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

Aim: To implement Naïve Bayesian Classification Algorithm.

Objective: To make students well versed in data mining algorithms like classification and implement Naïve Bayesian Classifier.

Theory:

- Classification is a form of data analysis that extracts models describing important data classes.
- Such models, called classifiers, predict categorical (discrete, unordered) class labels.
- For example, we can build a classification model to categorize bank loan applications as either safe or risky. Such analysis can help provide us with a better understanding of the data at large.
- Data classification is a two-step process, consisting of a learning step (where a classification model is constructed) and a classification step (where the model is used to predict class labels for given data).

Naïve Bayesian Classification:

- Bayesian classifiers are statistical classifiers. They can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.
- Bayesian classification is based on Bayes' theorem.
- Naive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence.
- Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(X|Y) = \frac{P(Y|X)\,P(X)}{P(Y)}$$

Where X and Y are the events and $P\,(Y) \neq 0$

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.

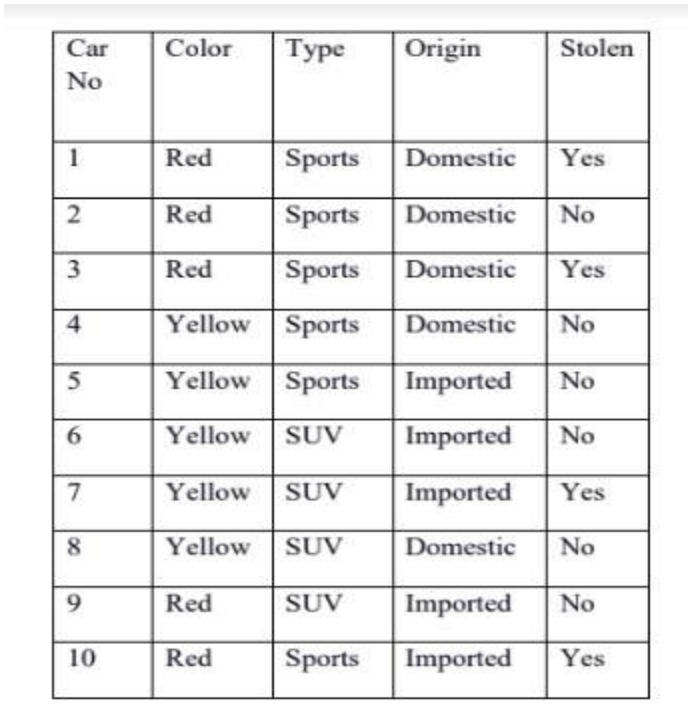
2) Suppose that there are m classes, C1,C2,...,Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naïve Bayesian classifier predicts that tuple X belongs to the class Ci if and only if P(Ci|X) > P(Cj|X) 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

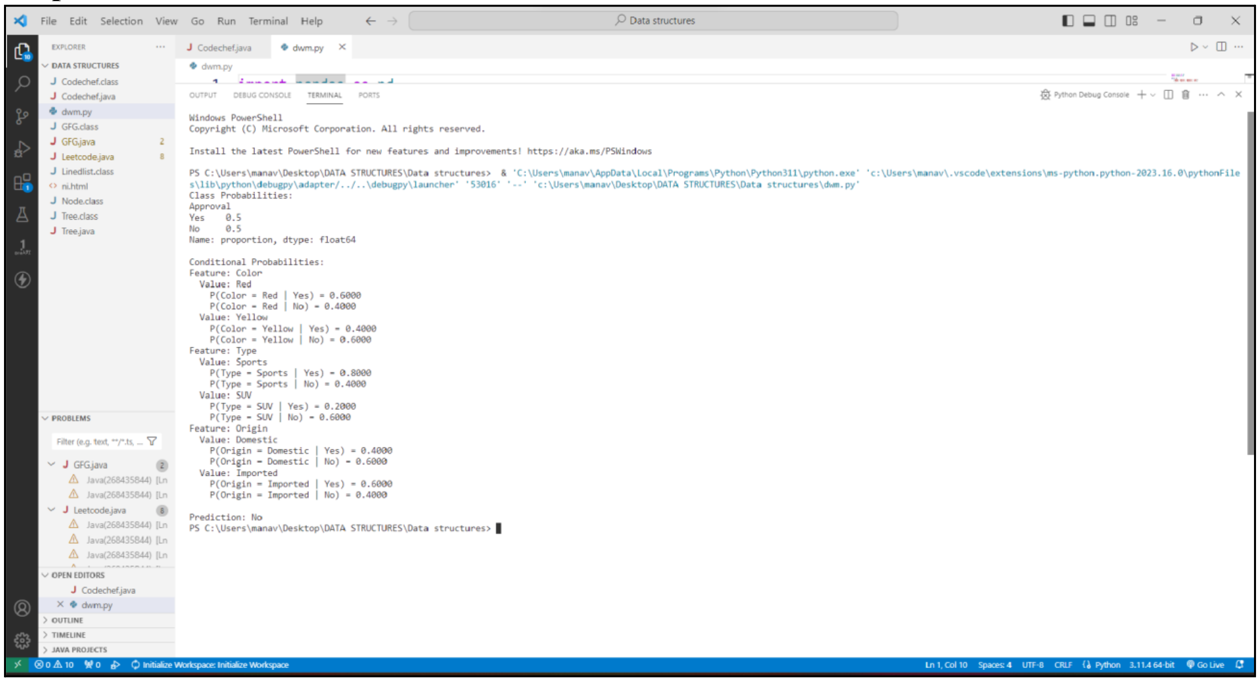
-Origin:
P(Domentic/Y)=2/5 P(Imported/Y)=3/5
P(Domentic /N)=3/5 P(Imported/N)=2/5

