PARSHWANATH CHARITABLE TRUST'S

A.P. SHAH INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering
Data Science

.

Academic Year: 2025-26 **Semester: V**
Class/Branch: T.E. DS **Subject: DWMLab**

EXPERIMENT NO. 4

1. **Aim:** To analyze and classify the dataset by implementing the Naive Bayes algorithm using Python.

2. **Objectives:** From this experiment, the student will be able to

   ● Learn about Naive Bayes classification technique.

3. **Theory:**

Naive Bayes classification:

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

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.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:
Suppose we have a dataset of weather conditions and corresponding target variable "Play". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions.

So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Code:

1. Defining Dataset:

In this example, you can use the dummy dataset with three columns: weather, temperature, and play. The first two are features (weather, temperature) and the other is the class label.

```
   # Assigning features and label variables
weather =
['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Su
nny', 'Rainy','Sunny','Overcast','Overcast','Rainy']
temp =
['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild'
,'Mild','Hot','Mild']
play =
['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes'
,'No']
```

2. Encoding Features:

First, you need to convert these string labels into numbers. for example: 'Overcast', 'Rainy', 'Sunny' as 0, 1, 2. This is known as label encoding. Scikit-learn provides the LabelEncoder library for encoding labels with a value between 0 and one less than the number of discrete classes.

```
# Import LabelEncoder
from sklearn import preprocessing

#creating labelEncoder
```

```
le = preprocessing.LabelEncoder()

# Converting string labels into numbers.
weather_encoded=le.fit_transform(weather)

print(weather_encoded)
```

```
      [2 2 0 1 1 1 0 2 2 1 2 0 0 1]
```

Similarly, you can also encode temp and play columns.

```
# Converting string labels into numbers
temp_encoded=le.fit_transform(temp)
label=le.fit_transform(play)

print("Temp:",temp_encoded)
print("Play:",label)
```

```
Temp: [1 1 1 2 0 0 0 2 0 2 2 2 1 2]
      Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

Now combine both the features (weather and temp) in a single variable (list of tuples).

```
#Combinig weather and temp into a single list of tuples
features = zip(weather_encoded, temp_encoded)
X=weather_encoded
Y=temp_encoded
#print(features)
print(list(zip(X, Y)))
features=list(zip(X, Y))
#print([i for i in zip(X, Y)])
```

[(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2), (2, 0), (1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]

3. Generating Model:

Generate a model using Naive Bayes classifier in the following steps:

- Create Naive Bayes classifier

- Fit the dataset on classifier
- Perform prediction

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB#Create a Gaussian Classifier

model = GaussianNB()# Train the model using the training sets
model.fit(features,label)#Predict Output

predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
print("Predicted Value:", predicted)
```

Output:

```
Predicted Value: [1]
```

Here, 1 indicates that players can 'play'.

4. Analysis using Classification Report

```
# Import metrics for evaluation
from sklearn.metrics import classification_report

# Predict for the entire dataset
y_pred = model.predict(features)

# Generate classification report
print("\nClassification Report:")
print(classification_report(label, y_pred, target_names=['No', 'Yes']))
```

4. Conclusion: