1. What is prior probability? Give an example.

Prior probability, in Bayesian statistics, is the probability of an event before new data is collected. This is the best rational assessment of the probability of an outcome based on the current knowledge before an experiment is performed.

**Understanding Prior Probability**

The prior probability of an event will be revised as new data or information becomes available, to produce a more accurate measure of a potential outcome. That revised probability becomes the posterior probability and is calculated using [Bayes' theorem](https://www.investopedia.com/terms/b/bayes-theorem.asp). In statistical terms, the posterior probability is the probability of event A occurring given that event B has occurred.

Example

For example, three acres of land have the labels A, B, and C. One acre has reserves of oil below its surface, while the other two do not. The prior probability of oil being found on acre C is one third, or 0.333. But if a drilling test is conducted on acre B, and the results indicate that no oil is present at the location, then the posterior probability of oil being found on acres A and C become 0.5, as each acre has one out of two chances.

1. What is posterior probability? Give an example.

Posterior probability is a revised probability that takes into account new available information. For example, let there be two urns, urn A having 5 black balls and 10 red balls and urn B having 10 black balls and 5 red balls. Now if an urn is selected at random, the probability that urn A is chosen is 0.5. This is the a *priori probability*. If we are given an additional piece of information that a ball was drawn at random from the selected urn, and that ball was black, what is the probability that the chosen urn is urn A? Posterior probability takes into account this additional information and revises the probability downward from 0.5 to 0.333 according to Bayes´ theorem, because a black ball is more probable from urn B than urn A.

3. What is likelihood probability? Give an example.

We can define likelihood as a quantitative estimation or measure that states the fitness of a model or hypothesis in observed data. It can also be interpreted as the chance of finding the desired result or data collection in a specific parameter set. Playing a fundamental role in [statistical inference](https://www.analyticsvidhya.com/blog/2022/02/statistical-inference-using-python/), the ultimate aim of likelihood is to conclude about the data’s characteristics. The role in achieving the same is seen through parameter estimation, which utilizes [Maximum Likelihood Estimation or MLE](https://www.analyticsvidhya.com/blog/2021/09/maximum-likelihood-estimation-a-comprehensive-guide/) to find parameter estimates.

Hypothesis testing uses likelihood ratios to assess the null hypothesis. Similarly, likelihood contributes by comparing models for model selection and checking. Researchers commonly utilize [Bayesian Information Criterion](https://www.analyticsvidhya.com/blog/2018/09/multivariate-time-series-guide-forecasting-modeling-python-codes/) (BIC) and Akaike Information Criterion (AIC) as measures in model selection. Likelihood-based methods play a significant role in constructing confidence intervals to estimate the parameters.

Example :- Let’s consider a fair coin toss. The likelihood of obtaining a ‘heads’ outcome in a single toss, assuming the coin is fair, is 0.5 since there are two equally likely possibilities (heads or tails).

4. What is Naïve Bayes classifier? Why is it named so?

# Naive Bayes Classifiers

**Last Updated :**01 Mar, 2024

A Naive Bayes classifiers, a family of algorithms based on Bayes’ Theorem. Despite the “naive” assumption of feature independence, these classifiers are widely utilized for their simplicity and efficiency in machine learning. The article delves into theory, implementation, and applications, shedding light on their practical utility despite oversimplified assumptions.

## What is Naive Bayes Classifiers?

Naive Bayes classifiers are a collection of classification algorithms based on Bayes’ Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. To start with, let us consider a dataset.

One of the most simple and effective classification algorithms, the Naïve Bayes classifier aids in the rapid development of machine learning models with rapid prediction capabilities.

Naïve Bayes algorithm is used for classification problems. It is highly used in text classification. In text classification tasks, data contains high dimension (as each word represent one feature in the data). It is used in spam filtering, sentiment detection, rating classification etc. The advantage of using naïve Bayes is its speed. It is fast and making prediction is easy with high dimension of data.

This model predicts the probability of an instance belongs to a class with a given set of feature value. It is a probabilistic classifier. It is because it assumes that one feature in the model is independent of existence of another feature. In other words, each feature contributes to the predictions with no relation between each other. In real world, this condition satisfies rarely. It uses Bayes theorem in the algorithm for training and prediction

## Why it is Called Naive Bayes?

The “Naive” part of the name indicates the simplifying assumption made by the Naïve Bayes classifier. The classifier assumes that the features used to describe an observation are conditionally independent, given the class label. The “Bayes” part of the name refers to Reverend Thomas Bayes, an 18th-century statistician and theologian who formulated Bayes’ theorem.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit(“Yes”) or unfit(“No”) for playing golf.Here is a tabular representation of our dataset.

|  | **Outlook** | **Temperature** | **Humidity** | **Windy** | **Play Golf** |
| --- | --- | --- | --- | --- | --- |
| 0 | Rainy | Hot | High | False | No |
| 1 | Rainy | Hot | High | True | No |
| 2 | Overcast | Hot | High | False | Yes |
| 3 | Sunny | Mild | High | False | Yes |
| 4 | Sunny | Cool | Normal | False | Yes |
| 5 | Sunny | Cool | Normal | True | No |
| 6 | Overcast | Cool | Normal | True | Yes |
| 7 | Rainy | Mild | High | False | No |
| 8 | Rainy | Cool | Normal | False | Yes |
| 9 | Sunny | Mild | Normal | False | Yes |
| 10 | Rainy | Mild | Normal | True | Yes |
| 11 | Overcast | Mild | High | True | Yes |
| 12 | Overcast | Hot | Normal | False | Yes |
| 13 | Sunny | Mild | High | True | No |

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

* Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are ‘Outlook’, ‘Temperature’, ‘Humidity’ and ‘Windy’.
* Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is ‘Play golf’.

## Assumption of Naive Bayes

The fundamental Naive Bayes assumption is that each feature makes an:

* **Feature independence:** The features of the data are conditionally independent of each other, given the class label.
* **Continuous features are normally distributed:** If a feature is continuous, then it is assumed to be normally distributed within each class.
* **Discrete features have multinomial distributions:** If a feature is discrete, then it is assumed to have a multinomial distribution within each class.
* **Features are equally important:** All features are assumed to contribute equally to the prediction of the class label.
* **No missing data:** The data should not contain any missing values.

With relation to our dataset, this concept can be understood as:

* We assume that no pair of features are dependent. For example, the temperature being ‘Hot’ has nothing to do with the humidity or the outlook being ‘Rainy’ has no effect on the winds. Hence, the features are assumed to be **independent**.
* Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can’t predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

*The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.Now, before moving to the formula for Naive Bayes, it is important to know about Bayes’ theorem.*

## ****Bayes’ Theorem****

Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes’ theorem is stated mathematically as the following equation:

𝑃(𝐴∣𝐵)=𝑃(𝐵∣𝐴)𝑃(𝐴)𝑃(𝐵)*P*(*A*∣*B*)=*P*(*B*)*P*(*B*∣*A*)*P*(*A*)​

where A and B are events and P(B) ≠ 0

* Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
* P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
* P(B) is Marginal Probability: Probability of Evidence.
* P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.
* P(B|A) is Likelihood probability i.e the likelihood that a hypothesis will come true based on the evidence.

Now, with regards to our dataset, we can apply Bayes’ theorem in following way:

𝑃(𝑦∣𝑋)=𝑃(𝑋∣𝑦)𝑃(𝑦)𝑃(𝑋)*P*(*y*∣*X*)=*P*(*X*)*P*(*X*∣*y*)*P*(*y*)​

where, y is class variable and X is a dependent feature vector (of size n) where:

𝑋=(𝑥1,𝑥2,𝑥3,…..,𝑥𝑛)*X*=(*x*1​,*x*2​,*x*3​,…..,*xn*​)

Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

X = (Rainy, Hot, High, False)  
y = No

So basically, (𝑦∣𝑋)*P*(*y*∣*X*)here means, the probability of “Not playing golf” given that the weather conditions are “Rainy outlook”, “Temperature is hot”, “high humidity” and “no wind”.

With relation to our dataset, this concept can be understood as:

* We assume that no pair of features are dependent. For example, the temperature being ‘Hot’ has nothing to do with the humidity or the outlook being ‘Rainy’ has no effect on the winds. Hence, the features are assumed to be **independent**.
* Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can’t predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

Now, its time to put a naive assumption to the Bayes’ theorem, which is, **independence** among the features. So now, we split **evidence** into the independent parts.

Now, if any two events A and B are independent, then,

P(A,B) = P(A)P(B)

Hence, we reach to the result:

𝑃(𝑦∣𝑥1,…,𝑥𝑛)=𝑃(𝑥1∣𝑦)𝑃(𝑥2∣𝑦)…𝑃(𝑥𝑛∣𝑦)𝑃(𝑦)𝑃(𝑥1)𝑃(𝑥2)…𝑃(𝑥𝑛)*P*(*y*∣*x*1​,…,*xn*​)=*P*(*x*1​)*P*(*x*2​)…*P*(*xn*​)*P*(*x*1​∣*y*)*P*(*x*2​∣*y*)…*P*(*xn*​∣*y*)*P*(*y*)​

which can be expressed as:

𝑃(𝑦∣𝑥1,…,𝑥𝑛)=𝑃(𝑦)∏𝑖=1𝑛𝑃(𝑥𝑖∣𝑦)𝑃(𝑥1)𝑃(𝑥2)…𝑃(𝑥𝑛)*P*(*y*∣*x*1​,…,*xn*​)=*P*(*x*1​)*P*(*x*2​)…*P*(*xn*​)*P*(*y*)∏*i*=1*n*​*P*(*xi*​∣*y*)​

Now, as the denominator remains constant for a given input, we can remove that term:

𝑃(𝑦∣𝑥1,…,𝑥𝑛)∝𝑃(𝑦)∏𝑖=1𝑛𝑃(𝑥𝑖∣𝑦)*P*(*y*∣*x*1​,…,*xn*​)∝*P*(*y*)∏*i*=1*n*​*P*(*xi*​∣*y*)

Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable y and pick up the output with maximum probability. This can be expressed mathematically as:

𝑦=𝑎𝑟𝑔𝑚𝑎𝑥𝑦(𝑦)∏𝑖=1𝑛𝑃(𝑥𝑖∣𝑦)*y*=*argmaxy*​*P*(*y*)∏*i*=1*n*​*P*(*xi*​∣*y*)

So, finally, we are left with the task of calculating (𝑦)*P*(*y*)and 𝑃(𝑥𝑖∣𝑦)*P*(*xi*​∣*y*).

Please note that (𝑦)*P*(*y*) is also called class probability and 𝑃(𝑥𝑖∣𝑦)*P*(*xi*​∣*y*) is called conditional probability.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of (𝑥𝑖∣𝑦).*P*(*xi*​∣*y*).

Let us try to apply the above formula manually on our weather dataset. For this, we need to do some precomputations on our dataset.

We need to find(𝑥𝑖∣𝑦𝑗)*P*(*xi*​∣*yj*​)for each 𝑥𝑖*xi*​ in X and𝑦𝑗*yj*​ in y. All these calculations have been demonstrated in the tables below:

So, in the figure above, we have calculated (𝑥𝑖 ∣𝑦𝑗)*P*(*xi*​ ∣*yj*​) for each 𝑥𝑖*xi*​ in X and 𝑦𝑗*yj*​ in y manually in the tables 1-4. For example, probability of playing golf given that the temperature is cool, i.e P(temp. = cool | play golf = Yes) = 3/9.

Also, we need to find class probabilities (𝑦)*P*(*y*) which has been calculated in the table 5. For example, P(play golf = Yes) = 9/14.

So now, we are done with our pre-computations and the classifier is ready!

Let us test it on a new set of features (let us call it today):

today = (Sunny, Hot, Normal, False)

𝑃(𝑌𝑒𝑠∣𝑡𝑜𝑑𝑎𝑦)=𝑃(𝑆𝑢𝑛𝑛𝑦𝑂𝑢𝑡𝑙𝑜𝑜𝑘∣𝑌𝑒𝑠)𝑃(𝐻𝑜𝑡𝑇𝑒𝑚𝑝𝑒𝑟𝑎𝑡𝑢𝑟𝑒∣𝑌𝑒𝑠)𝑃(𝑁𝑜𝑟𝑚𝑎𝑙𝐻𝑢𝑚𝑖𝑑𝑖𝑡𝑦∣𝑌𝑒𝑠)𝑃(𝑁𝑜𝑊𝑖𝑛𝑑∣𝑌𝑒𝑠)𝑃(𝑌𝑒𝑠)𝑃(𝑡𝑜𝑑𝑎𝑦)*P*(*Yes*∣*today*)=*P*(*today*)*P*(*SunnyOutlook*∣*Yes*)*P*(*HotTemperature*∣*Yes*)*P*(*NormalHumidity*∣*Yes*)*P*(*NoWind*∣*Yes*)*P*(*Yes*)​

and probability to not play golf is given by:

𝑃(𝑁𝑜∣𝑡𝑜𝑑𝑎𝑦)=𝑃(𝑆𝑢𝑛𝑛𝑦𝑂𝑢𝑡𝑙𝑜𝑜𝑘∣𝑁𝑜)𝑃(𝐻𝑜𝑡𝑇𝑒𝑚𝑝𝑒𝑟𝑎𝑡𝑢𝑟𝑒∣𝑁𝑜)𝑃(𝑁𝑜𝑟𝑚𝑎𝑙𝐻𝑢𝑚𝑖𝑑𝑖𝑡𝑦∣𝑁𝑜)𝑃(𝑁𝑜𝑊𝑖𝑛𝑑∣𝑁𝑜)𝑃(𝑁𝑜)𝑃(𝑡𝑜𝑑𝑎𝑦)*P*(*No*∣*today*)=*P*(*today*)*P*(*SunnyOutlook*∣*No*)*P*(*HotTemperature*∣*No*)*P*(*NormalHumidity*∣*No*)*P*(*NoWind*∣*No*)*P*(*No*)​

Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:

(𝑌𝑒𝑠∣𝑡𝑜𝑑𝑎𝑦)∝39.29.69.69.914≈0.02116*P*(*Yes*∣*today*)∝93​.92​.96​.96​.149​≈0.02116

and

(𝑁𝑜∣𝑡𝑜𝑑𝑎𝑦)∝35.25.15.25.514≈0.0068*P*(*No*∣*today*)∝53​.52​.51​.52​.145​≈0.0068

Now, since

(𝑌𝑒𝑠∣𝑡𝑜𝑑𝑎𝑦)+𝑃(𝑁𝑜∣𝑡𝑜𝑑𝑎𝑦)=1*P*(*Yes*∣*today*)+*P*(*No*∣*today*)=1

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

(𝑌𝑒𝑠∣𝑡𝑜𝑑𝑎𝑦)=0.021160.02116+0.0068≈0.0237*P*(*Yes*∣*today*)=0.02116+0.00680.02116​≈0.0237

and

(𝑁𝑜∣𝑡𝑜𝑑𝑎𝑦)=0.00680.0141+0.0068≈0.33*P*(*No*∣*today*)=0.0141+0.00680.0068​≈0.33

Since

(𝑌𝑒𝑠∣𝑡𝑜𝑑𝑎𝑦)>𝑃(𝑁𝑜∣𝑡𝑜𝑑𝑎𝑦)*P*(*Yes*∣*today*)>*P*(*No*∣*today*)

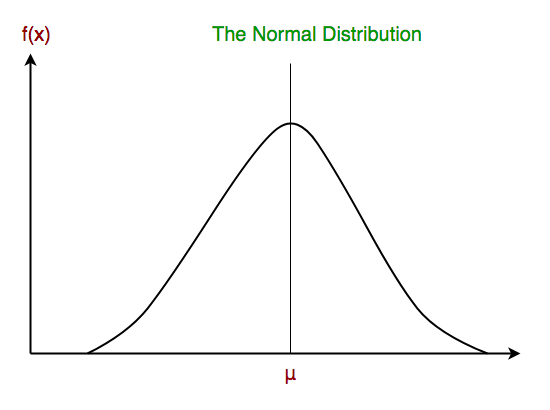
So, prediction that golf would be played is ‘Yes’.

The method that we discussed above is applicable for discrete data. In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of (𝑥𝑖∣𝑦).*P*(*xi*​∣*y*).

### Types of Naive Bayes Model

There are three types of Naive Bayes Model:

#### Gaussian Naive Bayes classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called [Normal distribution](https://en.wikipedia.org/wiki/Normal_distribution) When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:

Updated table of prior probabilities for outlook feature is as following:

The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

𝑃(𝑥𝑖∣𝑦)=12𝜋𝜎𝑦2𝑒𝑥𝑝(−(𝑥𝑖−𝜇𝑦)22𝜎𝑦2)*P*(*xi*​∣*y*)=2*πσy*2​​1​*exp*(−2*σy*2​(*xi*​−*μy*​)2​)

Now, we look at an implementation of Gaussian Naive Bayes classifier using scikit-learn.

|  | **Yes** | **No** | **P(Yes)** | **P(No)** |
| --- | --- | --- | --- | --- |
| **Sunny** | 3 | 2 | 3/9 | 2/5 |
| **Rainy** | 4 | 0 | 4/9 | 0/5 |
| **Overcast** | 2 | 3 | 2/9 | 3/5 |
| **Total** | 9 | 5 | 100% | 100% |

5. What is optimal Bayes classifier?

The Bayes optimal classifier is a probabilistic model that makes the most probable prediction for a new example, given the training dataset.

This model is also referred to as the Bayes optimal learner, the Bayes classifier, Bayes optimal decision boundary, or the Bayes optimal discriminant function.

6. Write any two features of Bayesian learning methods.

* It is one of the simplest and effective methods for calculating the conditional probability and text classification problems.
* A Naïve-Bayes classifier algorithm is better than all other models where assumption of independent predictors holds true.
* It is easy to implement than other models.
* It requires small amount of training data to estimate the test data which minimize the training time period.
* It can be used for Binary as well as Multi-class Classifications.

7. Define the concept of consistent learners.

We’ll define the notion of a consistent learning algorithm, or consistent learner, for a concept class C. 1This is by no means a wholly accurate depiction of the writings of William of Ockham. Those interested in the history are encouraged to look up the original work.

1 Definition 1 (Consistent Learner). We say that an algorithm L is a consistent learner for a concept class C using hypothesis class H, if for all n, for all c ∈ Cn and for all m, given (x1, c(x1)),(x2, c(x2)), . . . ,(xm, c(xm)) as input, where xi ∈ Xn, L outputs h ∈ Hn such that for i = 1, . . . , m, h(xi) = c(xi). We say that L is an efficient consistent learner if the running time of L is polynomial in n, size(c) and m.

A consistent learning algorithm is simply required to output a hypothesis that is consistent with all the training data provided to it. So far, we have not imposed any requirement on the hypothesis class H. This notion of consistency is closely related to the empirical risk minimisation principle in the machine learning literature, where the risk is defined using the zero-one loss.

The main result we will prove that if H is “small enough”, something that is made precise in the theorem below, then a consistent learner can be used to derive a PAC-learning algorithm. This theorem shows that short explanatory hypotheses do in fact also possess predictive power.

Theorem 2 (Occam’s Razor, Cardinality Version). Let C be a concept class and H a hypothesis class. Let L be a consistent learner for C using H. Then for all n ≥ 1, for all c ∈ Cn, for all D over Xn, for all 0 < < 1/2 and all 0 < δ < 1/2, if L is given a sample of size m drawn from EX(c, D), such that,

m ≥ 1 log |Hn| + log 1 δ ,

then L is guaranteed to output a hypothesis h ∈ Hn that with probability at least 1 − δ, satisfies err(h) ≤ .

If further more, L is an efficient consistent learner, log |Hn| is polynomial in n and size(c), and H is polynomially evaluatable, then C is efficiently PAC-learnable using H.

Proof. Fix a target concept c ∈ Cn and the target distribution D over Xn. Call a hypothesis, h ∈ Hn “bad” if err(h) ≥ . Let Ah be the event that m independent samples drawn from EX(c, D) are all consistent with h. Then, if h is bad, P [Ah] ≤ (1 − ) m ≤ e −m.

Consider the event, E = [ h∈Hn:h bad Ah

Then, by a simple application of the union bound we have, P [E] ≤ X h∈Hn:h bad P(Ah) ≤ |Hn| · e −m

Thus, whenever m is larger than the bound given in the statement of the theorem, except with probability δ, no “bad” hypothesis is consistent with m random examples drawn from EX(c, D). However, any hypothesis that is not “bad”, satisfies err(h) ≤ as required.

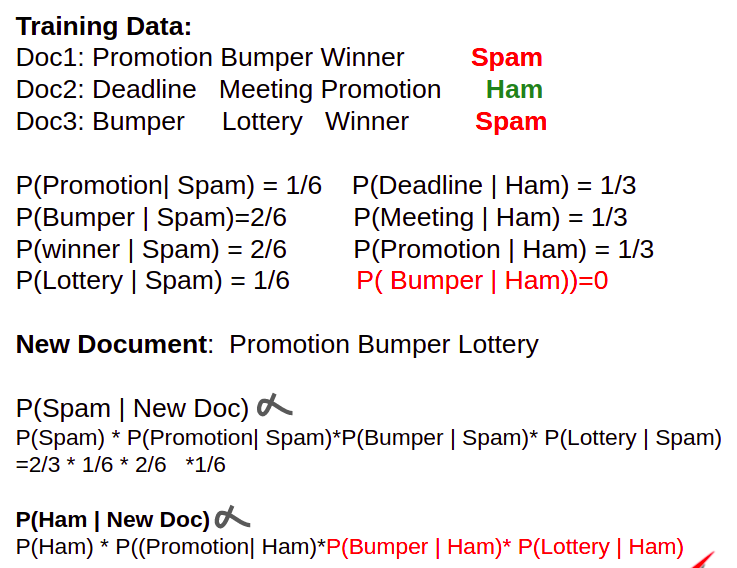
Remark 3. The version of the theorem described above only allows Hn to depend on Cn and n. It is possible to have a much more general version, where Hn may depend also on m, , and δ. As long as log |Hn| is a small enough function of these parameters, a PAC-learning algorithm can still be derived from a consistent learner. These more general versions appear in the book by Kearns and Vazirani (1994, Chap. 2).

8. Write any two strengths of Bayes classifier.

* Simple to Implement. The conditional probabilities are easy to evaluate.
* Very fast – no iterations since the probabilities can be directly computed. So this technique is useful where speed of training is important.
* If the conditional Independence assumption holds, it could give great results

9. Write any two weaknesses of Bayes classifier.

* Conditional Independence Assumption does not always hold. In most situations, the feature show some form of dependency.
* **Zero probability problem :**When we encounter words in the test data for a particular class that are not present in the training data, we might end up with zero class probabilities. See the example below for more details: P(bumper | Ham) is 0 since bumper does not occuer in any ham (non-spam) documents in the training data.

[](https://machinelearninginterview.com/wp-content/uploads/2021/08/image-5.png)

10. Explain how Naïve Bayes classifier is used for

* 1. Text classification

In **natural language processing** and**machine learning**, the **Naïve Bayes**approach is a potent and popular method for classifying text documents. This method classifies documents into predetermined types based on the likelihood of a word occurring, utilizing the concepts of the Bayes theorem. This article aims to implement Document Classification using Naïve Bayes using Python.

**Text Classification using Naive Bayes**

A probabilistic classification technique, the [naïve Bayes](https://www.geeksforgeeks.org/naive-bayes-classifiers/) algorithm is predicated on robust, if naïve, independence assumptions in its probability models. Despite their simplicity, these presumptions serve as the algorithm’s foundation. Even if it frequently deviates from reality, the independence assumption adds to its “naive” characterization.

The Naive Bayes algorithm uses Thomas Bayes’ [Bayes’ theorem](https://www.geeksforgeeks.org/bayes-theorem/), which forms the basis for probability model creation. The method can be trained using these probability models in supervised learning.

**Naive Bayes Algorithm**

The [Naive Bayes algorithm](https://www.geeksforgeeks.org/ml-naive-bayes-scratch-implementation-using-python/) is a probabilistic classification method that bases its predictions on the Bayes theorem. Based on observable data, the Bayes theorem determines a hypothesis’s probability. When using Naive Bayes, an instance’s features serve as the evidence, while the class to which the instance belongs serves as the hypothesis.

The algorithm employing the Bayes theory is broken down as follows:

**Bayes Theorem**

* P(C|F): Probability of the instance belonging to a specific class given its features.
* P(F|C): Probability of observing the features given the class.
* P(C): Prior probability of the class.
* P(F): Probability of observing the features.

The assumption of feature independence is what gives Naive Bayes its “naive” quality. It is computationally efficient since this makes calculations simpler.

Using the Bayes theorem to combine observable data (features) with previous information (prior probabilities) and assume feature independence, Naive Bayes provides predictions. Naive Bayes is efficient in a variety of classification tasks despite its simplicity, particularly in text classification and[natural language processing](https://www.geeksforgeeks.org/natural-language-processing-overview/).

**When to use Naive Bayes**

There are several instances in which Naive Bayes can be applied with great effectiveness. Here are some of those scenarios:

* **Text Classification:** Naive Bayes excels in text-based tasks such as spam filtering, sentiment analysis, and document categorization due to its simplicity and efficiency with high-dimensional data.
* **Limited Training Data**: Naive Bayes can perform well with limited training data, making it valuable when dealing with small datasets or situations where collecting extensive labeled data is challenging.
* **Simple and Quick Prototyping**: When a quick and simple solution is needed for prototyping or baseline performance, Naive Bayes is a suitable choice due to its ease of implementation.

**Implementation to classify text documents using Naive Bayes**

**Importing Libraries**

* Python3

|  |
| --- |
| #importing libraries  import prettytable |

The “[prettytable](https://www.geeksforgeeks.org/creating-tables-with-prettytable-library-python/)” library is imported by the code snippet, indicating a desire to provide tabular data that is aesthetically pleasing. This library is frequently used to present structured data in a table with formatting. Once imported, you can use its features to improve how tabular data is presented in your[Python](https://www.geeksforgeeks.org/introduction-to-python/)code.

**Classification using Naive Bayes**

* Python3

|  |
| --- |
| print('\n \*-----\* Classification using Naïve bayes \*-----\* \n')  total\_documents = int(input("Enter the Total Number of documents: "))  doc\_class = []  i = 0  keywords = []  while not i == total\_documents:      doc\_class.append([])      text = input(f"\nEnter the text of Doc-{i+1} : ").lower()      clas = input(f"Enter the class of Doc-{i+1} : ")      doc\_class[i].append(text.split())      doc\_class[i].append(clas)      keywords.extend(text.split())      i = i+1  keywords = set(keywords)  keywords = list(keywords)  keywords.sort()  to\_find = input(      "\nEnter the Text to classify using Naive Bayes: ").lower().split()    probability\_table = []  for i in range(total\_documents):      probability\_table.append([])      for j in keywords:          probability\_table[i].append(0)  doc\_id = 1  for i in range(total\_documents):      for k in range(len(keywords)):          if keywords[k] in doc\_class[i][0]:              probability\_table[i][k] += doc\_class[i][0].count(keywords[k])  print('\n') |

**Output:**

\*-----\* Classification using Naïve bayes \*-----\*   
Enter the Total Number of documents: 3  
Enter the text of Doc-1 : I watched the movie.   
Enter the class of Doc-1 : +  
Enter the text of Doc-2 : I hated the movie.  
Enter the class of Doc-2 : -  
Enter the text of Doc-3 : poor acting.   
Enter the class of Doc-3 : +  
Enter the Text to classify using Naive Bayes: I hated the acting.

This code starts a basic Naive Bayes [text classification](https://www.geeksforgeeks.org/text-mining-in-data-mining/). The user is prompted to enter the total number of documents, after which it collects details about each document, such as its text and class. After gathering the unique terms (keywords) that appear in every document, a probability table is created to count how many times each keyword appears in every document. When the user submits a text for classification, the likelihood that it belongs in each class is calculated based on the frequency of the term in the training materials. There’s a probability table with the outcomes.

**Probability of Documents**

* Python3

|  |
| --- |
| import prettytable  keywords.insert(0, 'Document ID')  keywords.append("Class")  Prob\_Table = prettytable.PrettyTable()  Prob\_Table.field\_names = keywords  Prob\_Table.title = 'Probability of Documents'  x = 0  for i in probability\_table:      i.insert(0, x+1)      i.append(doc\_class[x][1])      Prob\_Table.add\_row(i)      x = x+1  print(Prob\_Table)  print('\n')  for i in probability\_table:      i.pop(0)  totalpluswords = 0  totalnegwords = 0  totalplus = 0  totalneg = 0  vocabulary = len(keywords)-2  for i in probability\_table:      if i[len(i)-1] == "+":          totalplus += 1          totalpluswords += sum(i[0:len(i)-1])      else:          totalneg += 1          totalnegwords += sum(i[0:len(i)-1])  keywords.pop(0)  keywords.pop(len(keywords)-1) |

**Output:**

+---------------------------------------------------------------------------+  
| Probability of Documents |  
+-------------+---------+-------+---+--------+------+-----+---------+-------+  
| Document ID | acting. | hated | i | movie. | poor | the | watched | Class |  
+-------------+---------+-------+---+--------+------+-----+---------+-------+  
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | + |  
| 2 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | - |  
| 3 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | + |  
+-------------+---------+-------+---+--------+------+-----+---------+-------+

This code generates and shows a probability table by using the “prettytable” package. The keywords are arranged with ‘Document ID’ at the start and ‘Class’ at the conclusion. Field names are set to keywords when creating a [PrettyTable](https://www.geeksforgeeks.org/tag/python-prettytable/) object, and a title is supplied. Next, document IDs and class labels are added to the table together with the probability values from the probability\_table. The code determines the total number of occurrences and words for each class (‘+’ and ‘-‘) after printing the probability table. For additional examination, it modifies the vocabulary size and eliminates pointless components from the list of keywords.

