A Bayesian network (has a good wikipedia page) models relationships between features in a very general way. If you know what these relationships are, or have enough data to derive them, then it may be appropriate to use a Bayesian network.

A Naive Bayes classifier is a simple model that describes particular class of Bayesian network - where all of the features are class-conditionally independent. Because of this, there are certain problems that Naive Bayes cannot solve (example below). However, its simplicity also makes it easier to apply, and it requires less data to get a good result in many cases.

Naive Bayes is just a restricted/constrained form of a general Bayesian network where you enforce the constraint that the class node should have no parents and that the nodes corresponding to the attribute variables should have no edges between them.

For the Bayesian network as a classifier, the features are selected based on some scoring functions like Bayesian scoring function and minimal description length(the two are equivalent in theory to each other given that there are enough training data). The scoring functions mainly restrict the structure (connections and directions) and the parameters(likelihood) using the data. After the structure has been learned the class is only determined by the nodes in the Markov blanket(its parents, its children, and the parents of its children), and all variables given the Markov blanket are discarded.

For the Naive Bayesian Network which is more well-known nowadays, all features are considered as attributes and are independent given the class.

What are the main advantages of Naive Bayes?

The main advantages are:

It is fast, and easy to understand

It is not prone to overfitting

It does not need much training data

Like all other algorithms, Naive Bayes has its own set of cons as well:

* It does not work that well when the number of features is very high
* The assumption that input features are independent of each other may not always hold true
* You might lose important information while discretising the continuous variables.

[Beginner](https://www.analyticsvidhya.com/blog/category/beginner/) [Data Science](https://www.analyticsvidhya.com/blog/category/data-science/) [Machine Learning](https://www.analyticsvidhya.com/blog/category/machine-learning/) [Maths](https://www.analyticsvidhya.com/blog/category/maths/) [Probability](https://www.analyticsvidhya.com/blog/category/probability/) [Python](https://www.analyticsvidhya.com/blog/category/python-2/) [R](https://www.analyticsvidhya.com/blog/category/r/) [Technique](https://www.analyticsvidhya.com/blog/category/technique/)

***Note: This article was originally published on Sep 13th, 2015 and updated on Sept 11th, 2017***

**Overview**

* Understand one of the most popular and simple [machine learning](https://www.analyticsvidhya.com/machine-learning/?utm_source=blog&utm_medium=naive-bayes) classification algorithms, the Naive Bayes algorithm
* It is based on the Bayes Theorem for calculating probabilities and conditional probabilities
* Learn how to implement the Naive Bayes Classifier in R and Python

**Introduction**

Here’s a situation you’ve got into in your [data science](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle) project:

You are working on a classification problem and have generated your set of hypothesis, created features and discussed the importance of variables. Within an hour, stakeholders want to see the first cut of the model.

What will you do? You have hundreds of thousands of data points and quite a few variables in your training data set. In such a situation, if I were in your place, I would have used ‘**Naive Bayes**‘, which can be extremely fast relative to other [classification algorithms](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle). It works on Bayes theorem of probability to predict the class of unknown data sets.

In this article, I’ll explain the basics of this algorithm, so that next time when you come across large data sets, you can bring this algorithm to action. In addition, if you are a [newbie in Python](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle) or R, you should not be overwhelmed by the presence of available codes in this article.

If you prefer to learn Naive Bayes theorem from the basics concepts to the implementation in a structured manner, you can enroll in this free course:

* [Naive Bayes from Scratch](https://courses.analyticsvidhya.com/courses/naive-bayes?utm_source=blog&utm_medium=naive-bayes-explained)

Are you a beginner in Machine Learning? Do you want to master the machine learning algorithms like Naive Bayes? Here is a comprehensive course covering the machine learning and deep learning algorithms in detail –

