```

```
Confusion Matrix of Social Networking Ads =
 [[65  3]
 [ 7 25]]
```

```
#Dishant Sunil Patil : 60011200048 - Chemical
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Dataset2 Model Accuracy:", metrics.accuracy_score(labels_Purchased_test,labels_Purchased_predicted)) #for dataset2
```

```
Dataset2 Model Accuracy: 0.9
```