**Positive Class**

* Python3

|  |
| --- |
| # For positive class  temp = []  for i in to\_find:      count = 0      x = keywords.index(i)      for j in probability\_table:          if j[len(j)-1] == "+":              count = count+j[x]      temp.append(count)      count = 0  for i in range(len(temp)):      temp[i] = format((temp[i]+1)/(vocabulary+totalpluswords), ".4f")  print()  temp = [float(f) for f in temp]  print("Probabilities of Each word to be in '+' class are: ")  h = 0  for i in to\_find:      print(f"P({i}/+) = {temp[h]}")      h = h+1  print()  pplus = float(format((totalplus)/(totalplus+totalneg), ".8f"))  for i in temp:      pplus = pplus\*i  pplus = format(pplus, ".8f")  print("probability of Given text to be in '+' class is :", pplus)  print() |

**Output:**

Probabilities of Each word to be in '+' class are:   
P(i/+) = 0.1429  
P(hated/+) = 0.0714  
P(the/+) = 0.1429  
P(acting/+) = 0.1429  
probability of Given text to be in '+' class is : 0.00013890

With the input text, this code calculates the likelihood that each word belongs to the positive class (‘+’). Iteratively going over each word in “to\_find,” it determines how often each word occurs in the positive class based on the probability table and uses [Laplace smoothing](https://www.geeksforgeeks.org/applying-multinomial-naive-bayes-to-nlp-problems/)to obtain the conditional probabilities. After that, the results are written out, displaying the probability of each word receiving the positive class. Lastly, it uses these word probabilities to compute the overall chance that the input text belongs to the positive class, and it prints the outcome. Non-zero probabilities for unseen words are guaranteed by the Laplace smoothing.

**Negative class**

* Python3

|  |
| --- |
| # For Negative class  temp = []  for i in to\_find:      count = 0      x = keywords.index(i)      for j in probability\_table:          if j[len(j)-1] == "-":              count = count+j[x]      temp.append(count)      count = 0  for i in range(len(temp)):      temp[i] = format((temp[i]+1)/(vocabulary+totalnegwords), ".4f")  print()  temp = [float(f) for f in temp]  print("Probabilities of Each word to be in '-' class are: ")  h = 0  for i in to\_find:      print(f"P({i}/-) = {temp[h]}")      h = h+1  print()  pneg = float(format((totalneg)/(totalplus+totalneg), ".8f"))  for i in temp:      pneg = pneg\*i  pneg = format(pneg, ".8f")  print("probability of Given text to be in '-' class is :", pneg)  print('\n') |

**Output:**

Probabilities of Each word to be in '-' class are:   
P(i/-) = 0.1667  
P(hated/-) = 0.1667  
P(the/-) = 0.1667  
P(acting/-) = 0.0833  
probability of Given text to be in '-' class is : 0.00012863

The probability that each word in the input text belongs to the negative class (‘-‘) are calculated by this code. Iterating through every word in “to\_find,” it determines each word’s occurrences in the negative class using the probability table, and then computes conditional probabilities using Laplace smoothing, just like the positive class computation does. The probability of each word being assigned to the negative class is then printed along with the findings. Lastly, it uses these word probabilities to compute the overall chance that the input text belongs to the negative class, and it prints the result. In both positive and negative class calculations, the Laplace smoothing guarantees non-zero probabilities for unseen words.

**Prediction**

* Python3

|  |
| --- |
| if pplus > pneg:      print(          f"Using Naive Bayes Classification, We can clearly say that the given text belongs to '+' class with probability {pplus}")  else:      print(          f"Using Naive Bayes Classification, We can clearly say that the given text belongs to '-' class with probability {pneg}")  print('\n') |

**Output:**

Probabilities of Each word to be in '+' class are:   
P(i/+) = 0.1538  
P(hated/+) = 0.0769  
P(the/+) = 0.1538  
P(acting./+) = 0.1538  
probability of Given text to be in '+' class is : 0.00018651  
Probabilities of Each word to be in '-' class are:   
P(i/-) = 0.1818  
P(hated/-) = 0.1818  
P(the/-) = 0.1818  
P(acting./-) = 0.0909  
probability of Given text to be in '-' class is : 0.00018206  
Using Naive Bayes Classification, We can clearly say that the given text belongs to '+' class with probability 0.00018651

2. Spam filtering

A spam [filter](https://www.techtarget.com/whatis/definition/filter) is a program used to detect unsolicited, unwanted and [virus](https://www.techtarget.com/searchsecurity/definition/virus)-infected emails and prevent those messages from getting to a user's inbox. Like other types of filtering programs, a spam filter looks for specific criteria on which to base its judgments.

