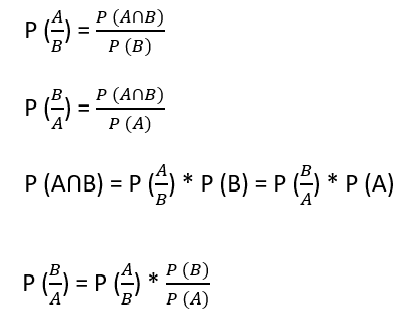
**Naive Bays Algorithm**

**Naive Bays Algorithm:**

Naive Bays classifier is a straightforward and powerful algorithm for the [**classification**](https://dataaspirant.com/2016/09/24/classification-clustering-alogrithms/) task. It works on

Bays theorem of probability to predict the class of unknown data set.

**How Naive Bays classifier algorithm works in machine learning:**

****

* Basically, we are trying to find probability of event A, given the event B is true. Event B is also

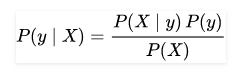
Termed as **evidence**.

* 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 (A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

we can apply Bays’ theorem in following way:

****

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

X=(x1,x2,x3,......xn)

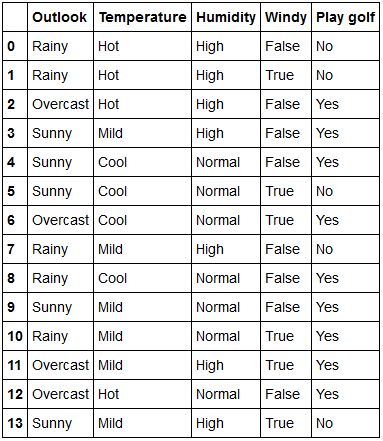
X=(Rainy,Hot,High,False)

y=No

So basically, P(X|y) here means, the probability of “Not playing golf” given that the weather conditions

are “Rainy outlook”, “Temperature is hot”, “high humidity” and “no wind”.

ex :



The dataset is divided into two parts, namely, **features** 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’.

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.

**probability of Outlook**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Outlook** | yes | no | P(yes) | P(no) |
| Rainy | 2 | 3 | 2/9 | 3/5 |
| Sunny | 3 | 2 | 3/9 | 2/5 |
| Overcast | 4 | 0 | 4/9 | 0/5 |
| total | 9 | 5 | 100% | 100% |

**probability of Temperature:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Temperature** | yes | no | P(yes) | P(no) |
| Hot | 2 | 2 | 2/9 | 2/5 |
| Mild | 4 | 2 | 4/9 | 2/5 |
| Cool | 3 | 1 | 3/9 | 1/5 |
| total | 9 | 5 | 100% | 100% |

**probability of Humidity:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Humidity** | yes | no | P(yes) | P(no) |
| High | 3 | 4 | 3/9 | 4/5 |
| Normal | 6 | 1 | 6/9 | 1/5 |
| Total | 9 | 5 | 100% | 100% |

**Probability of Wind yes and no:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **wind** | yes | no | P(yes) | P(no) |
| False(NO) | 6 | 2 | 6/9 | 2/5 |
| True(YES) | 3 | 3 | 3/9 | 3/5 |
| Total | 9 | 5 | 100% | 100% |

**probability of play:**

|  |  |  |
| --- | --- | --- |
| **Play** |  | P(yes)/p(no) |
| Yes | 9 | 9/14 |
| No | 5 | 5/14 |
| total | 14 | 100% |

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

today=(Sunny,Hot,Normal,False)

P(yes/today)=P(sunny/yes)P(Hot/yes)P(Normal/yes)P(Nowind/yes)P(yes)/P(today)

3/9\*2/9\*6/9\*6/9\*9/14=0.0211

P(no/today)=P(sunny/no)P(Hot/no)P(Normal/no)P(Nowind/no)P(no)/P(today)

2/5\*2/5\*1/5\*2/5\*5/14=0.0045

now since,

P(yes/today)+P(no/today)=1

P(yes/today)=0.0211/(0.0211+0.0045)=>0.0211/0.0256=>0.82

P(no/today)=0.0045/(0.0211+0.0045)=>0.0045/0.0256=>0.17

**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 Bays classifiers differ mainly by the assumptions they make regarding the distribution

Of P (xi | y).

