Name :- Vijay Gurung

ID Number : MST03 – 0070

*** Topic : Naive_Bayes_Algoritm ***

<u>Introduction</u> :

Naive Bayes is a probabilistic machine learning algorithm based on Bayes' Theorem, used primarily for classification tasks. It is termed **"naive"** because it assumes that the features in a dataset are independent of each other, which is rarely the case in real-world scenarios. Despite this assumption, Naive Bayes classifiers perform surprisingly well in many practical applications.

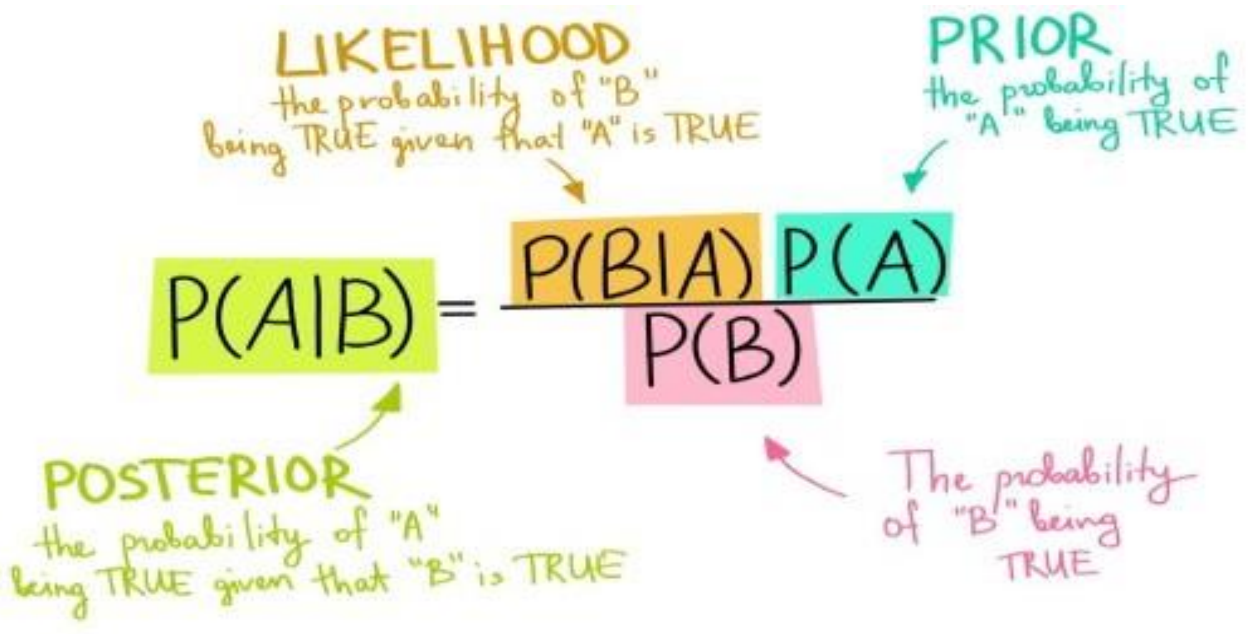
<u>Bayes' Theorem</u> :

Bayes' Theorem is the foundation of the Naive Bayes algorithm. It describes the probability of an event, based on prior knowledge of conditions that might be related to the event. The theorem is expressed as:

P(A|B)=P(B|A)·P(A)P(B)P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}P(A|B)=P(B)P(B|A)·P(A)

Where:

- P(A|B)P(A|B)P(A|B) is the posterior probability of class AAA given predictor BBB.
- P(B|A)P(B|A)P(B|A) is the likelihood of predictor BBB given class AAA.
- P(A)P(A)P(A) is the prior probability of class AAA.
- P(B)P(B)P(B) is the prior probability of predictor BBB.



Naive Bayes Classifier :

The Naive Bayes classifier applies Bayes' Theorem with the **"naive"** assumption of independence among predictors. The formula for the classifier becomes:

P(Ck|x1,x2,...,xn)=P(x1|Ck)·P(x2|Ck)·...·P(xn|Ck)·P(Ck)P(x1)·P(x2)·...·P(xn)P(C_k|x_1, x_2, ..., x_n) = \frac{P(x_1|C_k) \cdot P(x_2|C_k) \cdot ... \cdot P(x_n|C_k) \cdot P(C_k)}{P(x_1) \cdot P(x_2) \cdot ... \cdot P(x_n)}P(Ck|x1,x2,...,xn)=P(x1)·P(x2)·...·P(xn)P(x1|Ck)·P(x2|Ck)·...·P(xn|Ck)·P(Ck)

Where:

- P(Ck|x1,x2,...,xn)P(C_k|x_1, x_2, ..., x_n)P(Ck|x1,x2,...,xn) is the posterior probability of class CkC_kCk given predictors x1,x2,...,xnx_1, x_2, ..., x_nx1,x2,...,xn.
- P(xi|Ck)P(x_i|C_k)P(xi|Ck) is the likelihood of predictor xix_ixi given class CkC_kCk.
- P(Ck)P(C_k)P(Ck) is the prior probability of class CkC_kCk.
- P(xi)P(x_i)P(xi) is the prior probability of predictor xix_ixi.

