EXERCISE-5

AIM: IMPLEMENT NAÏVE BAYESIAN CLASSIFIER FOR A SAMPLE TRAINING DATA SET STORED AS A '.CSV' FILE. COMPUTE THE ACCURACY OF THE CLASSIFIER, CONSIDERING FEW TEST DATA SETS.

DESCRIPTION:

Naive Bayes is a **statistical classification technique** based on the **Bayes Theorem** and one of the simplest **Supervised Learning algorithms.** **A Naive Bayes classifier** assumes that the effect of a **particular feature in a class is independent of other features** and is based on **Bayes' theorem**.

Bayes' theorem is a **mathematical equation** used in **probability and statistics to calculate conditional probability**.

In other words, you can use this theorem to **calculate the probability of an event with functions** like the **Gaussian Probability Density function** based on its **association with another event.**

The simple formula of the Bayes theorem is:

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

Where **P (A) and P (B)** are **two independent events and (B) is not equal to zero.**

P (A | B): is the **conditional probability of an event A occurring given that B is true.**

P (B | A): is the **conditional probability of an event B occurring given that A is true.**

P (A) and P (B): are the probabilities of A and B occurring **independently of one another (the marginal probability).**

The **Naive Bayes algorithm** offers **plenty of advantages to its users**. That's why it has a lot of applications in various **industries, including Health, Technology, Environment**, etc.

Implementation of the Naïve Bayes involves below steps:

1. Install the Packages:

(a) Numpy: Numpy Python library is used for including any type **of mathematical operation in the code**. It is the fundamental package for scientific calculation in Python. It also supports to **add large, multidimensional arrays and matrices**. **So, in Python, we can import it as:**

import numpy as np

(b) Matplotlib: The second library is matplotlib, which is a **Python 2D plotting library**, and with this library, we need to import a sub-library pyplot. This library is **used to plot any type of charts in Python** for the code. **It will be imported as below:**

import matplotlib.pyplot as plt

Here we have used **plt a**s a short name for this library.

c) Pandas: The last library is the Pandas library, which is one of the most famous Python libraries and used for **importing and managing the datasets**. It is an **open-source data manipulation and analysis library**. It will be imported as below:

import pandas as pd

Here, we have used **pd** as a short name for this library.

d) Seaborn: Seaborn is a Python **data visualization library** based on **matplotlib.** It provides a **high-level interface for drawing attractive and informative statistical graphics**.

import seaborn as sns

Here, we have used **sns** as a short name for this library.

2. Importing the Dataset:

read_csv() function: Now to import the dataset, we will use **read_csv() function** of **pandas library**, which is used to read a csv file and performs various operations on it. Using this function, we can read a csv file locally as well as through an URL. **We can use read_csv function as below:**

For Eg: dataset = pd.read_csv ('NaiveBayes.csv')

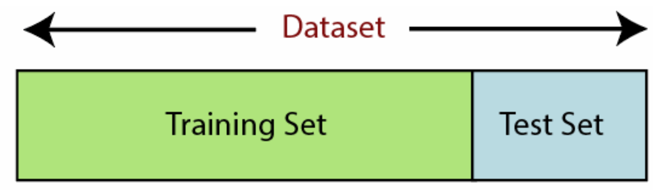
3. Separating Independent and Dependent Variables:

In machine learning, it is important to distinguish the matrix of features (independent variables) and dependent variables from dataset.

For Eg: In our dataset, there are **Two independent variables** that are **Age** and **Salary**, and **one is a dependent variable** which is **purchased**.

```
# Seperating Independent and Dependent Variable
x = dataset.iloc[:, [0,1]].values
y = dataset.iloc[:, 2].values
```

4. Splitting dataset into training and test set: we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model.



Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code:

```
# 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)
```

Explanation: We set test_size=0.25, which means **25%** of the whole data set will be assigned to the **testing** part, and the remaining **75%** will be used for the model's **training**.

5.Training model using Naive Bayes Classifier : Now, let's train our model using the Gaussian Naive Bayes classifier (a type of Naive Bayes Classifier).

```
# 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)
```

6. Find the Accuracy of the Model: Accuracy score in machine learning is an **evaluation metric** that **measures the number of correct predictions made by a model** in relation to **the total**

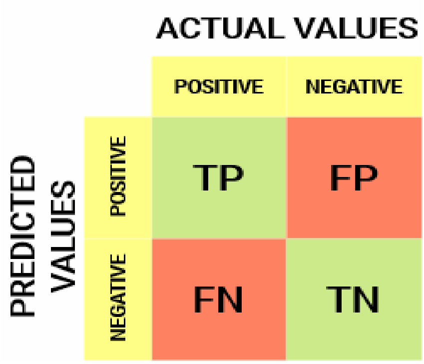
number of predictions made. We calculate it by dividing the number of correct predictions by the total number of predictions.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

```
# importing accuracy score
from sklearn.metrics import accuracy_score
# printing the accuracy of the model
print(accuracy_score(y_test,y_pred1))
```