* [Certified AI & ML Blackbelt+ Program](https://courses.analyticsvidhya.com/bundles/certified-ai-ml-blackbelt-plus?utm_source=blog&utm_medium=naive-bayes-explained)

|  |
| --- |
| **Project to apply Naive Bayes**  **Problem Statement**  HR analytics is revolutionizing the way human resources departments operate, leading to higher efficiency and better results overall. Human resources have been using analytics for years.  However, the collection, processing, and analysis of data have been largely manual, and given the nature of human resources dynamics and HR KPIs, the approach has been constraining HR. Therefore, it is surprising that HR departments woke up to the utility of [machine learning](https://www.analyticsvidhya.com/machine-learning/?utm_source=blog&utm_medium=naive-bayes) so late in the game. Here is an opportunity to try predictive analytics in identifying the employees most likely to get promoted.  [**Practice Now**](http://datahack.analyticsvidhya.com/contest/wns-analytics-hackathon-2018-1/?utm_source=av_blog&utm_medium=practice_blog_naive_bayes) |

**Table of Contents**

1. What is Naive Bayes algorithm?
2. How Naive Bayes Algorithms works?
3. What are the Pros and Cons of using Naive Bayes?
4. 4 Applications of Naive Bayes Algorithm
5. Steps to build a basic Naive Bayes Model in Python
6. Tips to improve the power of Naive Bayes Model

**What is Naive Bayes algorithm?**

It is a [classification technique](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle) based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

Above,

* *P*(*c|x*) is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
* *P*(*c*) is the prior probability of *class*.
* *P*(*x|c*) is the likelihood which is the probability of *predictor* given *class*.
* *P*(*x*) is the prior probability of *predictor*.

**How Naive Bayes algorithm works?**

Let’s understand it using an example. Below I have a training data set of weather and corresponding target variable ‘Play’ (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let’s follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Step 3: Now, use [Naive Bayesian](https://courses.analyticsvidhya.com/courses/naive-bayes?utm_source=blog&utm_medium=naive-bayes-explained) equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

**Problem:** Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P( Sunny | Yes) \* P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P( Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 \* 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

[Beginner](https://www.analyticsvidhya.com/blog/category/beginner/) [Data Science](https://www.analyticsvidhya.com/blog/category/data-science/) [Machine Learning](https://www.analyticsvidhya.com/blog/category/machine-learning/) [Maths](https://www.analyticsvidhya.com/blog/category/maths/) [Probability](https://www.analyticsvidhya.com/blog/category/probability/) [Python](https://www.analyticsvidhya.com/blog/category/python-2/) [R](https://www.analyticsvidhya.com/blog/category/r/) [Technique](https://www.analyticsvidhya.com/blog/category/technique/)

***Note: This article was originally published on Sep 13th, 2015 and updated on Sept 11th, 2017***

**Overview**

* Understand one of the most popular and simple [machine learning](https://www.analyticsvidhya.com/machine-learning/?utm_source=blog&utm_medium=naive-bayes) classification algorithms, the Naive Bayes algorithm
* It is based on the Bayes Theorem for calculating probabilities and conditional probabilities
* Learn how to implement the Naive Bayes Classifier in R and Python

**Introduction**

Here’s a situation you’ve got into in your [data science](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle) project:

You are working on a classification problem and have generated your set of hypothesis, created features and discussed the importance of variables. Within an hour, stakeholders want to see the first cut of the model.

What will you do? You have hundreds of thousands of data points and quite a few variables in your training data set. In such a situation, if I were in your place, I would have used ‘**Naive Bayes**‘, which can be extremely fast relative to other [classification algorithms](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle). It works on Bayes theorem of probability to predict the class of unknown data sets.

In this article, I’ll explain the basics of this algorithm, so that next time when you come across large data sets, you can bring this algorithm to action. In addition, if you are a [newbie in Python](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle) or R, you should not be overwhelmed by the presence of available codes in this article.

