**Naive Bayes Theorem**

**What is Probability?**

To understand the Naive Bayes Classifier from scratch, it is required to understand the term probability, as the algorithm itself works on the concept of probabilities of events. Let us try to understand the same.

Probability is the thing or term called in [mathematics](https://www.analyticsvidhya.com/blog/2021/06/how-to-learn-mathematics-for-machine-learning-what-concepts-do-you-need-to-master-in-data-science/) the “chance of something to take place”. In simple words, “the probability is a chance of some event to occur.”

We know that the sum of all probabilities is always one, and for Example, if we toss the coin in the air, the possibility is the head is 0.5 and the tails are also 0.5, which means that there is an equal, and 50% chance of heads and tails to come for the first trial.

**Mutually exclusive events** are those events that do not occur at the same time. For example, when a coin is tossed then the result will be either head or tail, but we cannot get both the results. Such events are also called disjoint events since they do not happen simultaneously. If A and B are mutually exclusive events then its probability is given by P(A Or B) or P (A U B). Let us learn the formula of P (A U B) along with rules and examples here in this article.

**What are Mutually Exclusive Events?**

In probability theory, two events are said to be mutually exclusive if they cannot occur at the same time or simultaneously. In other words, **mutually exclusive events** are called disjoint events. If two events are considered disjoint events, then the probability of both events occurring at the same time will be zero.

If A and B are the two events, then the probability of disjoint of event A and B is written by:

**Probability of Disjoint (or) Mutually Exclusive Event = P ( A and B) = 0**

**How to Find Mutually Exclusive Events?**

In probability, the specific addition rule is valid when two events are mutually exclusive. It states that the probability of either event occurring is the sum of probabilities of each event occurring. If A and B are said to be mutually exclusive events then the probability of an event A occurring or the probability of event B occurring that is P (a ∪ b) formula is given by P(A) + P(B), i.e.,

* **P (A Or B) = P(A) + P(B)**
* **P (A ∪ B) = P(A) + P(B)**

**Note:**

If the events A and B are not mutually exclusive, the probability of getting A or B that is P (A ∪ B) formula is given as follows:

P (A ∪ B) = P(A) + P(B) – P (A and B)

**Real-life Examples on Mutually Exclusive Events**

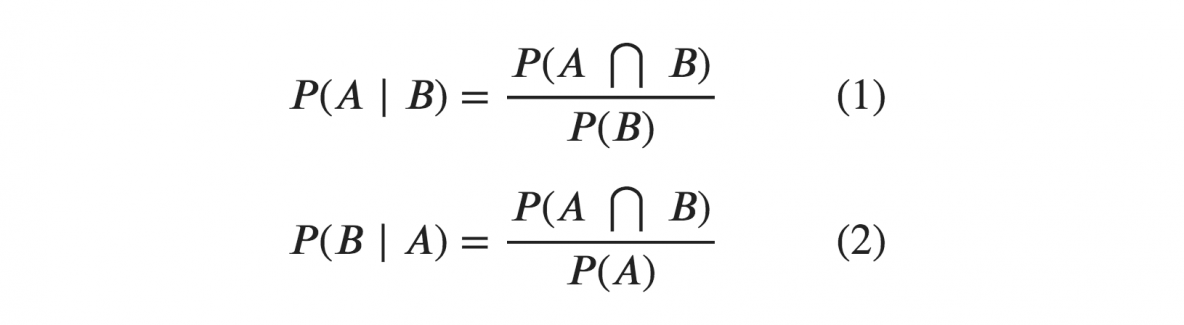
Some of the examples of the mutually exclusive events are:

* When tossing a coin, the event of getting head and tail are mutually exclusive. Because the probability of getting head and tail simultaneously is 0.
* In a six-sided die, the events “2” and “5” are mutually exclusive. We cannot get both the events 2 and 5 at the same time when we threw one die.
* In a deck of 52 cards, drawing a red card and drawing a club are mutually exclusive events because all the clubs are black.

**What is Conditional Probability?**

Now we know the meaning of probability, the next term to understand is conditional probability. [Conditional probability](https://www.analyticsvidhya.com/blog/2017/03/conditional-probability-bayes-theorem/) is defined as the probability of some event happening with respect to another event. In simple words, conditional probability is also a probability of some things occurring when a condition is involved.

The formula for the Conditional Probability is:



Source- Machine learning plus

P(A\*B) = Probability of events A and B both happening

P(A) = Probability of event A to occur.

P(B) = Probability of event B to occur.

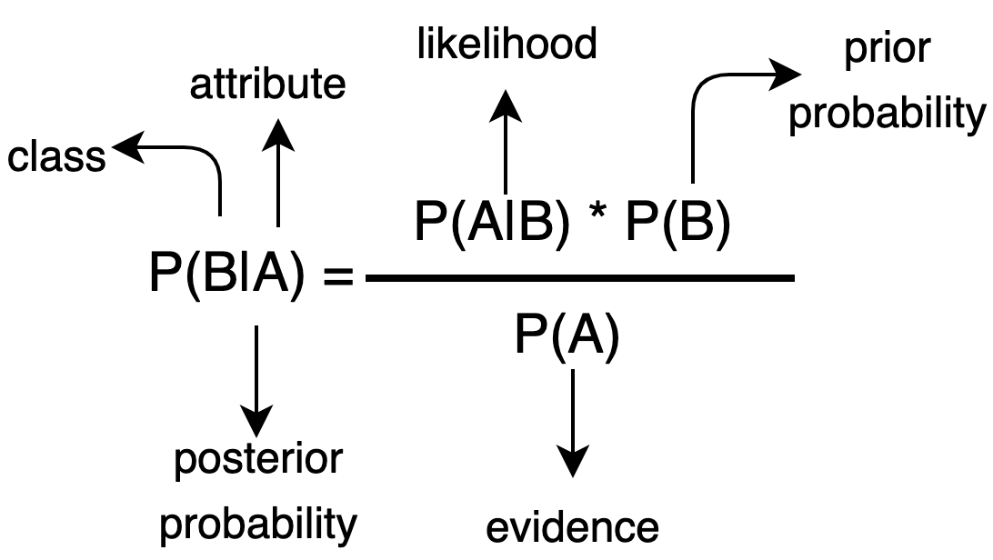
P(A|B) = Probability of event A happening when event B occurs.

P(B|A) = Probability of event B happening when event A occurs.

**Bayes Rule**

Now, we are prepared to learn the bayesian rule after knowing the two critical terms. Thomas Bayes, a British mathematician in 1763, gave the bayesian theorem, which helped calculate the probability of some events taking place with conditions.

The formula for Bayes Rule is:



Source- Medium

As we can see in the above image, the formula comprises a total of 4 terms. Let us try to understand them one by one.

P(B|A) = Probability of event B to happen when event A occurs.

P(A|B) = Probability of event A to happen when event B occurs.

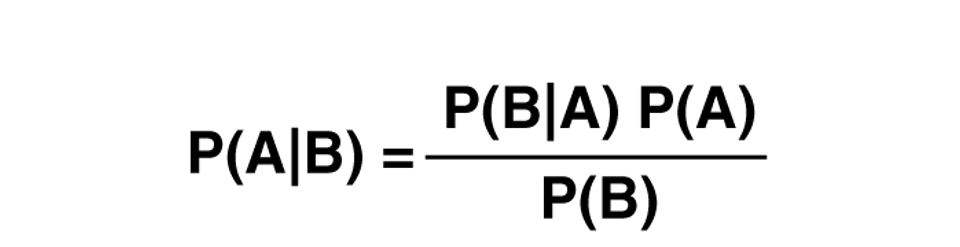
P(A) = Probability of event A to occur.

P(B) = Probability of event B to occur.

From the above formula, we can easily calculate the probability of some event happening with the condition if we have the average likelihood of vents happening and both events happening.

