Naïve Bayes Algorithm -Implementation from scratch in Python.

Never tell me the odds … .without first establishing a Bayesian Prior.

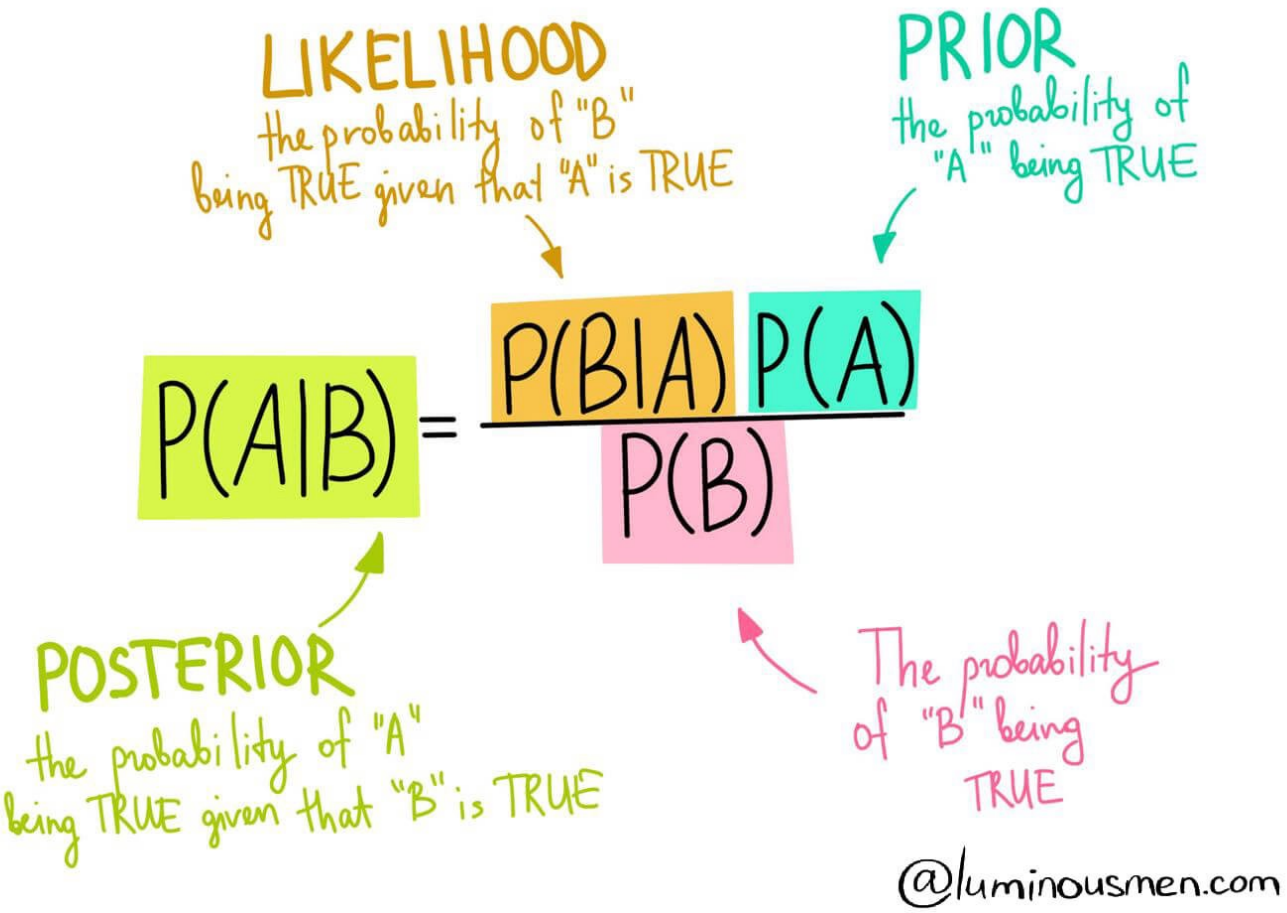
Photo by Moritz Kindler on Unsplash

Introduction

Naïve Bayes algorithm is a supervised classification algorithm based on **Bayes theorem** with strong(**Naïve**) **independence among features.**

Bayes' Theorem

In probability theory and statistics, **Bayes' theorem** describes the probability of an event, based on prior knowledge of conditions that might be related to the event — *Source: Wikipedia*



Bayes' Theorem

Naive Bayes Classifier formula can be written based on Bayes theorem as:

$$P(y \mid x_1, \cdots, x_j) = \frac{P(x_1, \cdots x_j \mid y)P(y)}{P(x_1, \cdots, x_j)}$$

Naive Bayes Classifier Formula

Where,

- **x1, … , xj** are j features that are independent of each other. **y** is the dependent variable.
- P(y|x1,…, xj): **Posterior Probability**
- P(x1, …, xj|y): **Likelihood** of features x1 to xj given that their class is y.

