EXPERIMENT – 07

AIM: Demonstration of Naïve Bayesian classifier for a sample training data set stored as a .CSV file. Calculate the accuracy, precision, and recall for your dataset.

DESCRIPTION:

The Naïve Bayes classifier is a probabilistic machine learning model based on Bayes' theorem, commonly used for classification tasks. It assumes that features are conditionally independent given the class label, which simplifies computations.

- 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.

EXAMPLE:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:
P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)
P(Sunny|Yes)= 3/10= 0.3
P(Sunny)= 0.35
P(Yes)=0.71
So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**
P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)
P(Sunny|NO)= 2/4=0.5
P(No)= 0.29
P(Sunny)= 0.35
So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**
So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**
Hence on a Sunny day, Player can play the game.

CODE:

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score

# load data from CSV
data = pd.read_csv('/content/tennisdata.csv')
print("THe first 5 values of data is :\n",data.head())
```

```
THe first 5 values of data is :
     Outlook Temperature Humidity  Windy PlayTennis
0      Sunny         Hot     High  False         No
1      Sunny         Hot     High   True         No
2   Overcast         Hot     High  False        Yes
3      Rainy        Mild     High  False        Yes
4      Rainy        Cool   Normal  False        Yes
```

```
# To Obtain Feature Columns
X = data.iloc[:,:-1]
X.head()
```

	Outlook	Temperature	Humidity	Windy
0	Sunny	Hot	High	False
1	Sunny	Hot	High	True
2	Overcast	Hot	High	False
3	Rainy	Mild	High	False
4	Rainy	Cool	Normal	False

```
y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is\n",y.head())
```

```
The first 5 values of Train output is
 0     No
1     No
2    Yes
3    Yes
4    Yes
Name: PlayTennis, dtype: object
```

```
# Convert then in Numbers
le_outlook = LabelEncoder()
X.Outlook = le_outlook.fit_transform(X.Outlook)

le_Temperature = LabelEncoder()
X.Temperature = le_Temperature.fit_transform(X.Temperature)

le_Humidity = LabelEncoder()
X.Humidity = le_Humidity.fit_transform(X.Humidity)

le_Windy = LabelEncoder()
X.Windy = le_Windy.fit_transform(X.Windy)

print("\nNow the Train data is :\n",X.head())
```

```
Now the Train data is :
    Outlook  Temperature  Humidity  Windy
0        2            1         0      0
1        2            1         0      1
2        0            1         0      0
3        1            2         0      0
4        1            0         1      0
```

```
le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
print("Now the Train output is ",y)
```

```
Now the Train output is  [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25)
```

```
classifier = GaussianNB()
classifier.fit(X_train,y_train)
```

```
GaussianNB
GaussianNB()
```

```
y_pred = classifier.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

```
print("Accuracy: ", accuracy)
print("Precision: ", precision)
print("Recall: ", recall)
```

```
Accuracy:  0.75
Precision:  0.6666666666666666
Recall:  1.0
```

OUTPUT ANALYSIS:

Accuracy

- **Definition**: The proportion of total predictions (both positive and negative) that were correct.
- **Interpretation**: In this example, an accuracy of **0.75** (or 75%) means the classifier correctly classified 75% of the test data instances. This metric gives a general sense of how well the model is performing overall, but it doesn't distinguish between the types of errors made.

Precision

- **Definition**: The ratio of true positive predictions to the total predicted positives.
- **Interpretation**: A precision of **0.66** (or 66%) means that, out of all instances the model predicted as positive (e.g., "Yes"), 66% were indeed positive. Precision reflects the model's ability to avoid false positives, so here, 34% of the positive predictions were incorrect (false positives).

Recall

- **Definition**: The ratio of true positive predictions to the actual positive instances in the dataset.
- **Interpretation**: A recall of **1.00** (or 100%) means the model correctly identified all actual positive instances. This high recall score indicates that the model is very effective at detecting positive instances without missing any (i.e., no false negatives). However, it may have predicted some negatives incorrectly as positives (reflected in the lower precision).

CONCLUSION:

In summary, in this example Naïve Bayes classifier is effective at capturing positive cases, but further optimization may be necessary to improve the precision and overall reliability of the predictions.