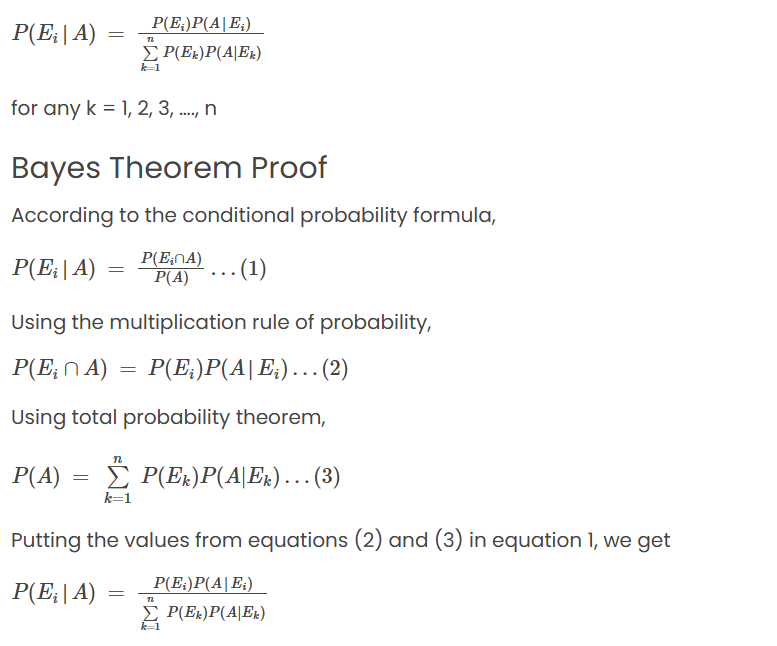
**What is Bayes’ theorem in probability:**

**Bayes’ theorem** describes the probability of occurrence of an event related to any condition. It is also considered for the case of [conditional probability](https://byjus.com/maths/conditional-probability-and-conditional-probability-examples/). Bayes theorem is also known as the formula for the probability of “causes”. For example: if we have to calculate the probability of taking a blue ball from the second bag out of three different bags of balls, where each bag contains three different colour balls viz. red, blue, black. In this case, the probability of occurrence of an event is calculated depending on other conditions is known as conditional probability. In this article, let us discuss the statement and proof for Bayes theorem, its derivation, formula, and many solved examples.



**Bayes Theorem Statement**

Let E1, E2,…, En be a set of events associated with a sample space S, where all the events E1, E2,…, En have nonzero probability of occurrence and they form a partition of S. Let A be any event associated with S, then according to Bayes theorem,



**Note:**

The following terminologies are also used when the Bayes theorem is applied:

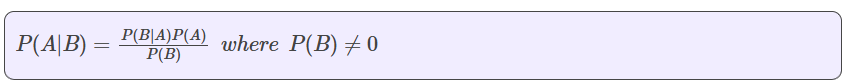
**Hypotheses:** The events E1, E2,… En is called the hypotheses

**Priori Probability:**The probability P(Ei) is considered as the priori probability of hypothesis Ei

**Posteriori Probability:** The probability P(Ei|A) is considered as the posteriori probability of hypothesis Ei  
Bayes’ theorem is also called the formula for the probability of “causes”. Since the Ei‘s are a partition of the sample space S, one and only one of the events Ei occurs (i.e. one of the events Ei must occur and the only one can occur). Hence, the above formula gives us the probability of a particular Ei (i.e. a “Cause”), given that the event A has occurred.

**Bayes Theorem Formula**

If A and B are two events, then the **formula for the Bayes theorem** is given by:



Where P(A|B) is the probability of condition when event A is occurring while event B has already occurred.  
Also, get the [Bayes Theorem Calculator](https://byjus.com/bayes-theorem-calculator/) here.

Bayes Theorem Derivation

Bayes Theorem can be derived for events and [random variables](https://byjus.com/maths/random-variable/) separately using the definition of conditional probability and density.

From the definition of conditional probability, Bayes theorem can be derived for events as given below:

P(A|B) = P(A ⋂ B)/ P(B), where P(B) ≠ 0

P(B|A) = P(B ⋂ A)/ P(A), where P(A) ≠ 0

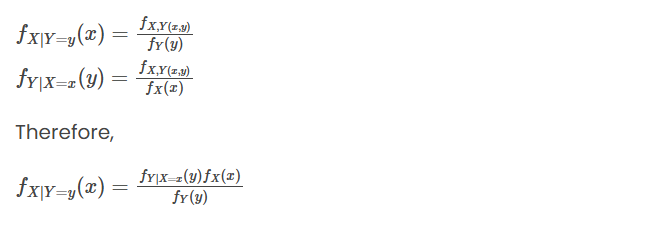
Here, the joint probability P(A ⋂ B) of both events A and B being true such that,

P(B ⋂ A) = P(A ⋂ B)

P(A ⋂ B) = P(A | B) P(B) = P(B | A) P(A)

P(A|B) = [P(B|A) P(A)]/ P(B), where P(B) ≠ 0

Similarly, from the definition of conditional density, Bayes theorem can be derived for two continuous random variables namely X and Y as given below:



**Examples and Solutions**

Some illustrations will improve the understanding of the concept.

**Example 1:**

A bag I contains 4 white and 6 black balls while another Bag II contains 4 white and 3 black balls. One ball is drawn at random from one of the bags, and it is found to be black. Find the probability that it was drawn from Bag I.

**Solution:**

Let E1 be the event of choosing bag I, E2 the event of choosing bag II, and A be the event of drawing a black ball.

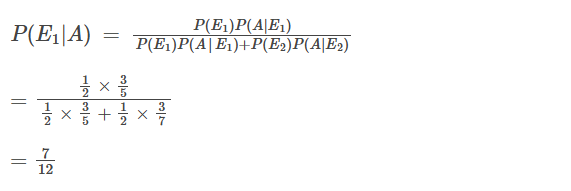
Then,

P(E1) = P(E2) = 12

Also, P(A|E1) = P(drawing a black ball from Bag I) = 6/10 = 3/5

P(A|E2) = P(drawing a black ball from Bag II) = 3/7

By using Bayes’ theorem, the probability of drawing a black ball from bag I out of two bags,



**Example 2:**

A man is known to speak the truth 2 out of 3 times. He throws a die and reports that the number obtained is a four. Find the probability that the number obtained is actually a four.

**Solution:**

Let A be the event that the man reports that number four is obtained.

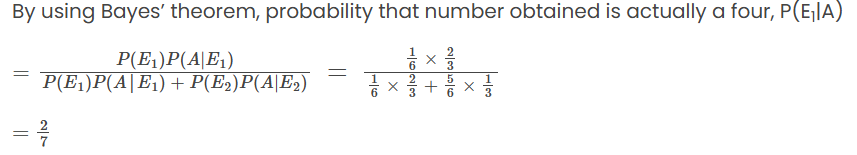
Let E1 be the event that four is obtained and E2 be its complementary event.

Then, P(E1) = Probability that four occurs = 1/6.

P(E2) = Probability that four does not occur = 1- P(E1) = 1 – (1/6) = 5/6.

Also, P(A|E1)= Probability that man reports four and it is actually a four = 2/3

P(A|E2) = Probability that man reports four and it is not a four = 1/3.



**Bayes Theorem Applications**

One of the many applications of Bayes’ theorem is Bayesian inference, a particular approach to statistical inference. Bayesian inference has found application in various activities, including medicine, science, philosophy, engineering, sports, law, etc. For example, we can use Bayes’ theorem to define the accuracy of medical test results by considering how likely any given person is to have a disease and the test’s overall accuracy. Bayes’ theorem relies on consolidating prior probability distributions to generate posterior probabilities. In Bayesian statistical inference, prior probability is the probability of an event before new data is collected.

**Question 1: What is the probability of a die showing a number 3 or number 5?**

Solution: Let,

P(3) is the probability of getting a number 3

P(5) is the probability of getting a number 5

P(3) = 1/6 and P(5) = 1/6

So,

P(3 or 5) = P(3) + P(5)

P(3 or 5) = (1/6) + (1/6) = 2/6

P(3 or 5) = 1/3

Therefore, the probability of a die showing 3 or 5 is 1/3.

**Question 2: Three coins are tossed at the same time. We say A as the event of receiving at least 2 heads. Likewise, B denotes the event of getting no heads and C is the event of getting heads on the second coin. Which of these is mutually exclusive?**

Solution:  Firstly, let us create a sample space for each event. For the event ‘A’ we have to get at least two head. Therefore, we have to include all the events that have two or more heads.

Or we can write:

A = {HHT, HTH, THH, HHH}.

This set A has 4 elements or events in it i.e. n(A) = 4

In the same way,  for event B, we can write the sample as:

B = {TTT} and n(B) = 1

Again using the same logic, we can write;

C = {THT, HHH, HHT, THH} and n(C) = 4

So B & C and A & B are mutually exclusive since they have nothing in their intersection.

**Question 3:** **The likelihood of the 3 teams a, b, c winning a football match are 1 / 3, 1 / 5 and 1 / 9 respectively. Find the probability that**

**a] out of the three teams, either team a or team b will win**

**b] either team a or team b or team c will win**

**c] none of the teams will win the match**

**d] neither team a nor team b will win the match**

**Answer:**

a) P (A or B will win) = 1/3 + 1/5 = 8/15

b) P (A or B or C will win) = 1/3 + 1/5 + 1/9 = 29/45

c) P (none will win) = 1 – P (A or B or C will win) = 1 – 29/45 = 16/45

d) P (neither A nor B will win) = 1 – P(either A or B will win)

= 1 – 8/15

= 7/15

**Question 4: If A and B are two independent events, then A and B’ is:**

Answer:  A ∩ B’ and A ∩ B are mutually exclusive events such that;

A = (A ∩ B’) ∪ (A ∩ B)

P(A) = P(A ∩ B’) + P(A ∩ B)

P(A ∩ B’) = P(A) – P(A ∩ B)

= P(A) – P(A).P(B)   (Since A and B are independent)

= P(A ∩ B’)

=> P(A) (1 – P(B)) = P(A) P(B’)

Thus, A and B’ are also independent.

**Question 5:** **If P (A) = 2 / 3, P (B) = 1 / 2 and P (A ∪ B) = 5 / 6 then events A and B are:**

**Answer:**

P (A ∪ B) = P (A) + P (B) − P (A ∩ B)

5 / 6 = (2 / 3) + (1 / 2) − P (A ∩ B)

⇒ P (A ∩ B) = 0

The events A and B are mutually exclusive.

**Question 6: A card is drawn at random from a well-shuffled deck of 52 cards. Find the probability that the card drawn is a king or an ace.**

**Answer:**

As per the definition of mutually exclusive events, selecting an ace and selecting a king from a well-shuffled deck of 52 cards are termed mutually exclusive events.

Assume X to be the event of drawing a king and Y to be the event of drawing an ace.

In a standard deck of 52 cards, there exists 4 kings and 4 aces.

P (an event) = count of favourable outcomes / total count of outcomes

P (selecting a king from a standard deck of 52 cards) = P (X) = 4 / 52 = 1 / 13

P (selecting an ace from a standard deck of 52 cards) = P (Y) = 4 / 52 = 1 / 13

To compute P (king or ace).

By the formula of addition theorem for mutually exclusive events,

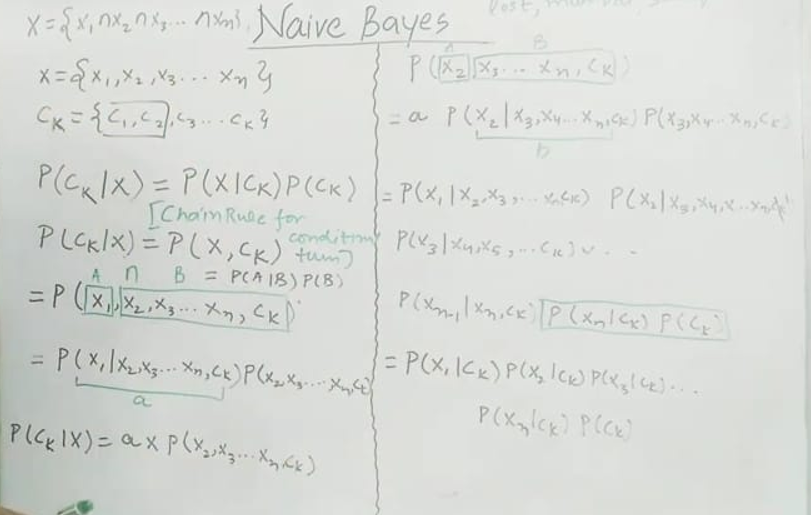
P (X U Y) = P (X) + P (Y)

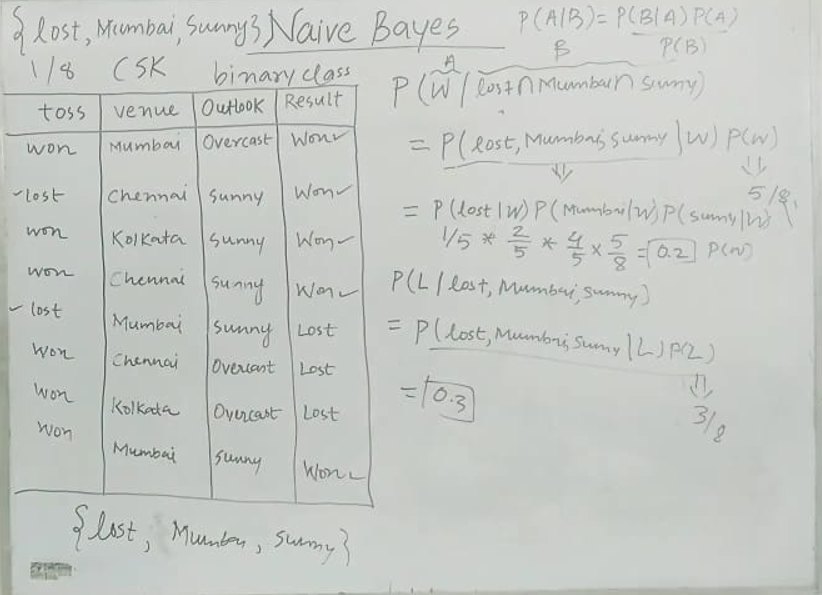
P (X U Y) = (1 / 13) + (1 / 13)

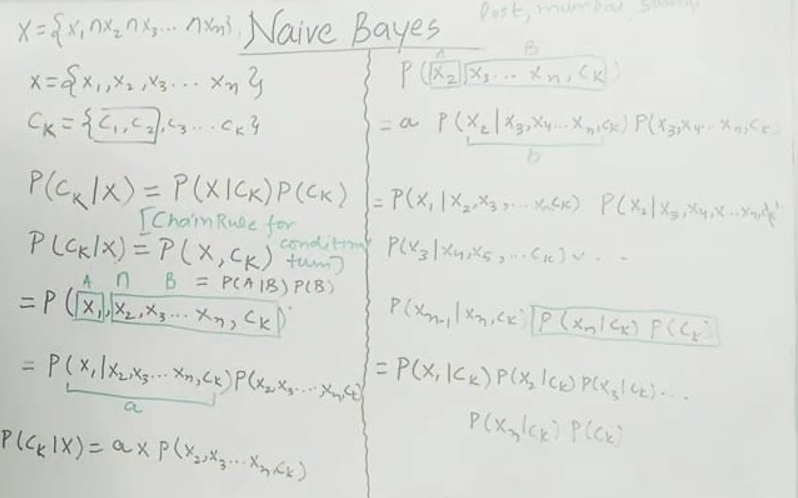
= (1 + 1) / 13

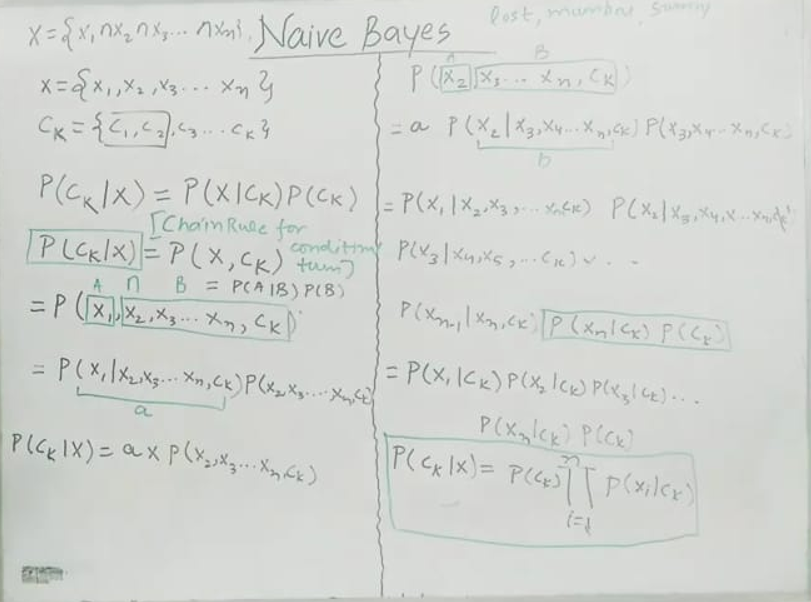
= 2 / 13

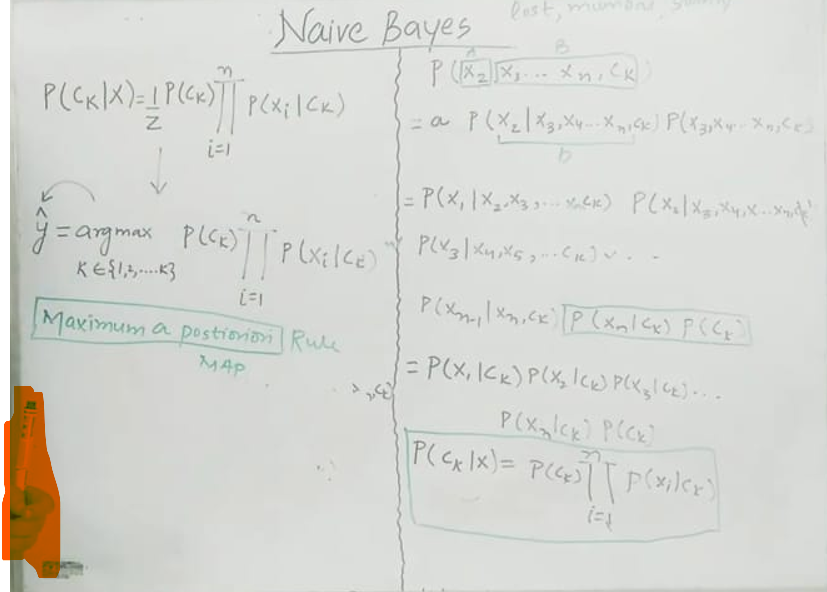
The probability of selecting a king or an ace from a well-shuffled deck of 52 cards = 2 / 13.







****



<https://www.analyticsvidhya.com/blog/2023/01/naive-bayes-algorithms-a-complete-guide-for-beginners/>

**What is Naive Bayes Algorithm?**

Now is the best time to understand the naive Bayes algorithm, as the core fundamentals are clear. In real-time, there can be many events and many conditions that can happen simultaneously with events. So, in this case, we expand the bayesian theorem to solve this type of issue. If the features are independent, we can quickly **extend**the theorem and calculate the probability of the same.

The same bayesian theorem formula can be used here for multiple events and conditions, and one can easily calculate the probability with the help of the same.

The algorithm is one of the most useful algorithms in machine learning which helps in several classification problems, sentiment analysis, face recognition, etc.

“**Naive Bayes** is a **probabilistic classifier** based on **Bayes’ Theorem** with a strong (naive) assumption:

All features are **independent** of each other given the class label.

Despite this simplification, it performs surprisingly well in many real-world problems like **text classification**, **spam detection**, and **sentiment analysis**.

”

**How Naive Bayes Works?**

Let’s explain by taking a example –

**Problem Setup: Gender Classification**

We want to classify whether a person is **Male** or **Female** based on two features:

1. **Height** (in cm)
2. **Voice Pitch** (in Hz)

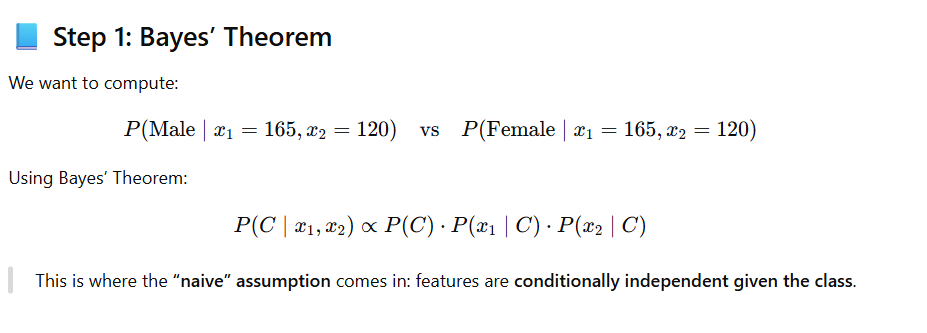
**Dataset (simplified)**

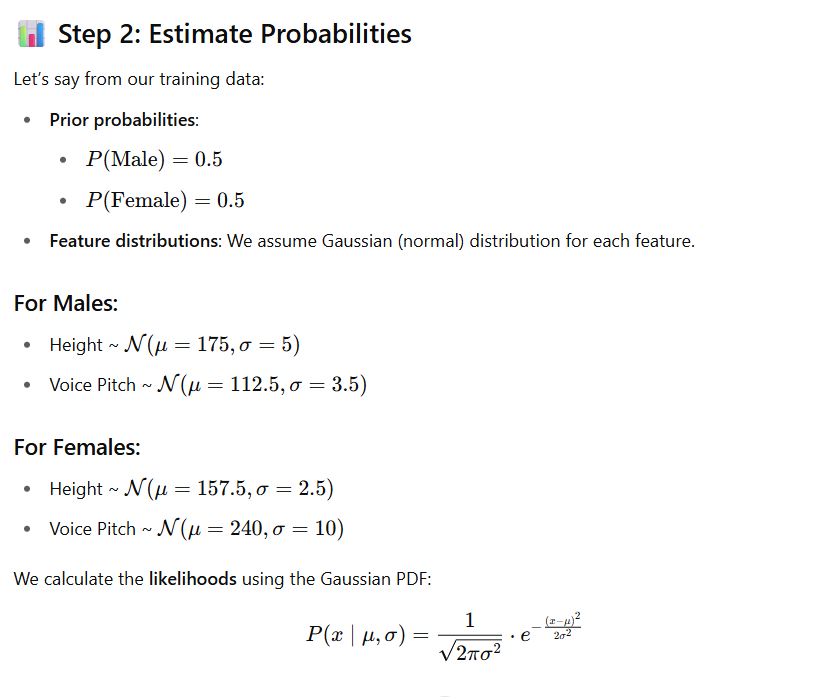
| **Height (x₁)** | **Voice Pitch (x₂)** | **Gender (Target: C)** |
| --- | --- | --- |
| 180 | 110 | Male |
| 170 | 115 | Male |
| 160 | 230 | Female |
| 155 | 250 | Female |

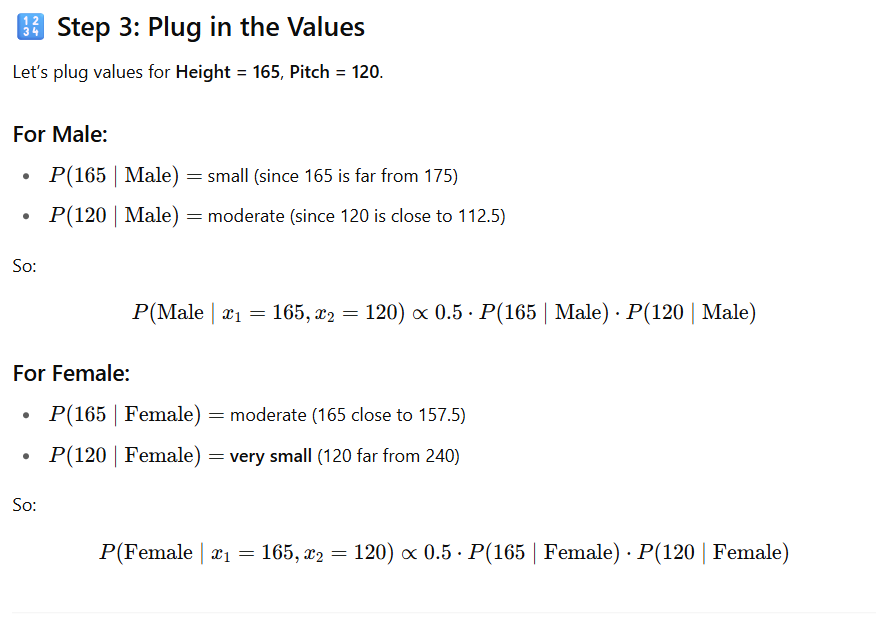
Now suppose we get a **new person with**:

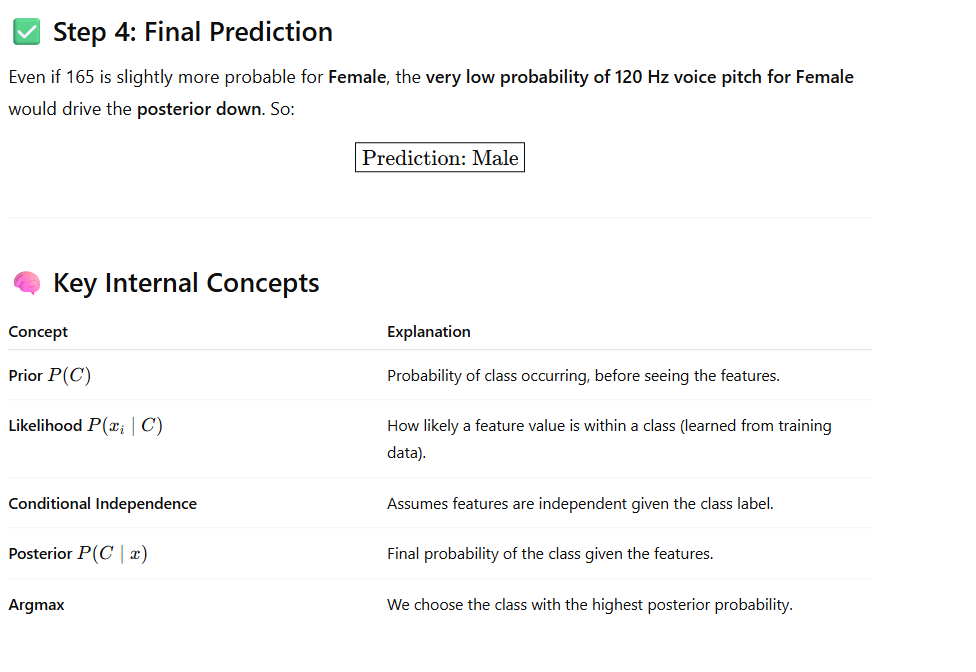
* Height = 165 cm
* Voice Pitch = 120 Hz

We want to predict: **Male or Female?**

‘



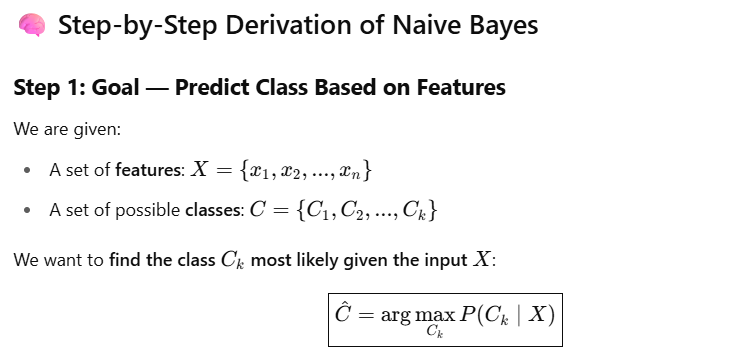


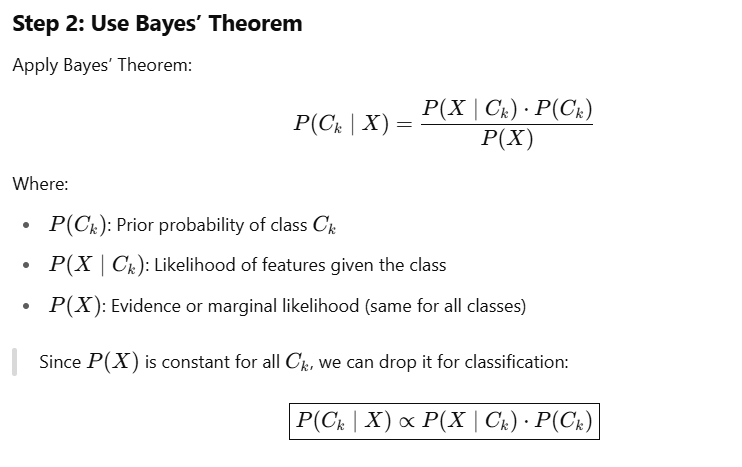


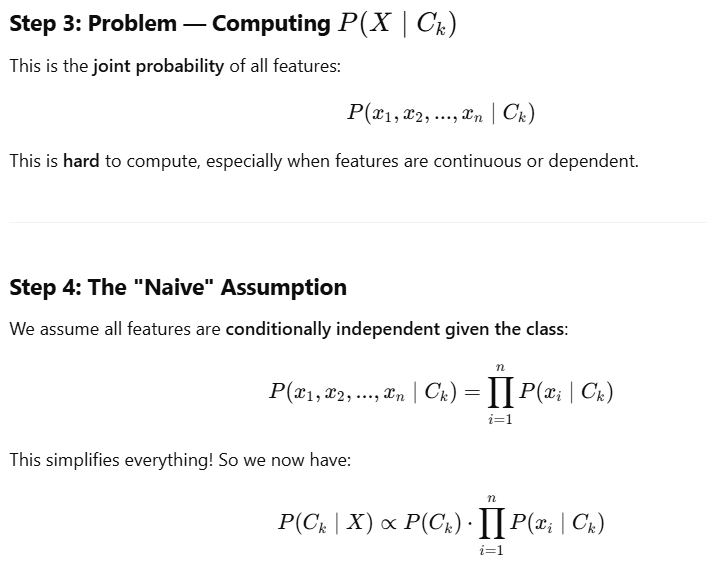
**Summary**

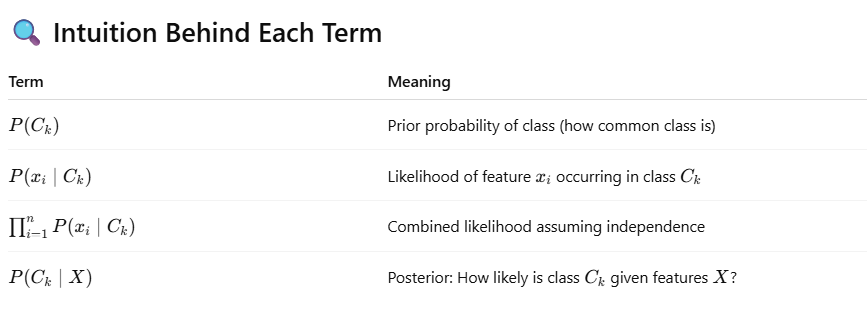
* Naive Bayes **learns class-conditional distributions** for each feature.
* It assumes all features are **independent given the class**.
* It’s fast, effective, and especially good for **text classification** or **well-separated distributions**.
* In our example, voice pitch played a big role due to very different distributions between genders.

**Let’s dive deeply into the intuition and derivation of the Naive Bayes classifier :-**



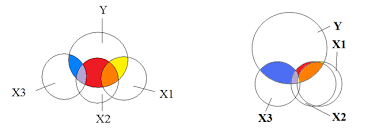






**What is Multicollinearity?**

Multicollinearity in machine learning is a term that deals with the linearity of the features of data feed. In simple words, the dataset having **correlations between its independent features** is called multilinear.



To understand the concept better, let us take an example.

Suppose we have a dataset with three columns, age, marks, and passed. Here the age is the age of the students, marks are the number obtained by students in exams, and the past is a categorical column that indicates whether a student **passed or not**.

Now here, the age and marks are the training columns means these columns should be fed to the algorithm, and the passed column should be the target column that a machine learning algorithm will predict. Now in some cases, the age and the marks columns are correlated somehow, and they are not independent anymore. It is called that the data has **Multicollinearity**in its features.

The professor checking the answer sheets can be biased toward students having less age and marks them with good numbers. Both columns are now correlated, and Multicollinearity is present in this dataset.

**How to Check Multicollinearity?**

One of the basic assumptions of the naive Bayes algorithm is related to the Multicollinearit; it is required to check whether the data has **Multicollinearity**.

To check the some, we can use the following code:

import pandas as pd

df = pd.read\_csv("data.csv")

df.corr()Copy Code

The following code results in the **Pearson Correlation** between the independent and dependent columns; we can check the relation between all the independent columns with another independent column to check for Multicollinearity.

**Why is it Naive?**

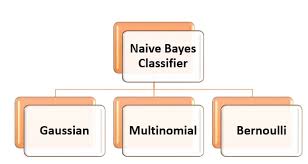
Now a question might appear in your mind: Why is the algorithm called naive?

The main reason behind the name of the Naive Bayes Classifier is its assumption that ut assume while working on particular datasets and the **Multicollinearity**.

Here Naive Bayes Classifier assumes that the dataset provided to the algorithm is independent and the independent features are separate and not dependent on some other factors, which is why the **Naive Bayes** algorithm is called **Naive**.

Types of Naive Bayes

There are mainly a total of three types of naive byes algorithms. Different types of naive Bayes are used for different use cases. Let us try to understand them one by one.



**1. Bernoulli Naive Bayes**

This Naive Bayes Classifier is used when there is a **boolean** type of dependent or target variable present in the dataset. For example, a dataset has target column categories as Yes and No.

This type of Naive is mainly used in a binary categorical tagete column where the problem statement is to predict only **Yes or No**. For Example, sentiment analysis with Positive and Negative Categories, A specific ord id present in the text or not, etc.

**Code Example:**

from sklearn.datasets import make\_classification

from sklearn.naive\_bayes import BernoulliNB

from sklearn.model\_selection import train\_test\_split

nb\_samples = 100

X, Y = make\_classification(n\_samples=nb\_samples, n\_features=2, n\_informative=2, n\_redundant=0)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.25)

bnb = BernoulliNB(binarize=0.0)

bnb.fit(X\_train, Y\_train)

bnb.score(X\_test, Y\_test)Copy Code

**2. Multinomial Naive Bayes**

This type of naive Bayes is used where the data is multinomial distributed. This type of naive Bayes is mainly used when there is a **text classification** problem.

For Example, if you want to predict whether a text belongs to which tag, education, politics, e-tech, or some other tag, you can use the multinomial Naive Bayes Classifier to classify the same.

This naive base **outperforms**text classification problems and is used the most out of all the other Naive Bayes Classifier.

**Code Example:**

from sklearn.feature\_extraction import DictVectorizer

from sklearn.naive\_bayes import MultinomialNB

data = [

{'parth1': 100, 'parth2': 50, 'parth3': 25, 'parth4': 100, 'parth5': 20},

{'parth1': 5, 'parth2': 5, 'parth3': 0, 'parth4': 10, 'parth5': 500, 'parth6': 1}

]

dv = DictVectorizer(sparse=False)

X = dv.fit\_transform(data)

Y = np.array([1, 0])

mnb = MultinomialNB()

mnb.fit(X, Y)

test\_data = data = [

{'parth1': 80, 'parth2': 20, 'parth3': 15, 'parth4': 70, 'parth5': 10, 'parth6':

1},

]

{'parth1': 10, 'parth2': 5, 'parth3': 1, 'parth4': 8, 'parth5': 300, 'parth6': 0}

mnb.predict(dv.fit\_transform(test\_data))Copy Code

**3. Gaussian Naive Bayes**

This type of naive is used when the predictor variables have **continuous values** instead of discrete ones. Here it is assumed that the distribution of the data is **Gaussian distribution**.

**Code Example:**

from sklearn.datasets import make\_classification

from sklearn.naive\_bayes import GaussianNB

from sklearn.model\_selection import train\_test\_split

nb\_samples = 100

X, Y = make\_classification(n\_samples=nb\_samples, n\_features=2, n\_informative=2, n\_redundant=0)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.25)

gnb = GaussianNB(binarize=0.0)

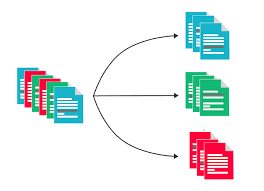
gnb.fit(X\_train, Y\_train)

bngnb.score(X\_test, Y\_test)Copy Code

Applications of Naive Bayes

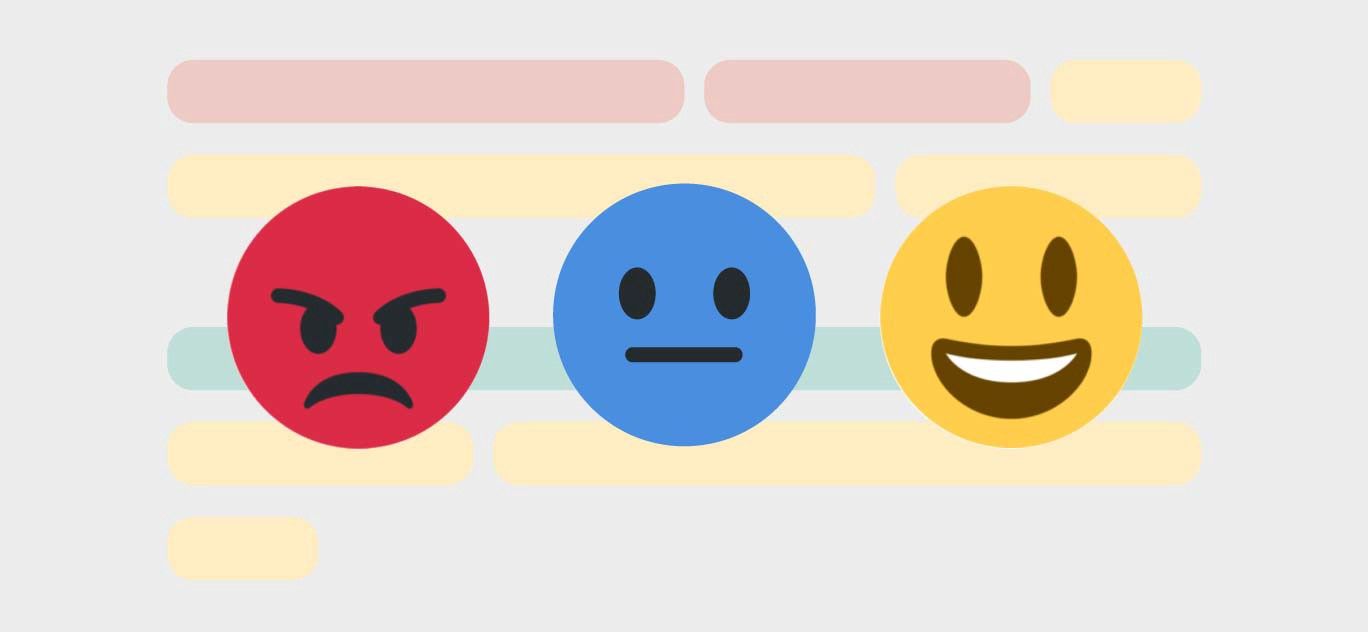
**1. Text Classification**

The naive Bayes algorithms are known to perform best on **text classification** problems. The algorithm is mainly used when there is a problem statement related to the text and its classification. Several naive Bayes algorithms are tried and tuned according to the problem statement and used for a better accurate model. For Example: classifying the tags from the text etc.



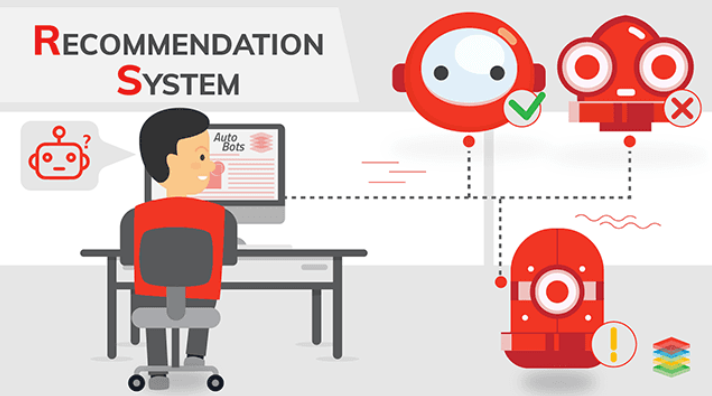
**2. Sentiment Analysis**

Algorithms like Bernoulli naive are used most for these [**sentiment analysis**](https://www.analyticsvidhya.com/blog/2022/10/sentiment-analysis-using-vader/) problems. This algorithm is known to outperform on binary classification problems and is hence used most for such cases.



**3. Recommendation Systems**

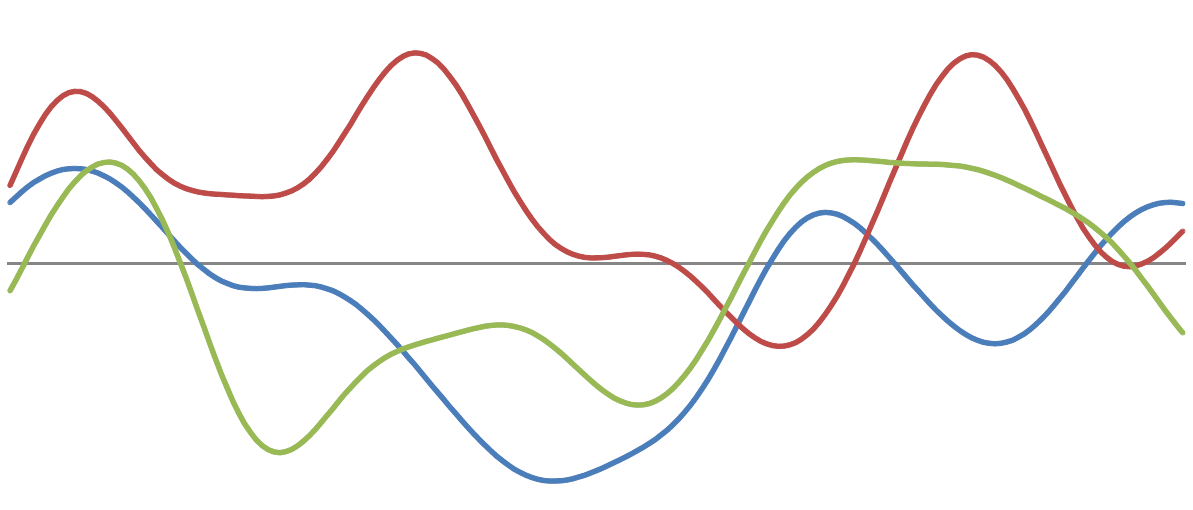
There are a total of two recommendation systems, content-based and collaborative filtering. The naive Bayes with **collaborative filtering-based models** is known for their best accuracy on recommendation problems. The naive Bayes algorithms help achieve better accuracies for recommending features to the users based on their interests and related to other users’ interests.



Source: https://www.xenonstack.com/hubfs/xenonstack-deep-learning-based-recommendation-system.png

**4. Real-Time Predictions**

The Naive Bayes algorithms are **eager learning** algorithms that try to learn from the training data and assume some of the parameters. Now, whenever the test data is provided for prediction to the algorithm, the algorithm calculates the results according to its knowledge gained from the training and offers faster and more accurate results. Hence it could be used for **real-time predictions**.



**Advantages and Disadvantages of Naive Bayes**

**Advantages**

**1. Faster Algorithms:**

The Naive Bayes algorithm is a parametric algorithm that tries to assume certain things while training and using the knowledge for prediction. Hence it takes significantly less time for prophecy and is a **faster algorithm**.

**2. Less Training Data:**

The naive Bayes algorithm assumes the **independent features to be independent** of each other, and if it exists, then the naive Bayes needs less data for training and performs better.

**3. Performance:**

The Naive Bayes algorithm achieves faster and more accurate performance with less data, and its handling of categorical text data surpasses that of other algorithms, making comparisons inequitable.

**Disadvantages**

**1. Independent Features:**

In a real-time dataset, obtaining independent features that are entirely independent of each other is almost impossible. There are typically two to three features that correlate with each other, thus not fully satisfying the assumption at all times.

**2. Zero Frequency Error:**

The zero frequency error in naive Bayes is one of the most critical CONs of the Naive Bayes algorithm. According to this error, if a category is absent in both the training data and the test data, then the Naive Bayes algorithm will assign it zero probability, resulting in what is known as the Zero Frequency error in Naive Bayes.

To address this kind of issue, we can use Laplace smoothing techniques.

When to Use Naive Bayes?

Well, the Naive Bayes algorithm is the best-performing and faster algorithm compared to other algorithms. However, still, there are cases where it cannot perform well, and some different algorithms should be used to handle such cases.

The Naive Bayes algorithm can be used if there is no multicollinearity in the independent features and if the features’ probabilities provide some valuable information to the algorithms.

This algorithm should also be preferred for text classification problems. One should avoid using the Naive Bayes algorithm when the data is entirely numeric and multicollinearity is present in the dataset.

If it is necessary to use the Naive Bayes algorithm, then one can use the following steps to improve the performance of Naive Bayes algorithms.

How to Improve Naive Bayes?

**1. Remove Correlated Features:**

Naive Bayes algorithms perform well on datasets with no correlations in independent features. **removing the correlated features** may improve the performance of the algorithm

**2. Feature Engineering:**

Try to apply feature engineering to the dataset and its features, **combine some** of the elements, and **extract some parts** of them out of existing ones. This may help the Naive Bayes algorithm learn the data quickly and results in an accurate model.

**3. Use Some Domain Knowledge:**

Oe should always try to apply some **domain knowledge** to the dataset and its features and take steps according to it. It may help the algorithm to make decisions faster and achieve higher accuracies.

**4. Probabilistic Features:**

The Naive Bayes algorithm works on the concept of probabilities, so try to improve the features that give **more weightage to the algorithms and their probabilities**, try to implement those, and run the roses in a loop to know which features are best for the algorithm.

**5. Laplace Transform:**

In some cases, the category may be present in the test dataset and was not present while training and the model will assign it with zero probability. Here we should handle this issue by **Laplace transform**.

**6. Feature Transformation:**

It is always better to have normal distributions in the datasets and try to apply **box-cox and yeo-johnson** feature transformation techniques to achieve the normal distributions in the dataset.

**How Naive Bayes handles different data characteristics:**

**1. Non-linear relationships**

* **Naive Bayes does *not* model complex non-linear relationships** between features.
* Why? Because it assumes that all features are **conditionally independent** given the class. So, it doesn’t "learn" interactions or non-linear boundaries.
* **Workaround**: You can still apply it by **transforming features** or using **non-linear basis functions** (like polynomial terms) beforehand.

**2. Polynomial data**

* Naive Bayes doesn’t **automatically handle** polynomial relationships.
* But if you **add polynomial features** manually (e.g., using PolynomialFeatures from sklearn.preprocessing), you can still apply Naive Bayes.
* Example: Add features like x1^2, x2^2, x1\*x2 → Then run Naive Bayes.

**3. Outliers**

* **Sensitive** to outliers, especially in **Gaussian Naive Bayes**.
* Outliers can skew the estimated **mean** and **variance**, making probability calculations inaccurate.
* **Mitigation strategies**:
  + Use **robust scaling** (like RobustScaler).
  + Use **discretization** (convert continuous to bins).
  + Switch to **Bernoulli** or **Multinomial** NB if suitable.

**4. Non-homogeneous feature scales**

* Naive Bayes **doesn’t require feature scaling**, unlike distance-based algorithms (like KNN, SVM).
* But in **Gaussian NB**, feature scale **affects the variance** term → could make certain features dominate.
* Solution: **Standardize features** to zero mean, unit variance using StandardScaler.

**5. Imbalanced Data**

* Naive Bayes works **better than many algorithms on imbalanced datasets** because it:
  + Uses **prior probabilities** P(Ck)P(C\_k)P(Ck​), so it can naturally account for class imbalance.
  + You can adjust prior probabilities manually using class\_prior in GaussianNB.

**❌ Can Naive Bayes be used for Regression?**

🟥 **No, Naive Bayes is not designed for regression problems.**

**Why?**

* Naive Bayes is **probabilistic classification**: it estimates the **probability of a class given features**.
* In regression, we want to predict **continuous output**, which NB doesn’t do.

**🧠 Alternatives for regression using similar probabilistic reasoning:**

| **Technique** | **Use Case** |
| --- | --- |
| **Bayesian Regression** | Probabilistic regression, gives prediction + uncertainty |
| **Gaussian Processes** | Kernel-based regression with probabilistic output |
| **Quantile Regression** | Estimates different percentiles, robust to outliers |

**Summary Table**

| **Challenge** | **Naive Bayes Handling** | **Notes** |
| --- | --- | --- |
| Non-linear data | ❌ Poor by default | Can work if you engineer non-linear features |
| Polynomial relationships | ❌ Needs manual feature transformation | Use PolynomialFeatures |
| Outliers | ⚠️ Sensitive | Use robust preprocessing or discretization |
| Non-homogeneous scales | ⚠️ Can affect Gaussian NB | Use StandardScaler or MinMaxScaler |
| Imbalanced data | ✅ Handles via class priors | Use class\_prior argument |
| Regression tasks | ❌ Not supported | Use Bayesian Regression or other models |

Code examples:--

# Fix for MultinomialNB: Ensure we use the same scaler and shift consistently to keep features positive

scaler\_mn = StandardScaler()

X\_train\_mn = scaler\_mn.fit\_transform(X\_train) + abs(scaler\_mn.transform(X\_train).min()) + 1 # ensure positive

X\_test\_mn = scaler\_mn.transform(X\_test) + abs(scaler\_mn.transform(X\_train).min()) + 1

clf\_mnb = MultinomialNB()

clf\_mnb.fit(X\_train\_mn, y\_train)

y\_pred\_mnb = clf\_mnb.predict(X\_test\_mn)

results['MultinomialNB'] = classification\_report(y\_test, y\_pred\_mnb, output\_dict=True)

results.keys() # Return the models evaluated again after fix

# 1. GaussianNB with StandardScaler (for continuous, normally distributed features)

from sklearn.naive\_bayes import GaussianNB

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report

# Load and split data

X, y = load\_breast\_cancer(return\_X\_y=True)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

# Standardize and apply GaussianNB

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

gnb = GaussianNB()

gnb.fit(X\_train\_scaled, y\_train)

y\_pred\_gnb = gnb.predict(X\_test\_scaled)

report\_gnb = classification\_report(y\_test, y\_pred\_gnb)

# 2. MultinomialNB (for positive count features, simulate with digits dataset)

from sklearn.naive\_bayes import MultinomialNB

from sklearn.datasets import load\_digits

digits = load\_digits()

X\_mn, y\_mn = digits.data, digits.target

X\_train\_mn, X\_test\_mn, y\_train\_mn, y\_test\_mn = train\_test\_split(X\_mn, y\_mn, test\_size=0.3, random\_state=0)

# Shift data to positive values if needed

X\_train\_mn = X\_train\_mn + 1

X\_test\_mn = X\_test\_mn + 1

mnb = MultinomialNB()

mnb.fit(X\_train\_mn, y\_train\_mn)

y\_pred\_mnb = mnb.predict(X\_test\_mn)

report\_mnb = classification\_report(y\_test\_mn, y\_pred\_mnb)

# 3. BernoulliNB (binary features)

from sklearn.naive\_bayes import BernoulliNB

# Binarize breast cancer dataset

X\_bin = (X > X.mean(axis=0)).astype(int)

X\_train\_bin, X\_test\_bin, y\_train\_bin, y\_test\_bin = train\_test\_split(X\_bin, y, test\_size=0.3, random\_state=0)

bnb = BernoulliNB()

bnb.fit(X\_train\_bin, y\_train\_bin)

y\_pred\_bnb = bnb.predict(X\_test\_bin)

report\_bnb = classification\_report(y\_test\_bin, y\_pred\_bnb)

(report\_gnb, report\_mnb, report\_bnb)

import numpy as np

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

from sklearn.datasets import load\_breast\_cancer, load\_digits

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.naive\_bayes import GaussianNB, MultinomialNB, BernoulliNB

def plot\_decision\_boundary(model, X, y, title):

h = 0.02 # step size in the mesh

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),

np.arange(y\_min, y\_max, h))

Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.4, cmap=plt.cm.RdYlBu)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdYlBu, edgecolor='k')

plt.title(title)

plt.xlabel("PCA Component 1")

plt.ylabel("PCA Component 2")

plt.show()

# Load data

X, y = load\_breast\_cancer(return\_X\_y=True)

X\_digits, y\_digits = load\_digits(return\_X\_y=True)

# Reduce to 2D

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(StandardScaler().fit\_transform(X))

X\_digits\_pca = pca.fit\_transform(X\_digits)

# 1. GaussianNB

X\_train\_g, X\_test\_g, y\_train\_g, y\_test\_g = train\_test\_split(X\_pca, y, test\_size=0.3, random\_state=0)

clf\_gnb = GaussianNB()

clf\_gnb.fit(X\_train\_g, y\_train\_g)

plot\_decision\_boundary(clf\_gnb, X\_train\_g, y\_train\_g, "GaussianNB Decision Boundary")

# 2. MultinomialNB (with digits)

X\_digits\_mn = X\_digits + 1 # make it all positive

X\_train\_mn, X\_test\_mn, y\_train\_mn, y\_test\_mn = train\_test\_split(X\_digits\_mn, y\_digits, test\_size=0.3, random\_state=0)

X\_digits\_pca\_train = pca.fit\_transform(X\_train\_mn)

clf\_mnb = MultinomialNB()

clf\_mnb.fit(X\_digits\_pca\_train, y\_train\_mn)

plot\_decision\_boundary(clf\_mnb, X\_digits\_pca\_train, y\_train\_mn, "MultinomialNB Decision Boundary")

# 3. BernoulliNB

X\_bin = (X > np.mean(X, axis=0)).astype(int)

X\_pca\_bin = pca.fit\_transform(X\_bin)

X\_train\_b, X\_test\_b, y\_train\_b, y\_test\_b = train\_test\_split(X\_pca\_bin, y, test\_size=0.3, random\_state=0)

clf\_bnb = BernoulliNB()

clf\_bnb.fit(X\_train\_b, y\_train\_b)

plot\_decision\_boundary(clf\_bnb, X\_train\_b, y\_train\_b, "BernoulliNB Decision Boundary")