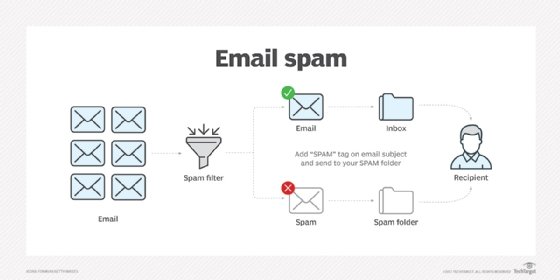
Internet service providers ([ISPs](https://www.techtarget.com/whatis/definition/ISP-Internet-service-provider)), free online email services and businesses use [email spam](https://www.techtarget.com/searchsecurity/definition/spam) filtering tools to minimize the risk of distributing spam. For example, one of the simplest and earliest versions of spam filtering, like the one that was used by Microsoft's Hotmail, was set to watch out for particular words in the subject lines of messages. An email was excluded from the user's inbox whenever the filter recognized one of the specified words.

This method is not especially effective and often omits perfectly legitimate messages, called *false positives*, while letting actual spam messages through.

More sophisticated programs, such as [Bayesian filters](https://www.techtarget.com/whatis/definition/Bayesian-filter) and other [heuristic](https://www.techtarget.com/whatis/definition/heuristic) filters, identify spam messages by recognizing suspicious word patterns or word frequency. They do this by learning the user's preferences based on the emails marked as spam. The spam software then creates rules and applies them to future emails that target the user's inbox.

For example, whenever users mark emails from a specific sender as spam, the Bayesian filter recognizes the pattern and automatically moves future emails from that sender to the spam folder.

ISPs apply spam filters to both inbound and outbound emails. However, [small to medium enterprises](https://www.techtarget.com/whatis/definition/small-to-medium-enterprise-SME) usually focus on inbound filters to protect their network. There are also many different spam filtering solutions available. They can be hosted in the cloud, hosted on servers or integrated into email software, such as Microsoft Outlook.



3. Market sentiment analysis

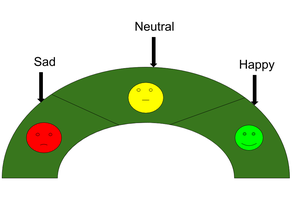
# What is Sentiment Analysis?

**Sentiment analysis** is a popular task in**natural language processing**. The goal of sentiment analysis is to classify the text based on the mood or mentality expressed in the text, which can be positive negative, or neutral.

## What is Sentiment Analysis?

[Sentiment analysis](https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/) is the process of classifying whether a block of text is positive, negative, or neutral. The goal that Sentiment mining tries to gain is to be analysed people’s opinions in a way that can help businesses expand. It focuses not only on polarity (positive, negative & neutral) but also on emotions (happy, sad, angry, etc.). It uses various [Natural Language Processing](https://www.geeksforgeeks.org/natural-language-processing-nlp-tutorial/) algorithms such as Rule-based, Automatic, and Hybrid.

let’s consider a scenario, if we want to analyze whether a product is satisfying customer requirements, or is there a need for this product in the market. We can use sentiment analysis to monitor that product’s reviews. Sentiment analysis is also efficient to use when there is a large set of unstructured data, and we want to classify that data by automatically tagging it. Net Promoter Score (NPS) surveys are used extensively to gain knowledge of how a customer perceives a product or service. Sentiment analysis also gained popularity due to its feature to process large volumes of NPS responses and obtain consistent results quickly.



*Sentiment*

## Why is Sentiment Analysis Important?

Sentiment analysis is the contextual meaning of words that indicates the social sentiment of a brand and also helps the business to determine whether the product they are manufacturing is going to make a demand in the market or not.

According to the survey,80% of the world’s data is unstructured. The data needs to be analyzed and be in a structured manner whether it is in the form of emails, texts, documents, articles, and many more.

1. Sentiment Analysis is required as it stores data in an efficient, cost friendly.
2. Sentiment analysis solves real-time issues and can help you solve all real-time scenarios.

Here are some key reasons why sentiment analysis is important for business:

* **Customer Feedback Analysis**: Businesses can analyze customer reviews, comments, and feedback to understand the sentiment behind them helping in identifying areas for improvement and addressing customer concerns, ultimately enhancing customer satisfaction.
* **Brand Reputation Management**: Sentiment analysis allows businesses to monitor their brand reputation in real-time.  
  By tracking mentions and sentiments on social media, review platforms, and other online channels, companies can respond promptly to both positive and negative sentiments, mitigating potential damage to their brand.
* **Product Development and Innovation**: Understanding customer sentiment helps identify features and aspects of their products or services that are well-received or need improvement. This information is invaluable for product development and innovation, enabling companies to align their offerings with customer preferences.
* **Competitor Analysis**: Sentiment Analysis can be used to compare the sentiment around a company’s products or services with those of competitors.  
  Businesses identify their strengths and weaknesses relative to competitors, allowing for strategic decision-making.
* **Marketing Campaign Effectiveness**  
  Businesses can evaluate the success of their marketing campaigns by analyzing the sentiment of online discussions and social media mentions.  
  Positive sentiment indicates that the campaign is resonating with the target audience, while negative sentiment may signal the need for adjustments.