If you prefer to learn Naive Bayes theorem from the basics concepts to the implementation in a structured manner, you can enroll in this free course:

* [Naive Bayes from Scratch](https://courses.analyticsvidhya.com/courses/naive-bayes?utm_source=blog&utm_medium=naive-bayes-explained)

Are you a beginner in Machine Learning? Do you want to master the machine learning algorithms like Naive Bayes? Here is a comprehensive course covering the machine learning and deep learning algorithms in detail –

* [Certified AI & ML Blackbelt+ Program](https://courses.analyticsvidhya.com/bundles/certified-ai-ml-blackbelt-plus?utm_source=blog&utm_medium=naive-bayes-explained)

|  |
| --- |
| **Project to apply Naive Bayes**  **Problem Statement**  HR analytics is revolutionizing the way human resources departments operate, leading to higher efficiency and better results overall. Human resources have been using analytics for years.  However, the collection, processing, and analysis of data have been largely manual, and given the nature of human resources dynamics and HR KPIs, the approach has been constraining HR. Therefore, it is surprising that HR departments woke up to the utility of [machine learning](https://www.analyticsvidhya.com/machine-learning/?utm_source=blog&utm_medium=naive-bayes) so late in the game. Here is an opportunity to try predictive analytics in identifying the employees most likely to get promoted.  [**Practice Now**](http://datahack.analyticsvidhya.com/contest/wns-analytics-hackathon-2018-1/?utm_source=av_blog&utm_medium=practice_blog_naive_bayes) |

**Table of Contents**

1. What is Naive Bayes algorithm?
2. How Naive Bayes Algorithms works?
3. What are the Pros and Cons of using Naive Bayes?
4. 4 Applications of Naive Bayes Algorithm
5. Steps to build a basic Naive Bayes Model in Python
6. Tips to improve the power of Naive Bayes Model

**What is Naive Bayes algorithm?**

It is a [classification technique](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle) based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

Above,

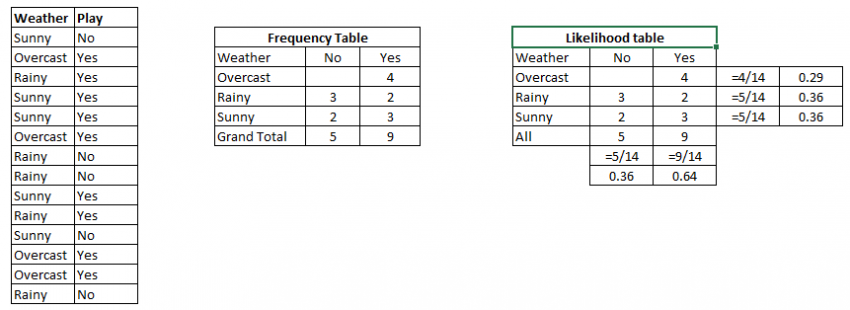
* *P*(*c|x*) is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
* *P*(*c*) is the prior probability of *class*.
* *P*(*x|c*) is the likelihood which is the probability of *predictor* given *class*.
* *P*(*x*) is the prior probability of *predictor*.

**How Naive Bayes algorithm works?**

Let’s understand it using an example. Below I have a training data set of weather and corresponding target variable ‘Play’ (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let’s follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.



Step 3: Now, use [Naive Bayesian](https://courses.analyticsvidhya.com/courses/naive-bayes?utm_source=blog&utm_medium=naive-bayes-explained) equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

**Problem:** Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P( Sunny | Yes) \* P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P( Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 \* 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

**What are the Pros and Cons of Naive Bayes?**

***Pros:***

* It is easy and fast to predict class of test data set. It also perform well in multi class prediction
* When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
* It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

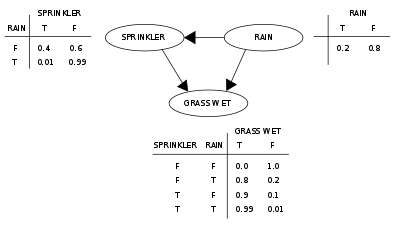
***Cons:***

* If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
* On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.
* Another limitation of [Naive Bayes](https://courses.analyticsvidhya.com/courses/naive-bayes?utm_source=blog&utm_medium=naive-bayes-explained) is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Naive Bayesian classifier (NBC) assumes independence between all attributes given a class, which is seldom true. That is why it is called Naive. In contrast, in a Bayesian network (BN) you can make a more detailed (true) model of the problem using several layers of dependecies. However, NBC can actually be modelled as a simple form of NB. So there is actually not a clear cut difference between them, other than that NBC makes a very much simplified assumption. The question is rather whether a NBC is good enough to do the job, rather than whether which approach is best.

