EXPERIMENT - 10

Objective: IMPLEMENT NAIVE BAYES MODELS

Theory :

Naive Bayes is a family of probabilistic classifiers based on **Bayes' Theorem** with a **strong (naive) independence assumption** between features :

1. **Bayes' Theorem (core)**

P(y| x)=P(x| y)P(y)P(x)P(y \mid x) = \dfrac{P(x \mid y) P(y)}{P(x)}P(y| x)=P(x)P(x| y)P(y) For classification we compare P(y)P(x| y)P(y)P(x\mid y)P(y)P(x| y) across classes

2. **Naive assumption**

Features x=(x1,…,xn)x = (x_1,\dots,x_n)x=(x1,…,xn) are assumed conditionally independent given class yyy:

P(x| y)=∏iP(xi| y)P(x\mid y)=\prod_i P(x_i\mid y)P(x| y)=∏iP(xi| y).

3. **Common variants**

- Gaussian NB — continuous features modeled with class-conditional Gaussian distributions (estimate mean & variance per feature per class).
- Multinomial NB — counts/features (e.g., word counts); models P(x| y)P(x\mid y)P(x| y) with class-specific multinomial distribution (uses Laplace smoothing)

4. **Training (estimation)**

Estimate prior: P(y)=count(y)NP(y)=\frac{\text{count}(y)}{N}P(y)=Ncount(y).

5. Prediction

- For each class compute score sy=logP(y)+∑ilogP(xi∣ y)s_y = \log P(y) + \sum_i \log P(x_i \mid y)sy=logP(y)+∑ilogP(xi∣ y).
  Predict class with highest score.

Pros & Cons

- **Pros:** fast, simple, works well in practice (especially for text).
- **Cons:** strong independence assumption may be violated; relies on correct likelihood model (e.g., Gaussian may be bad for heavy-tailed data).

Practical tips

- Use **Laplace smoothing** for discrete counts to avoid zero probabilities.
- Standardize features for Gaussian NB if scales differ greatly (though NB uses per-class variances).
- Always use **log-probabilities**.
- Evaluate with cross-validation and a confusion matrix.

Algorithm:

Step 1: Collect the Data

- Take your training data with inputs (features) and their class labels.

Step 2: Separate the Data by Class

- Group the data according to each class (like “Yes” or “No”).
- This helps you count how often each feature value appears in each class.

Step 3: Find Prior Probabilities

- Find how likely each class is overall.

  Example: If 3 out of 5 times Play = “Yes”, then

  → P(Yes) = 3/5, P(No) = 2/5

Step 4: Find Conditional Probabilities

- For each feature (like Outlook = Sunny), count how often it appears in each class.
- Example: P(Sunny | Yes) = Number of “Sunny” days when Play = Yes ÷ Total “Yes” cases.

Step 5: Multiply the Probabilities :

For a new case (like “Sunny” and “Cool”), multiply all probabilities for each class:

P(Yes)×P(Sunny| Yes)×P(Cool| Yes)P(Yes) × P(Sunny|Yes) ×

P(Cool|Yes)P(Yes)×P(Sunny| Yes)×P(Cool| Yes)

P(No)×P(Sunny| No)×P(Cool| No)P(No) × P(Sunny|No) × P(Cool|No)P(No)×P(Sunny| No)×P(Cool| No)

Step 6: Compare the Results :

Whichever class gives the higher probability, that becomes your predicted class.

Example:

If P(Yes| data)>P(No| data)P(Yes|data) > P(No|data)P(Yes| data)>P(No| data), then predict “Yes”.

Step 7: End

- Output the predicted class for the given input.

Code :

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
```

```
# Load the Iris dataset
iris = load_iris()

X = iris.data # Features

y = iris.target # Target variable (species)

# Split the data 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 the Gaussian Naive Bayes model

gnb = GaussianNB()

# Train the model

gnb.fit(X_train, y_train)

# Make predictions on the test set

y_pred = gnb.predict(X_test)



# Evaluate the model

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy of Gaussian Naive Bayes: {accuracy:.2f}")

[08/11/25, 12:40:47 AM] Sampada: # Example of making a single prediction

new_data_point = [[5.1, 3.5, 1.4, 0.2]] # Example measurements for a new iris flower

predicted_species = gnb.predict(new_data_point)

print(f"Predicted species for {new_data_point}: {iris.target_names[predicted_species[0]]}")
```

Output:

```
Accuracy of Gaussian Naive Bayes: 0.98
Predicted species for [[5.1, 3.5, 1.4, 0.2]]: setosa
```