**7. Demonstration of the Naive Bayes Classification Algorithm using a basic example.**

**Description:**

1. **Data Preparation**:
   * The dataset is created as a dictionary and then converted to a pandas DataFrame.
   * We map categorical variables to numeric values using .map() so that Naive Bayes can handle the features (since it works with numeric data).
2. **Features and Target Variable**:
   * X represents the features (Outlook, Temperature).
   * y represents the target variable (Play Tennis).
3. **Splitting Data**:
   * We split the data into training and testing sets using train\_test\_split(). This helps us evaluate the model's performance on unseen data.
4. **Model Training**:
   * We use GaussianNB from sklearn, which is the Naive Bayes model that assumes features follow a Gaussian (normal) distribution.
5. **Model Evaluation**:
   * We make predictions using model.predict() and evaluate accuracy using accuracy\_score().

**Step 1: The Naive Bayes Formula**

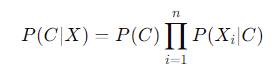
Naive Bayes relies on **Bayes' Theorem**, which can be expressed as:

P(C∣X)=P(X∣C)⋅P(C)/P(X)

Where:

* P(C∣X) is the probability of the class CCC given the features XXX.
* P(X∣C) is the probability of the features XXX given the class CCC.
* P(C) is the prior probability of the class.
* P(X) is the probability of the features.

For Naive Bayes, we make a "naive" assumption that all features are independent, so the formula simplifies to:



Where Xi represents each individual feature.

**Step 2: Example Dataset**

Let’s say we have a small dataset of whether someone plays tennis based on two features: **Outlook** (Sunny or Overcast) and **Temperature** (Hot or Mild). The target variable is whether they **Play Tennis** (Yes or No).

| **Outlook** | **Temperature** | **Play Tennis** |
| --- | --- | --- |
| Sunny | Hot | No |
| Sunny | Mild | No |
| Overcast | Hot | Yes |
| Overcast | Mild | Yes |
| Sunny | Hot | No |
| Overcast | Mild | Yes |

**Step 3: Calculate the Prior Probabilities**

We need to calculate the prior probabilities for the target class, **Play Tennis**.

* P(Play Tennis = Yes) = Number of "Yes" / Total = 3/6 = 0.5
* P(Play Tennis = No) = Number of "No" / Total = 3/6 = 0.5

**Step 4: Calculate the Likelihood Probabilities (Conditional Probabilities)**

We also need to calculate the likelihood of each feature given the class. For instance, we calculate the probability of "Outlook = Sunny" given "Play Tennis = Yes" (and so on for all feature-class combinations).

**For Play Tennis = Yes:**

* P(Outlook = Sunny | Play Tennis = Yes) = 0 (because there is no "Sunny" when "Play Tennis = Yes")
* P(Outlook = Overcast | Play Tennis = Yes) = 1 (because "Overcast" is always there when "Play Tennis = Yes")
* P(Temperature = Hot | Play Tennis = Yes) = 0.33 (1 out of 3)
* P(Temperature = Mild | Play Tennis = Yes) = 0.67 (2 out of 3)

**For Play Tennis = No:**

* P(Outlook = Sunny | Play Tennis = No) = 1 (because "Sunny" is always there when "Play Tennis = No")
* P(Outlook = Overcast | Play Tennis = No) = 0 (because there is no "Overcast" when "Play Tennis = No")
* P(Temperature = Hot | Play Tennis = No) = 0.67 (2 out of 3)
* P(Temperature = Mild | Play Tennis = No) = 0.33 (1 out of 3)

**Step 5: Making Predictions**

Now, let's say we want to predict whether someone will play tennis given **Outlook = Sunny** and **Temperature = Hot**.

For **Play Tennis = Yes**:

P(PlayTennis=Yes∣Outlook=Sunny,Temperature=Hot)=P(PlayTennis=Yes)⋅P(Outlook=Sunny∣PlayTennis=Yes)⋅P(Temperature=Hot∣PlayTennis=Yes)P(Play Tennis = Yes | Outlook = Sunny, Temperature = Hot) = P(Play Tennis = Yes) \cdot P(Outlook = Sunny | Play Tennis = Yes) \cdot P(Temperature = Hot | Play Tennis = Yes)P(PlayTennis=Yes∣Outlook=Sunny,Temperature=Hot)=P(PlayTennis=Yes)⋅P(Outlook=Sunny∣PlayTennis=Yes)⋅P(Temperature=Hot∣PlayTennis=Yes) =0.5⋅0⋅0.33=0

For **Play Tennis = No**:

P(PlayTennis=No∣Outlook=Sunny,Temperature=Hot)=P(PlayTennis=No)⋅P(Outlook=Sunny∣PlayTennis=No)⋅P(Temperature=Hot∣PlayTennis=No)P(Play Tennis = No | Outlook = Sunny, Temperature = Hot) = P(Play Tennis = No) \cdot P(Outlook = Sunny | Play Tennis = No) \cdot P(Temperature = Hot | Play Tennis = No)P(PlayTennis=No∣Outlook=Sunny,Temperature=Hot)=P(PlayTennis=No)⋅P(Outlook=Sunny∣PlayTennis=No)⋅P(Temperature=Hot∣PlayTennis=No) =0.5⋅1⋅0.67=0.335

**Step 6: Conclusion**

Since the probability of **Play Tennis = No** is higher (0.335) compared to **Play Tennis = Yes** (0), we predict that the person **will not play tennis** given **Outlook = Sunny** and **Temperature = Hot**.

**Python Code for Naive Bayes Classification**

import pandas as pd

from sklearn.naive\_bayes import GaussianNB

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Sample dataset

data = {

'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Overcast', 'Sunny', 'Overcast'],

'Temperature': ['Hot', 'Mild', 'Hot', 'Mild', 'Hot', 'Mild'],

'Play Tennis': ['No', 'No', 'Yes', 'Yes', 'No', 'Yes']

}

# Create a DataFrame

df = pd.DataFrame(data)

# Convert categorical data to numeric (encoding)

df['Outlook'] = df['Outlook'].map({'Sunny': 0, 'Overcast': 1})

df['Temperature'] = df['Temperature'].map({'Hot': 0, 'Mild': 1})

df['Play Tennis'] = df['Play Tennis'].map({'No': 0, 'Yes': 1})

# Define features (X) and target variable (y)

X = df[['Outlook', 'Temperature']] # Features

y = df['Play Tennis'] # Target variable

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Initialize and train the Naive Bayes classifier

model = GaussianNB()

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Print predictions

print("Predictions:", y\_pred)

**Out Put:**