Naïve Bayes Classifier

# What is Classifier?

A Classifier is a Machine learning model that is used to discriminate different objects based on certain features.

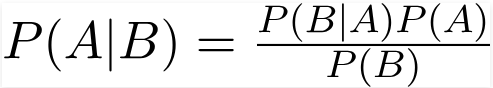
# Principle of Naïve Bayes Classifier

A Naïve Bayes Classifier is a probabilistic machine learning model that’s used for classification tasks. The classifier works based on **“Bayes Theorem”.**

# Bayes Theorem

For a given Hypothesis (A) and Evidence (B), Bayes Theorem states,

That the relationship between the probability of hypothesis and the probability of hypothesis after getting the evidence.



P(A|B) = Conditional Probability of A given B, P(B|A) = Conditional Probability of B for given A

P(A), P(B) = Probability of A, B

# Assumptions

* **All the predictors are independent**
* **All the predictors have an equal effect on the outcome.**

# Real time Applications

* **Naive Bayes is an eager supervised learning classifier and it is sure fast.**
* **Text Classification, Spam detection and Sentiment Analysis**

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**.

# Types of Naïve Bayes Classifier

1. **Multinomial Naïve Bayes**

This is mostly used **for document classification problems**, whether a document belongs to the category of sports, politics, technology etc.

“The features/predictors used by the classifier are the frequency of the words present in the document.” For example “count how often word occurs in the document”.

1. **Bernoulli Naïve Bayes**

This is similar to Multinomial Naïve Bayes **but the predictors are Boolean variables.**

The parameters that we use to predict target variable take up only values ‘yes’ or ‘no’, for example if a word occurs in the text or not.

1. **Gaussian Naïve Bayes**

When the **predictors take up a continuous value** and are not discrete, we **assume that features follows a normal distribution.**

Normal distribution is also called Gaussian distribution, when a plotted curve gives a bell shaped curve and it is symmetric about the mean of the feature values.

# Algorithm

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



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



Just to clear, an example of a feature vector and corresponding class variable can be:

X = (Rainy, Hot, High, False)

y = No

So basically, 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”.

# Coding Part:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

plt.rcParams["figure.figsize"]=(10,10)

**from sklearn.naive\_bayes import GaussianNB**

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_iris

from sklearn.metrics import accuracy\_score

data=load\_iris()

X= data.data

Y= data.target

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,random\_state=1, test\_size=0.4)

**model = GaussianNB()**

model.fit(X\_train,Y\_train)

Y\_pred = model.predict(X\_test)

print("Accuracy of the model: ", (accuracy\_score(Y\_test,Y\_pred))\*100)

Output: Accuracy of the model: 95.0