Exercise 3 Date: 05/02/2025

Naïve Bayes Classifier

Aim: To build a Naïve Bayes Classifier to predict survival on the Titanic dataset.

Algorithm:

Naïve Bayes is a supervised learning algorithm based on applying Bayes' theorem with the naïve assumption that the features are independent.The key idea of Naïve Bayes is to compute the posterior probability for each class and select the class with the highest probability.

$$P(C_k|x) = \frac{P(x|C_k)\,P(C_k)}{P(x)}$$

Where:

- $(P(C_k|x))$ is the *posterior probability*, representing the probability of class (C_k) given the observed feature (x).
- $P(x|C_k)$ is the *likelihood*, indicating the probability of observing the feature (x) given that the class is (C_k).
- $P(C_k)$ is the *prior probability* of class (C_k), representing the overall probability of this class before considering the feature.
- $P(x)$ is the *evidence* or *marginal likelihood*, which is the probability of observing the feature x under all classes.

Step 1: Import Libraries

- Import necessary Python libraries such as pandas for data manipulation, and specific modules from scikit-learn (train_test_split, GaussianNB, and evaluation metrics) for model building and evaluation.

Step 2: Load the Dataset

- Load the dataset using pd.read_csv() and inspect the structure and details of the dataset using df.info().

Step 3: Prepare the Data

- Drop Irrelevant Columns: Remove columns that are not useful for prediction. For example, the Name column is dropped because it does not provide predictive information.
- Encode Categorical Variables: Convert categorical features (like Sex) to numerical values (e.g., male: 0, female: 1).
- Feature Selection: Define the feature matrix X (all predictors) and the target vector y (the variable to predict, i.e., Survived).

Step 4: Train-Test Split

Split the dataset into training and testing sets using train_test_split.

Step 5: Train the Model

- Initialize the Gaussian Naïve Bayes classifier.
- Fit the model using the training data (X_train and y_train). The classifier learns the conditional probabilities for each feature given the class label.

Step 6: Make Predictions

- Use the trained model to make predictions on the test data (X_test).
- The predicted labels are stored in y_pred.

Step 7: Evaluate the Model

- Evaluate the performance of the model using key metrics:
  - Accuracy Score: Measures the proportion of correct predictions.
  - Confusion Matrix: Provides detailed insight into true positives, true negatives, false positives, and false negatives.
  - Classification Report: Summarizes precision, recall, f1-score, and support for each class.

Import the libraries

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Load the Dataset

In [2]:
```
df = pd.read_csv('titanic.csv')
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 887 entries, 0 to 886
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Survived                 887 non-null    int64  
 1   Pclass                   887 non-null    int64  
 2   Name                     887 non-null    object 
 3   Sex                      887 non-null    object 
 4   Age                      887 non-null    float64
 5   Siblings/Spouses Aboard  887 non-null    int64  
 6   Parents/Children Aboard  887 non-null    int64  
 7   Fare                     887 non-null    float64
dtypes: float64(2), int64(4), object(2)
memory usage: 55.6+ KB
```

In [4]:
```
df.isna().sum()
```

Out[4]:
```
Survived                   0
Pclass                     0
Name                       0
Sex                        0
Age                        0
Siblings/Spouses Aboard    0
Parents/Children Aboard    0
Fare                       0
dtype: int64
```

Prepare the data

In [5]:
```
df = df.drop(columns=['Name'])
```

In [6]:
```
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
```

Split features and target

In [8]:
```
X = df.drop('Survived', axis=1)
y = df['Survived']
```

Split dataset into train and test data

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y
)
```

Train the Naïve Bayes Model

In [10]:
```
model = GaussianNB()
```

In [11]:
```
model.fit(X_train, y_train)
```

Out[11]:
```
▾ GaussianNB
GaussianNB()
```

In [12]:
```
y_pred = model.predict(X_test)
```

Performance metrics

In [13]:
```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

In [14]:
```
print("\nNaïve Bayes Classifier Performance:")
print(f"Accuracy: {accuracy:.4f}\n")
print("Confusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)
```

```
Naïve Bayes Classifier Performance:
Accuracy: 0.7790

Confusion Matrix:
[[138  26]
 [ 33  70]]

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.84      0.82       164
           1       0.73      0.68      0.70       103

    accuracy                           0.78       267
   macro avg       0.77      0.76      0.76       267
weighted avg       0.78      0.78      0.78       267
```

Result

A Naïve Bayes Classifier was built to predict survival based on multiple features from the Titanic dataset, achieving an accuracy of 77.90%.