Name:	
Roll No:	
Class/Sem:	TE/V
Experiment No.:	7
Title:	Implementation of any one classifier using languages like JAVA/ python.
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Aim: To implement Naïve Bayesian classification

Objective: Develop a program to implement Bayesian classification.

Theory:

The Naive Bayes is a classification algorithm that is suitable for binary and multiclass classification. Naïve Bayes performs well in cases of categorical input variables compared to numerical variables. It is useful for making predictions and forecasting data based on historical results.

The naïve Bayesian classifier, or simple Bayesian classifier, works as follows:

1) Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, X = (x1, x2,...,xn), depicting n measurements made on the tuple from n attributes, respectively, A1, A2,..., An.
2) Suppose that there are m classes, C1,C2,...,Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naïve Bayesian classifier predicts that tuple X belongs to the class Ci if and only if

   P(Ci|X) > P(Cj|X) Thus we maximize P(Ci|X).The class Ci for which P(Ci|X) is maximized is called the maximum posteriori hypothesis.

   By Bayes' theorem,

   P(Ci|X) = P(X|Ci)*P(Ci)/ P(X)
3) As P(X) is constant for all classes, only P(X|Ci)P(Ci) need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P(C1) = P(C2) = ⋯ = P(Cm), and we would therefore maximize P(X|Ci). Otherwise, we maximize P(X|Ci)P(Ci). Note that the class prior probabilities may be estimated by P(Ci)=|Ci,D|/|D|, where |Ci,D| is the number of training tuples of class Ci in D.

The equation: Posterior = Prior x (Likelihood over Marginal probability)

There are four parts:

- Posterior probability (updated probability after the evidence is considered)
- Prior probability (the probability before the evidence is considered)
- Likelihood (probability of the evidence, given the belief is true)
- Marginal probability (probability of the evidence, under any circumstance)

Bayes' Rule can answer a variety of probability questions, which help us (and machines) understand the complex world we live in.

Example:

Car No	Color	Type	Origin	Stolen
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	No
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

P (yes) =5/10
P (No)=5/10

-Color:
P(Red/Y)=3/5 P(yellow/Y)=2/5
P(Red/N)=2/5 P(yellow/N)=3/5

-Type:
P(SUV/Y)=1/5 P(Sports/Y)=4/5
P(SUV/N)=3/5 P(Sports/N)=2/5

-Origin:
P(Domentic/Y)=2/5 P(Imported/Y)=3/5
P(Domentic /N)=3/5 P(Imported/N)=2/5

P(x|Yes).P(Yes)= 0.024
P(x|No).P(No)=0.072
So, Bayesian Classification Predicts the class "NO"

Code:

```
import numpy as np
import pandas as pd
# Import necessary modules
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
d1 = pd. read_csv ('diabetes_csv.csv')

d1.head()

# Loading data
# Create feature and target arrays
X = d1[d1.columns[:-1]]
y = d1[d1.columns[-1]]
# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state=42)
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)

d1

from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):",
metrics.accuracy_score(y_test, y_pred)*100)
# Predict on dataset which model has not seen before
print(gnb.predict(X_test))
```

Output:

Python Output

```
Gaussian Naive Bayes model accuracy(in %): 76.62337662337663
['tested_negative' 'tested_negative' 'tested_negative' 'tested_negative'
 'tested_positive' 'tested_positive' 'tested_negative' 'tested_positive'
 'tested_negative' 'tested_positive' 'tested_negative' 'tested_positive'
 'tested_positive' 'tested_negative' 'tested_negative' 'tested_negative'
 'tested_negative' 'tested_negative' 'tested_positive' 'tested_negative'
 'tested_negative' 'tested_negative' 'tested_positive' 'tested_negative'
 'tested_positive' 'tested_positive' 'tested_negative' 'tested_negative'
 'tested_negative' 'tested_negative' 'tested_positive' 'tested_positive'
 'tested_positive' 'tested_positive' 'tested_positive' 'tested_positive'
 'tested_positive' 'tested_negative' 'tested_positive' 'tested_negative'
 'tested_negative' 'tested_negative' 'tested_negative' 'tested_positive'
 'tested_negative' 'tested_positive' 'tested_positive' 'tested_negative'
 'tested_negative' 'tested_positive' 'tested_negative' 'tested_positive'
 'tested_positive' 'tested_negative' 'tested_negative' 'tested_negative'
 'tested_positive' 'tested_negative' 'tested_negative' 'tested_positive'
 'tested_positive' 'tested_negative' 'tested_positive' 'tested_positive'
 'tested_negative' 'tested_positive' 'tested_negative' 'tested_positive'
 'tested_negative' 'tested_positive' 'tested_positive' 'tested_negative'
 'tested_negative' 'tested_negative' 'tested_negative' 'tested_positive'
 'tested_negative' 'tested_negative' 'tested_negative' 'tested_negative'
 'tested_positive' 'tested_negative' 'tested_negative' 'tested_negative'
 'tested_negative' 'tested_positive' 'tested_positive' 'tested_negative'
 'tested_negative' 'tested_negative' 'tested_negative' 'tested_negative'
 'tested_negative' 'tested_negative' 'tested_negative' 'tested_negative'
 'tested_negative' 'tested_negative' 'tested_positive' 'tested_negative'
 'tested_positive' 'tested_negative' 'tested_positive' 'tested_negative'
 'tested_positive' 'tested_negative' 'tested_negative' 'tested_positive'
 'tested_negative' 'tested_positive' 'tested_negative' 'tested_negative'
 'tested_negative' 'tested_positive' 'tested_negative' 'tested_negative'
 'tested_positive' 'tested_negative' 'tested_negative' 'tested_positive'
 'tested_negative' 'tested_negative' 'tested_negative' 'tested_negative'
 'tested_negative' 'tested_positive' 'tested_negative' 'tested_positive'
 'tested_positive' 'tested_positive' 'tested_positive' 'tested_positive'
 'tested_negative' 'tested_positive' 'tested_positive' 'tested_negative'
 'tested_negative' 'tested_positive' 'tested_positive' 'tested_negative'
 'tested_negative' 'tested_negative' 'tested_negative' 'tested_positive'
 'tested_negative' 'tested_negative' 'tested_negative' 'tested_negative'
 'tested_negative' 'tested_positive' 'tested_negative' 'tested_negative'
 'tested_positive' 'tested_negative']
```

Conclusion:

We observed that Bayesian Classification represents a supervised learning method as well as a statistical classification method. The Bayesian classification is used as a probabilistic learning method as Naive Bayes classifiers are among the most successful known algorithms for learning to classify the datasets. Naïve Bayes classification Algorithm is one of the probabilistic algorithms which classifies the datasets according to its knowledge data and creates the Result as per the given knowledge