There is one more reason why an unrestricted Bayesian network (BN) may perform better.  In the case of a naive Bayes (NB) network, the class node does not have any parents.  Parents of the class node, which behave differently from its children (in terms of independence conditions), may offer a way of influencing the prior probability distribution over classes.

In theory a full BN should perform better but in practice, precisely because of the Markov blanket around the class node, when there are not too many missing vales, nodes around the class node may shield it from the influences of less important features. Less important does not mean worthless in terms of predicting the class.  I have seen cases in which simple NB networks performed better than BNs, even though they should have performed worse (for the reasons outlined by Tomas and Vikas above).

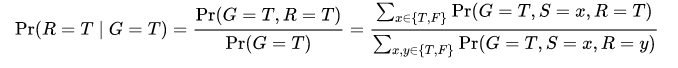
Two events can cause grass to be wet: an active sprinkler or rain. Rain has a direct effect on the use of the sprinkler (namely that when it rains, the sprinkler usually is not active). This situation can be modeled with a Bayesian network (shown to the right). Each variable has two possible values, T (for true) and F (for false). 

The [joint probability function](https://en.wikipedia.org/wiki/Joint_probability_distribution) is, by the [chain rule of probability](https://en.wikipedia.org/wiki/Chain_rule_of_probability),

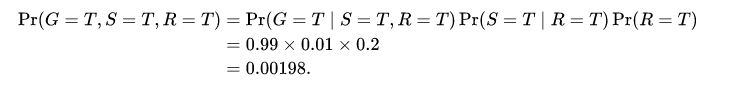
Pr ( G , S , R ) = Pr ( G ∣ S , R ) Pr ( S ∣ R ) Pr ( R ) {\displaystyle \Pr(G,S,R)=\Pr(G\mid S,R)\Pr(S\mid R)\Pr(R)}

where *G* = "Grass wet (true/false)", *S* = "Sprinkler turned on (true/false)", and *R* = "Raining (true/false)".

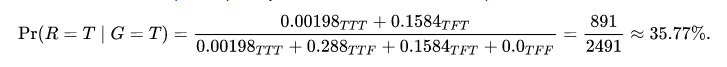
The model can answer questions about the presence of a cause given the presence of an effect (so-called inverse probability) like "What is the probability that it is raining, given the grass is wet?" by using the [conditional probability](https://en.wikipedia.org/wiki/Conditional_probability) formula and summing over all [nuisance variables](https://en.wikipedia.org/wiki/Nuisance_variable):

Pr ( R = T ∣ G = T ) = Pr ( G = T , R = T ) Pr ( G = T ) = ∑ x ∈ { T , F } Pr ( G = T , S = x , R = T ) ∑ x , y ∈ { T , F } Pr ( G = T , S = x , R = y ) {\displaystyle \Pr(R=T\mid G=T)={\frac {\Pr(G=T,R=T)}{\Pr(G=T)}}={\frac {\sum \_{x\in \{T,F\}}\Pr(G=T,S=x,R=T)}{\sum \_{x,y\in \{T,F\}}\Pr(G=T,S=x,R=y)}}}

Using the expansion for the joint probability function Pr ( G , S , R ) {\displaystyle \Pr(G,S,R)} and the conditional probabilities from the [conditional probability tables (CPTs)](https://en.wikipedia.org/wiki/Conditional_probability_table) stated in the diagram, one can evaluate each term in the sums in the numerator and denominator. For example,