P(x|Yes).P(Yes)= 0.024
P(x|No).P(No)=0.072

So, Bayesian Classification Predicts the class “NO” **Program:** import pandas as pd import numpy as np cars = [

```
    ("Red", "Sports", 'Domestic', "Yes"),
    ("Red", "Sports", 'Domestic', "No"),
    ("Red", "Sports", 'Domestic', "Yes"),
    ("Yellow", "Sports", 'Domestic', "No"),
    ("Yellow", "Sports", 'Imported', "Yes"),
    ("Yellow", "SUV", 'Imported', "No"),
    ("Yellow", "SUV", 'Imported', "Yes"),
    ("Yellow", "SUV", 'Domestic', "No"),
    ("Red", "SUV", 'Imported', "No"),
    ("Red", "Sports", 'Imported', "Yes")
]
df = pd.DataFrame(cars, columns=['Color', 'Type', 'Origin',
'Approval']) class_probabilities =
df['Approval'].value_counts(normalize=True) conditional_probabilities
= {} for feature in df.columns[:-1]: conditional_probabilities[feature] =
{} for value in df[feature].unique():
conditional_probabilities[feature][value] = {} for label in
df['Approval'].unique(): subset = df[df[feature] == value]
conditional_probabilities[feature][value][label] =
len(subset[subset['Approval'] == label]) / len(df[df['Approval'] ==
label]) def predict(features): probabilities = {} for label in
df['Approval'].unique():
    probabilities[label] = class_probabilities[label]
    for feature in features:
        probabilities[label] *= conditional_probabilities[feature][features[feature]][label]
return max(probabilities, key=probabilities.get) new_example = {'Color': 'Red', 'Type':
'SUV', 'Origin': 'Domestic'} prediction = predict(new_example) print("Class
Probabilities:") print(class_probabilities) print("\nConditional Probabilities:") for
feature, values in conditional_probabilities.items():
  print(f"Feature: {feature}") for
  value, labels in values.items():
    print(f" Value: {value}") for label,
    probability in labels.items():
      print(f"    P({feature} = {value} | {label}) = {probability:.4f}")
print("\nPrediction:", prediction)
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\manav\Desktop\DATA STRUCTURES\Data structures>  & 'C:\Users\manav\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\manav\.vscode\extensions\ms-python.python-2023.16.0\pythonFile
s\lib\python\debugpy\adapter/../..\debugpy\launcher' '53016' '--' 'c:\Users\manav\Desktop\DATA STRUCTURES\Data structures\dwm.py'
Class Probabilities:
Approval
Yes    0.5
No     0.5
Name: proportion, dtype: float64

Conditional Probabilities:
Feature: Color
  Value: Red
    P(Color = Red | Yes) = 0.6000
    P(Color = Red | No) = 0.4000
  Value: Yellow
    P(Color = Yellow | Yes) = 0.4000
    P(Color = Yellow | No) = 0.6000
Feature: Type
  Value: Sports
    P(Type = Sports | Yes) = 0.8000
    P(Type = Sports | No) = 0.4000
  Value: SUV
    P(Type = SUV | Yes) = 0.2000
    P(Type = SUV | No) = 0.6000
Feature: Origin
  Value: Domestic
    P(Origin = Domestic | Yes) = 0.4000
    P(Origin = Domestic | No) = 0.6000
  Value: Imported
    P(Origin = Imported | Yes) = 0.6000
    P(Origin = Imported | No) = 0.4000

Prediction: No
PS C:\Users\manav\Desktop\DATA STRUCTURES\Data structures>
```

Conclusion:

implementing the Naïve Bayes classifier in Java or Python is a practical approach for performing classification tasks, especially in situations where simplicity, speed, and efficiency are required. While Naïve Bayes is based on "naïve" independence assumptions, it has proven to be a valuable tool in machine learning and data analysis, providing a foundation for many applications where probabilistic classification is needed. Its efficiency, interpretability, and performance in text classification make it a compelling choice for a wide range of real-world applications.