Print the Confusion Matrix: The **confusion matrix** is one of the most popular and widely used performance measurement techniques for **classification models.**

Confusion Matrix as the name suggests gives us a **matrix as output and describes the complete performance of the model** and it also used to **determine the performance of the classification models for a given set of test data.**




```
# 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')
```

Explanation: **Heat Maps** are most commonly used to **display a more generalized view of numeric values as a graphical representation of data** where the individual values contained in a matrix **are represented as colors.**

Print the Classification Report:

```
# 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       0.90      0.96      0.93        68
           1       0.89      0.78      0.83        32

    accuracy                           0.90       100
   macro avg       0.90      0.87      0.88       100
weighted avg       0.90      0.90      0.90       100
```

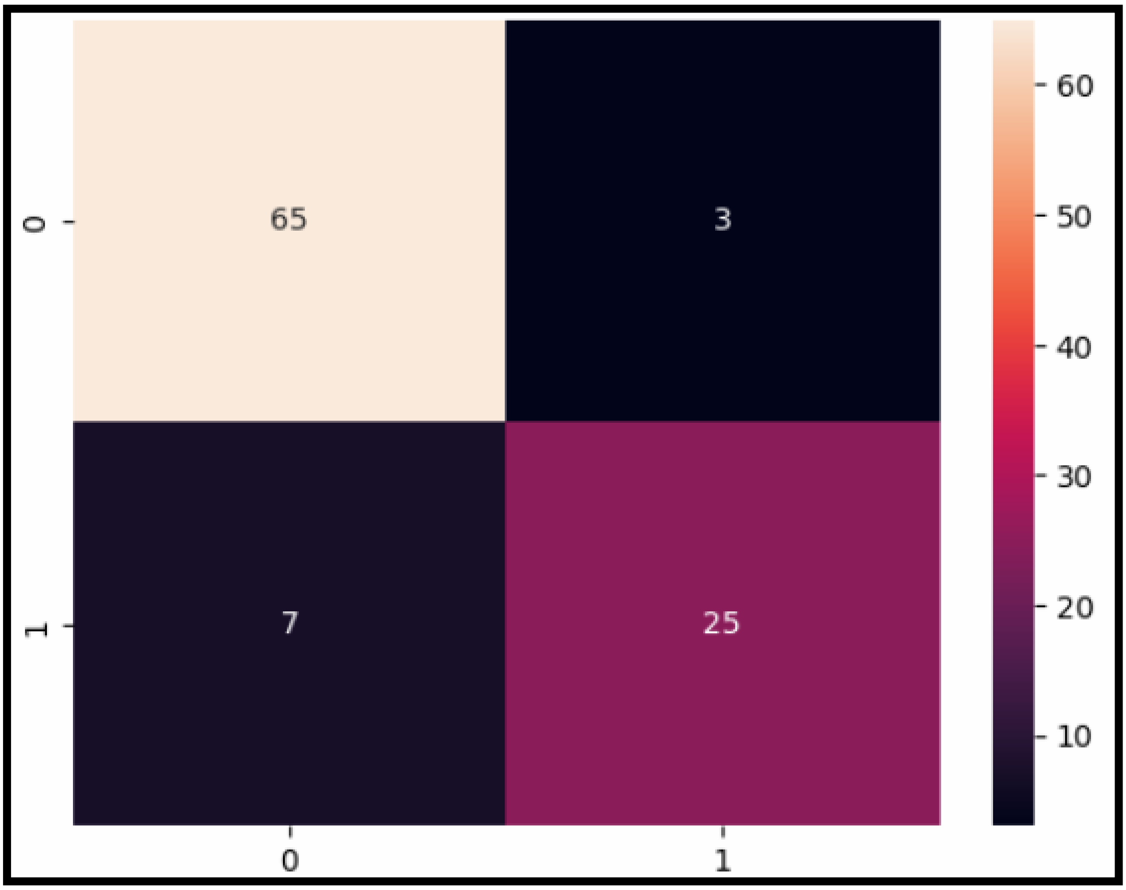
PROGRAM:

```
# 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('NaiveBayes.csv')
# Seperating Independent and Dependent Variable
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)
# 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)
y_pred1
# 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_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))
```

INPUT / OUTPUT:

Confusion Matrix



Classification Report

	precision	recall	f1-score	support
0	0.90	0.96	0.93	68
1	0.89	0.78	0.83	32
accuracy			0.90	100
macro avg	0.90	0.87	0.88	100
weighted avg	0.90	0.90	0.90	100

CONCLUSION: **Program is executed successfully without any error.**