**Naive Bayes**

* <https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>
* Another best Reference: [**https://www.kaggle.com/pranavpandey2511/naive-bayes-classifier-from-scratch**](https://www.kaggle.com/pranavpandey2511/naive-bayes-classifier-from-scratch)
* Best Reference:[**https://www.geeksforgeeks.org/naive-bayes-classifiers/**](https://www.geeksforgeeks.org/naive-bayes-classifiers/)

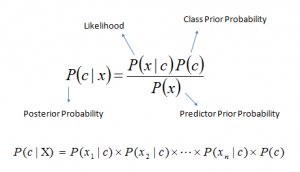
## **Table of Contents**

1. What is the 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

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.

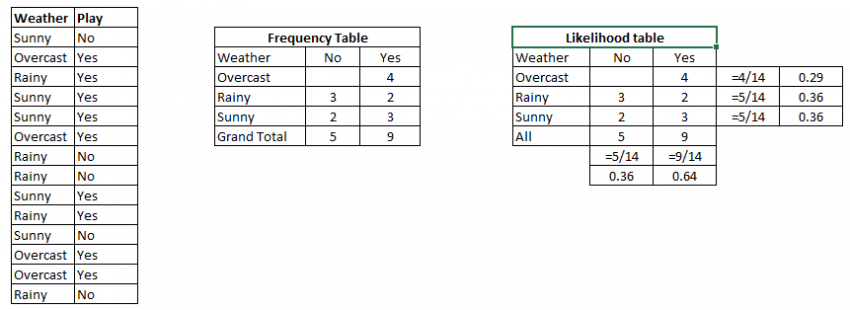
Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



## **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 conditions. Let’s follow the below steps to perform it.

Step 1: Convert the data set into a frequency table



Step 3: Now, use the Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of the prediction.

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

We can solve it using the above-discussed method of the 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 classes 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 the class of the test data set. It also performs well in multi-class prediction
* When the assumption of independence holds, a Naive Bayes classifier performs better compared to other models like logistic regression and you need less training data.
* It performs well in the case of categorical input variables compared to a numerical variable(s). For numerical variables, the normal distribution is assumed (bell curve, which is a strong assumption).

**Why should we use Naive Bayes?**

* As stated above, It is ***easy*** to build and is particularly useful for ***very large data sets***.
* It is **extremely fast** for both training and prediction.
* It provides a straightforward probabilistic prediction.
* It is often very easily interpretable.
* It has very few (if any) tunable parameters.
* It performs well in the case of categorical input variables compared to a numerical variable(s). For numerical variables, the normal distribution is assumed (bell curve, which is a strong assumption).

**Cons:**

* If a categorical variable has a category (in the test data set), which was not observed in the training data set, then the 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 is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors that are completely independent.

## 4 **Applications of Naive Bayes Algorithms**

* Real-time Prediction: Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real-time.
* Multi-class Prediction: This algorithm is also well known for multi-class prediction feature. Here we can predict the probability of multiple classes of the target variables.
* Text classification/ Spam Filtering/ Sentiment Analysis: Naive Bayes classifiers mostly used in text classification (due to better results in multi-class problems and independence rule) have a higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
* Recommendation System: Naive Bayes Classifier and [Collaborative Filtering](https://en.wikipedia.org/wiki/Collaborative_filtering) together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

## H**ow to build a basic model using Naive Bayes in Python and R?**

Again, scikit learn (python library) will help here to build a Naive Bayes model in Python. There are three types of Naive Bayes model under the scikit-learn library:

* [Gaussian:](http://scikit-learn.org/stable/modules/naive_bayes.html) It is used in classification and it assumes that features follow a normal distribution.
* [Multinomial](http://scikit-learn.org/stable/modules/naive_bayes.html): It is used for discrete counts. For example, let’s say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of “word occurring in the document”, we have “count how often word occurs in the document”, you can think of it as “number of times outcome number x\_i is observed over the n trials”.
* [Bernoulli](http://scikit-learn.org/stable/modules/naive_bayes.html): The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with a ‘bag of words’ model where the 1s & 0s are “word occurs in the document” and “word does not occur in the document” respectively.

### **Popular Variants of Naive Bayes Classifier (**[**https://en.wikipedia.org/wiki/Naive\_Bayes\_classifier**](https://en.wikipedia.org/wiki/Naive_Bayes_classifier)**):**

* The conventional version of the Naive Bayes is the **Gaussian NB**, which works best for continuous types of data. The underlying assumption of Gaussian NB is that the features follow a normal distribution.
* The other variants which are discussed in this section are best used for text classification problems, wherein the data features are discrete. **BernoulliNB** is the Naive Bayes version similar to the one described in the “bag of words” type, wherein the features are vectorized in a binary fashion.
* Whereas, **MultinomialNB** is the non-binary version of BernoulliNB. As the word implies, Multinomial means “many counts”. Furthermore, ComplementNB implements the Complement Naive Bayes (CNB) algorithm. CNB is an adaptation of the standard Multinomial Naive Bayes (MNB) algorithm that is particularly suited for imbalanced data sets wherein the algorithm uses statistics from the *complement* of each class to compute the model’s weight.
* The inventors of CNB show empirically that the parameter estimates for CNB are more stable than those for MNB. Further, CNB regularly outperforms MNB (often by a considerable margin) on text classification tasks. Below is the comparison of test simulation results on each of these classifiers with the corresponding codes.