Types of Naive Bayes Classifiers :

1. **Gaussian Naive Bayes:** Assumes that the continuous values associated with each feature are distributed according to a Gaussian (normal) distribution.
2. **Multinomial Naive Bayes:** Used for discrete data, commonly used for document classification problems.
3. **Bernoulli Naive Bayes:** Used for binary/boolean features.

Example:

Spam Email Classification

Let's consider a simple example of classifying emails as **"Spam" or "Not Spam"** based on the occurrence of certain words.

Step-by-Step Process :

1. **Prepare the Data:** Suppose we have a training dataset with the following emails:

Email Text	Class
Buy cheap meds	Spam
Win a free lottery	Spam
Meeting tomorrow	Not Spam
Project deadline	Not Spam

2. **Extract Features:** We consider each word as a feature. For simplicity, assume our vocabulary consists of "buy", "cheap", "meds", "win", "free", "lottery", "meeting", "tomorrow", "project", "deadline".

3. **Calculate Probabilities:**

   Calculate the prior probabilities for each class:
   P(Spam)=Number of Spam EmailsTotal Emails=24=0.5P(Spam) = \frac{\text{Number of Spam Emails}}{\text{Total Emails}} = \frac{2}{4} = 0.5P(Spam)=Total EmailsNumber of Spam Emails =42=0.5 P(Not Spam)=Number of Not Spam EmailsTotal Emails=24=0.5P(Not\ Spam) = \frac{\text{Number of Not Spam Emails}}{\text{Total Emails}} = \frac{2}{4} = 0.5P(Not Spam)=Total EmailsNumber of Not Spam Emails=42=0.5

Calculate the likelihood probabilities for each word given the class. For example:
P(buy|Spam)=Count of "buy" in SpamTotal words in Spam=16P(buy|Spam) = \frac{\text{Count of "buy" in Spam}}{\text{Total words in Spam}} = \frac{1}{6}P(buy|Spam)=Total words in SpamCount of "buy" in Spam=61
P(meeting|Not Spam)=Count of "meeting" in Not SpamTotal words in Not Spam=16P(meeting|Not\ Spam) = \frac{\text{Count of "meeting" in Not Spam}}{\text{Total words in Not Spam}} = \frac{1}{6}P(meeting|Not Spam)=Total words in Not SpamCount of "meeting" in Not Spam=61

4.Classify a New Email: Suppose we have a new email: "win meds".

Calculate the posterior probabilities for each class:
P(Spam|win meds)∝P(win|Spam)·P(meds|Spam)·P(Spam)P(Spam|win\ meds) \propto P(win|Spam) \cdot P(meds|Spam) \cdot P(Spam)P(Spam|win meds)∝P(win|Spam)·P(meds|Spam)·P(Spam)
P(Spam|win meds)∝16·16·0.5=0.0139P(Spam|win\ meds) \propto \frac{1}{6} \cdot \frac{1}{6} \cdot 0.5 = 0.0139P(Spam|win meds)∝61·61·0.5=0.0139

P(Not Spam|win meds)∝P(win|Not Spam)·P(meds|Not Spam)·P(Not Spam)P(Not\ Spam|win\ meds) \propto P(win|Not\ Spam) \cdot P(meds|Not\ Spam) \cdot P(Not\ Spam)P(Not Spam|win meds)∝P(win|Not Spam)·P(meds|Not Spam)·P(Not Spam)
P(Not Spam|win meds)∝06·06·0.5=0P(Not\ Spam|win\ meds) \propto \frac{0}{6} \cdot \frac{0}{6} \cdot 0.5 = 0P(Not Spam|win meds)∝60·60·0.5=0

Since P(Spam|win meds)>P(Not Spam|win meds)P(Spam|win\ meds) > P(Not\ Spam|win\ meds)P(Spam|win meds)>P(Not Spam|win meds), the email is classified as "Spam".

Conclusion :

Naive Bayes is a simple yet powerful algorithm suitable for text classification, spam detection, sentiment analysis, and more. Its assumptions of feature independence and simplicity make it computationally efficient, but it's essential to consider the context of the problem to ensure it’s the right choice.