NAIVE Bayes algorithm

Naive Bayes Algorithm is a classification algorithm based on Bayes Theorem. It is called naive because it assumes that the features in a dataset are independent of each other. This assumption is not true in real life but it simplifies the computation and gives good results in most of the cases.

Bayes Theorem

Bayes Theorem is a mathematical formula used for calculating conditional probability. It is defined as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where A and B are events and P(B) != 0

Naive Bayes Algorithm

Naive Bayes Algorithm is based on Bayes Theorem. It is defined as:

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(x_1, x_2, \ldots, x_n|y)P(y)}{P(x_1, x_2, \ldots, x_n)}$$

where y is the class variable and x1, x2, ..., xn are the features.

The algorithm assumes that the features are independent of each other. So, the above equation can be written as

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(x_1, x_2, \ldots, x_n)}$$

The denominator is constant for a given input. So, the equation can be written as:

$$P(y|x_1, x_2, \ldots, x_n) \propto P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)$$

The class with the highest probability is the output of the algorithm.

In [1]:
```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.datasets import load_iris
```

In [2]:
```
# load the dataset
iris = load_iris()
X = iris.data
y = iris.target

# train test split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [3]:
```
# model initialize
gnb = GaussianNB()

# train the model
gnb.fit(X_train, y_train)

# predict the test data
y_pred = gnb.predict(X_test)
```

```
# evaluate the model
print("Accuracy Score: ", accuracy_score(y_test, y_pred))
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
print("Classification Report: \n", classification_report(y_test, y_pred))
```

```
Accuracy Score:  0.9777777777777777
Confusion Matrix:
 [[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.92      0.96        13
           2       0.93      1.00      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45
```