- P(y): **Prior Probability**
- P(x1, …, xj): **Marginal Probability**

How Does Naïve Bayes Algorithm Works?

Let's understand through an example:

Step 1: We start by importing dataset and necessary dependencies

We will be using the weather dataset for training. This dataset includes features [Outlook, Temp, Humidity, Windy], and the corresponding target variable 'Play'. Now, we need to predict whether players will play or not based on given weather conditions.

```
#Weather Dataset

Outlook   Temp    Humidity   Windy  Play
Rainy     Hot     High       f      no
Rainy     Hot     High       t      no
Overcast  Hot     High       f      yes
Sunny     Mild    High       f      yes
Sunny     Cool    Normal     f      yes
Sunny     Cool    Normal     t      no
Overcast  Cool    Normal     t      yes
Rainy     Mild    High       f      no
Rainy     Cool    Normal     f      yes
Sunny     Mild    Normal     f      yes
Rainy     Mild    Normal     t      yes
Overcast  Mild    High       t      yes
Overcast  Hot     Normal     f      yes
Sunny     Mild    High       t      no
```

Step1: Loading Dataset

Step 2: Calculate Prior Probability of Classes P(y)

```
#Frequency table

P(Play=Yes) = 9/14 = 0.64
P(Play=No) = 5/14 = 0.36
```

Prior Probability Calculation Function

Step 3: Calculate the Likelihood Table for all features

```
#Likelihood Table
```

#Outlook

Play	**Overcast**	**Rainy**	**Sunny**
Yes	4/9	2/9	3/9
No	0/5	3/5	2/5
	4/14	5/14	5/14

#Temp

Play	**Cool**	**Mild**	**Hot**
Yes	3/9	4/9	2/9
No	1/5	2/5	2/5
	4/14	6/14	4/14

#Humidity

Play	**High**	**Normal**
Yes	3/9	6/9
No	4/5	1/5
	7/14	7/14

#Windy

Play	**f**	**t**
Yes	6/9	3/9
No	2/5	3/5
	8/14	6/14

Step 4: Now, Calculate Posterior Probability for each class using the Naive Bayesian equation. The Class with maximum probability is the outcome of the prediction.

Query: Whether Players will play or not when the weather conditions are [Outlook=Rainy, Temp=Mild, Humidity=Normal, Windy=t]?

Calculation of Posterior Probability:

$$P(y=Yes|x) = P(Yes|Rainy,Mild,Normal,t)$$

$$= \frac{P(Rainy,Mild,Normal,t|Yes) * P(Yes)}{P(Rainy,Mild,Normal,t)}$$

$$= \frac{P(Rainy|Yes)*P(Mild|Yes)*P(Normal|Yes)*P(t|Yes)*P(Yes)}{P(Rainy)*P(Mild)*P(Normal)*P(t)}$$

Since Conditional independence of two random variables, A and B gave C holds just in case

P(A, B | C) = P(A | C) * P(B | C)

$$= \frac{(2/9) * (4/9) * (6/9) * (3/9) * (9/14)}{(5/14) * (6/14) * (7/14) * (6/14)}$$

$$= \mathbf{0.43}$$

$$P(y=No|x) = P(No|Rainy,Mild,Normal,t)$$

$$= \frac{P(Rainy,Mild,Normal,t|No) * P(No)}{P(Rainy,Mild,Normal,t)}$$

$$= \frac{P(Rainy|No)*P(Mild|No)*P(Normal|No)*P(t|No)*P(No)}{P(Rainy)*P(Mild)*P(Normal)*P(t)}$$

$$= \frac{(3/5) * (2/5) * (1/5) * (3/5) * (5/14)}{(5/14) * (6/14) * (7/14) * (6/14)}$$

$$= \mathbf{0.31}$$

Now, **P(Play=Yes|Rainy,Mild,Normal,t**) has the highest Posterior probability**.**

From the above calculation, we can say that there is a high probability for the players **to Play** in the given weather condition i.e., data belongs to a class **Yes.**

Complete Source Code of Naïve Bayes Classifier:

Here's the Output:

Weather Dataset:

Train Accuracy: 92.86

```
Query 1:- [['Rainy' 'Mild' 'Normal' 't']] ---> ['yes']
Query 2:- [['Overcast' 'Cool' 'Normal' 't']] ---> ['yes']
Query 3:- [['Sunny' 'Hot' 'High' 't']] ---> ['no']
```