Pr ( G = T , S = T , R = T ) = Pr ( G = T ∣ S = T , R = T ) Pr ( S = T ∣ R = T ) Pr ( R = T ) = 0.99 × 0.01 × 0.2 = 0.00198. {\displaystyle {\begin{aligned}\Pr(G=T,S=T,R=T)&=\Pr(G=T\mid S=T,R=T)\Pr(S=T\mid R=T)\Pr(R=T)\\&=0.99\times 0.01\times 0.2\\&=0.00198.\end{aligned}}} 

Then the numerical results (subscripted by the associated variable values) are

Pr ( R = T ∣ G = T ) = 0.00198 T T T + 0.1584 T F T 0.00198 T T T + 0.288 T T F + 0.1584 T F T + 0.0 T F F = 891 2491 ≈ 35.77 % . {\displaystyle \Pr(R=T\mid G=T)={\frac {0.00198\_{TTT}+0.1584\_{TFT}}{0.00198\_{TTT}+0.288\_{TTF}+0.1584\_{TFT}+0.0\_{TFF}}}={\frac {891}{2491}}\approx 35.77\%.} 

To answer an interventional question, such as "What is the probability that it would rain, given that we wet the grass?" the answer is governed by the post-intervention joint distribution function



obtained by removing the factor Pr ( G ∣ S , R ) {\displaystyle \Pr(G\mid S,R)} from the pre-intervention distribution. The do operator forces the value of G to be true. The probability of rain is unaffected by the action:



Pr ( R ∣ do ( G = T ) ) = Pr ( R ) . {\displaystyle \Pr(R\mid {\text{do}}(G=T))=\Pr(R).}

To predict the impact of turning the sprinkler on:

Pr ( R , G ∣ do ( S = T ) ) = Pr ( R ) Pr ( G ∣ R , S = T ) {\displaystyle \Pr(R,G\mid {\text{do}}(S=T))=\Pr(R)\Pr(G\mid R,S=T)} 

with the term Pr ( S = T ∣ R ) {\displaystyle \Pr(S=T\mid R)} removed, showing that the action affects the grass but not the rain.

These predictions may not be feasible given unobserved variables, as in most policy evaluation problems. The effect of the action do ( x ) {\displaystyle {\text{do}}(x)} can still be predicted, however, whenever the back-door criterion is satisfied.[[1]](https://en.wikipedia.org/wiki/Bayesian_network#cite_note-pearl2000-1)[[2]](https://en.wikipedia.org/wiki/Bayesian_network#cite_note-2) It states that, if a set *Z* of nodes can be observed that [*d*-separates](https://en.wikipedia.org/wiki/Bayesian_network#d-separation)[[3]](https://en.wikipedia.org/wiki/Bayesian_network#cite_note-3) (or blocks) all back-door paths from *X* to *Y* then

Pr ( Y , Z ∣ do ( x ) ) = Pr ( Y , Z , X = x ) Pr ( X = x ∣ Z ) . {\displaystyle \Pr(Y,Z\mid {\text{do}}(x))={\frac {\Pr(Y,Z,X=x)}{\Pr(X=x\mid Z)}}.}

A back-door path is one that ends with an arrow into *X*. Sets that satisfy the back-door criterion are called "sufficient" or "admissible." For example, the set *Z* = *R* is admissible for predicting the effect of *S* = *T* on *G*, because *R* *d*-separates the (only) back-door path *S* ← *R* → *G*. However, if *S* is not observed, no other set *d*-separates this path and the effect of turning the sprinkler on (*S* = *T*) on the grass (*G*) cannot be predicted from passive observations. In that case *P*(*G* | do(*S* = *T*)) is not "identified". This reflects the fact that, lacking interventional data, the observed dependence between *S* and *G* is due to a causal connection or is spurious (apparent dependence arising from a common cause, *R*). (see [Simpson's paradox](https://en.wikipedia.org/wiki/Simpson%27s_paradox))

"Bayesian Network Classifiers, Machine Learning, 29, 131–163 (1997)". Of particular interest is section 3.