Naive Bayes

Link : https://www.javatpoint.com/machine-learning-naive-bayes-classifier
Link : https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html
Link:https://www.kdnuggets.com/2019/11/probability-learning-naive-bayes.html

Naive Bayes is based on Bayes Theorem

Naive Bayes is a simplification of Bayes' theorem which is used as a classification algorithm for binary of multi-class problems. It is called naive because it makes a very important but somehow unreal assumption: that all the features of the data points are independent of each other. By doing this it largely simplifies the calculations needed for Bayes' classification, while maintaining pretty decent results. These kinds of algorithms are often used as a baseline for classification problems.

Why is it called Naïve Bayes?
The Naive Bayesian classifier is based on Bayes' theorem with the independence assumptions between predictors

Assumptions made by Naïve Bayes

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

independent

contribution to the outcome.

The independence assumption is never correct but often works well in practice. **Hence the name 'Naïve'.**

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

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.

Naive Bayes is **called naive** because it assumes that each input variable is independent. This is a strong assumption and unrealistic for real data; however, the technique is very effective on a large range of complex problems.

BAYES THEOREM

1.Conditional Probability -
P(A/given B)=P(A intersection B)/P(B)
2.Independent Events - two events do not depend on each other
3.Dependent Events - two events depend on each other

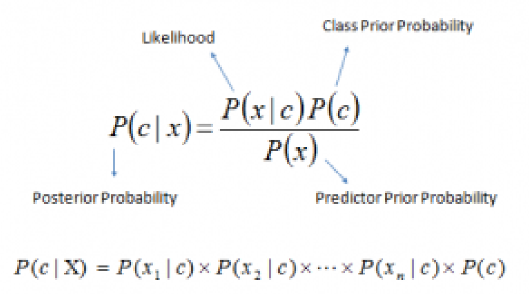
Bayes Theorem

P(A/B)=P(B/A)*P(A)/P(B)

P(A/B) ->Proster Probability
P(A)->Likehood
P(B)->Marginal

Example

Dataset :
X={x1,x2,x3 …..} {y}
{x1,x2,x3 …..} ->features
y->output



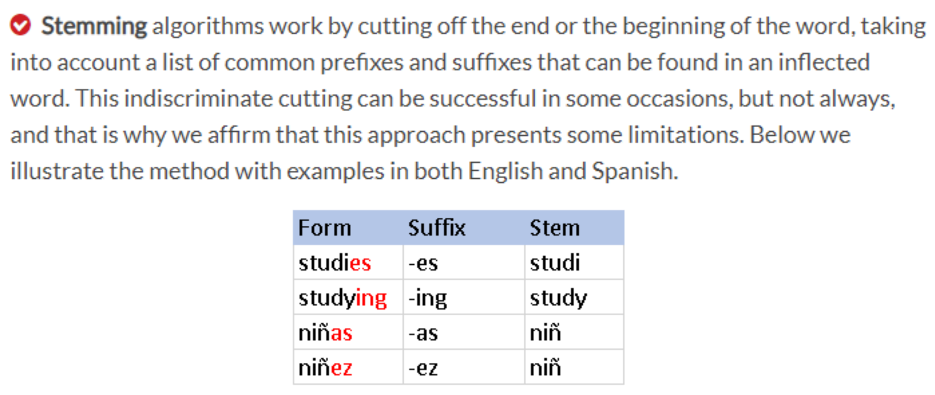
$$P(c\mid x)=\frac{P(x\mid c)P(c)}{P(x)}$$

$$P(c\mid X) = P(x_1\mid c)\times P(x_2\mid c)\times\cdots\times P(x_n\mid c)\times P(c)$$

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*.

Stopwords - > do not add much value to description
Stemming ->



✓ **Stemming** algorithms work by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word. This indiscriminate cutting can be successful in some occasions, but not always, and that is why we affirm that this approach presents some limitations. Below we illustrate the method with examples in both English and Spanish.

Form	Suffix	Stem
studies	-es	studi
studying	-ing	study
niñas	-as	niñ
niñez	-ez	niñ

Lemmitization -> Lemmatization and stemming both does the same work but lemmatization gives a meaningful answers which is understandable by humans

Countvectorizer - >
TfidfVectorizer ->

TF- Term Frequency Formulae : No.of rep words in a sentence / No.of words in a sentence
IDF -inverse document frequency Formulae : log(No.of sentences/No.of sentences containing words)

We multiply TF and IDF to convert sentences into vectors

TF-IDF is better than Bag of words

Sentence embedding techniques represent entire sentences and their semantic information as vectors. This helps the machine in understanding the context, intention, and other nuances in the entire text.

Data preprocessing is a **data** mining technique that involves transforming raw **data** into an understandable format.

Within the field of machine learning, there are two main types of tasks: supervised, and unsupervised. The main difference between the two types is that supervised learning is done using a **ground truth**, or in other words, we have prior knowledge of what the output values for our samples should be. Therefore, the goal of supervised learning is to learn a function that, given a sample of data and desired outputs, best approximates the relationship between input and output observable in the data. Unsupervised learning, on the other hand, does not have labeled outputs, so its goal is to infer the natural structure present within a set of data points.

Supervised learning is typically done in the context of classification, when we want to map input to output labels, or regression, when we want to map input to a continuous output. Common algorithms in supervised learning include logistic regression, naive bayes, support vector machines, artificial neural networks, and random forests. In both regression and classification, the goal is to find specific relationships or structure in the input data that allow us to effectively produce correct output data. Note that “correct” output is determined entirely from the training data, so while we do have a ground truth that our model will assume is true, it is not to say that data labels are always correct in real-world situations. Noisy, or incorrect, data labels will clearly reduce the effectiveness of your model.
When conducting supervised learning, the main considerations are model complexity, and the bias-variance tradeoff. Note that both of these are interrelated.

Steps to implement:

- Data Pre-processing step
- Fitting Naive Bayes to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)

A **Confusion matrix** is an N x N **matrix** used for evaluating the performance of a classification model, where N is the number of target classes. The **matrix** compares the actual target values with those predicted by the **machine learning** model

- Visualizing the test set result.