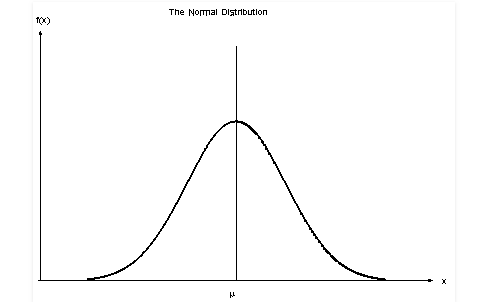
**Gaussian Naive Bays classifier:**

In Gaussian Naive Bays, continuous values associated with each feature are assumed to be

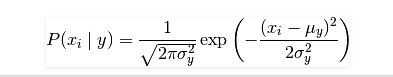
Distributed according to a **Gaussian distribution**. A Gaussian distribution is also called [Normal distribution](https://en.wikipedia.org/wiki/Normal_distribution).

When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values

as shown below:



The conditional probability is given by:



Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes’ theorem with the “naive” assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1through x_n, Bayes’ theorem states the following relationship:

P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}
                                 {P(x_1, \dots, x_n)}

Using the naive independence assumption that

P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y),

for all i, this relationship is simplified to

P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}
                                 {P(x_1, \dots, x_n)}

Since P(x_1, \dots, x_n) is constant given the input, we can use the following classification rule:

P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)

\Downarrow

\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),

and we can use Maximum A Posteriori (MAP) estimation to estimate P(y) and P(x_i \mid y); the former is then the relative frequency of class y in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(x_i \mid y).

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. (For theoretical reasons why naive Bayes works well, and on which types of data it does, see the references below.)

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

On the flip side, although naive Bayes is known as a decent classifier, it is known to be a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.

**Gaussian Naive Bayes**

[GaussianNB](http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB) implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

P(x_i \mid y) &= \frac{1}{\sqrt{2\pi\sigma^2_y}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma^2_y}\right)

The parameters \sigma_y and \mu_y are estimated using maximum likelihood.

## Multinomial Naive Bayes

[**MultinomialNB**](http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB) implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors \theta_y = (\theta_{y1},\ldots,\theta_{yn}) for each class y, where n is the number of features (in text classification, the size of the vocabulary) and \theta_{yi} is the probability P(x_i \mid y) of feature i appearing in a sample belonging to class y.

The parameters \theta_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

\hat{\theta}_{yi} = \frac{ N_{yi} + \alpha}{N_y + \alpha n}

where N_{yi} = \sum_{x \in T} x_i is the number of times feature i appears in a sample of class y in the training set T, and N_{y} = \sum_{i=1}^{|T|} N_{yi} is the total count of all features for class y.

The smoothing priors \alpha \ge 0 accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting \alpha = 1 is called Laplace smoothing, while \alpha < 1 is called Lidstone smoothing.

## Bernoulli Naive Bayes

[BernoulliNB](http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB) implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors; if handed any other kind of data, a BernoulliNB instance may binarize its input (depending on the binarize parameter).

The decision rule for Bernoulli naive Bayes is based on

P(x_i \mid y) = P(i \mid y) x_i + (1 - P(i \mid y)) (1 - x_i)

which differs from multinomial NB’s rule in that it explicitly penalizes the non-occurrence of a feature i that is an indicator for class y, where the multinomial variant would simply ignore a non-occurring feature.

In the case of text classification, word occurrence vectors (rather than word count vectors) may be used to train and use this classifier. BernoulliNB might perform better on some datasets, especially those with shorter documents. It is advisable to evaluate both models, if time permits.