**Project Report on Santander Customer Transaction Prediction**

Data Preprocessing

It is a data mining technique that transforms raw data into an understandable format. Raw data(real world data) is always incomplete and that data cannot be sent through a model. That would cause certain errors. That is why we need to preprocess data before sending through a model.

Steps in Data Preprocessing

Here are the steps I have followed;  
1. Import libraries  
2. Read data  
3. Checking for missing values  
5. Standardize the data  
6. Feature Selection  
7. Data splitting

8. Model Building

9. Model Evaluation

10. predict target values in test data

1. Import Data

As main libraries, I am using Pandas, Numpy and time;  
**Pandas**: Use for data manipulation and data analysis.  
**Numpy**: a fundamental package for scientific computing with Python.

As for the visualization I am using **Matplotlib**and Seaborn.  
For the data preprocessing techniques and algorithms, Iused**Scikit-learn**libraries and many other.

# main libraries  
import pandas as pd  
import numpy as np  
import time# visual libraries  
from matplotlib import pyplot as plt  
import seaborn as sns  
from mpl\_toolkits.mplot3d import Axes3D   
plt.style.use('ggplot')# sklearn libraries  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import normalize  
from sklearn.metrics import confusion\_matrix,accuracy\_score,precision\_score,recall\_score,f1\_score,matthews\_corrcoef,classification\_report,roc\_curve  
from sklearn.externals import joblib  
from sklearn.preprocessing import StandardScaler  
from sklearn.decomposition import PCA

2. Read Data

*# Read the data in the CSV file using pandas*  
df = pd.read\_csv(‘train.csv')  
df =   
> (200000, 202)

3. Checking for missing values

df.isnull().any().sum()  
> 0

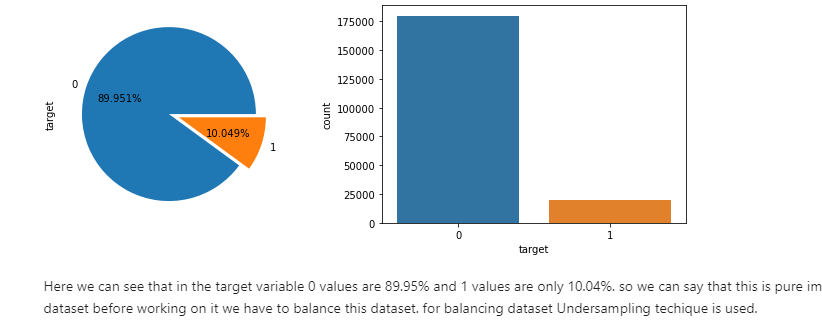
Since there are no missing values found in the dataset, I didn’t use any missing values handling techniques.

Let’s look at the data

So the dataset is labeled as 0s and 1s.

* 0 = non default
* 1 = default

All = df.shape[0]  
default = df[df['Class'] == 1]  
non default = df[df['Class'] == 0]



As we can see here is that only 10.049% of 1 are present that’s why our data is imbalanced so first we have to balance our data using data balancing techniques such as undersampling.

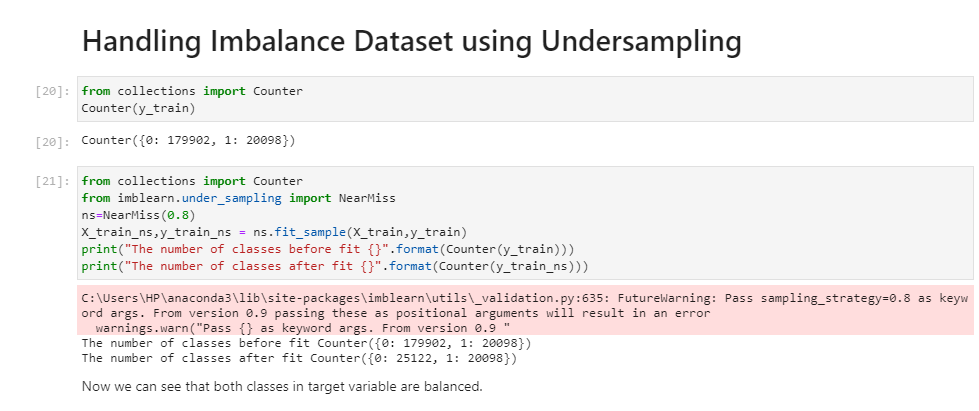
**Undersampling Imbalanced Datasets**

Random undersampling involves randomly selecting examples from the majority class to delete from the training dataset.

This has the effect of reducing the number of examples in the majority class in the transformed version of the training dataset. This process can be repeated until the desired class distribution is achieved, such as an equal number of examples for each class.

This approach may be more suitable for those datasets where there is a class imbalance although a sufficient number of examples in the minority class, such a useful model can be fit.

A limitation of undersampling is that examples from the majority class are deleted that may be useful, important, or perhaps critical to fitting a robust decision boundary. Given that examples are deleted randomly, there is no way to detect or preserve “*good*” or more information-rich examples from the majority class.



**Normalization**

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Here’s the formula for normalization:

[Normalization equation](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/Norm_eq.gif)

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

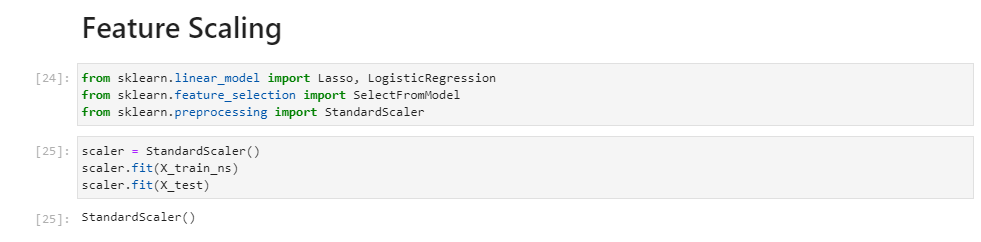
**Standardization**

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation**.**

Here’s the formula for standardization:

[Standardization equation](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/Stand_eq.gif)

Feature scaling: Mu is the mean of the feature values and Feature scaling: Sigma is the standard deviation of the feature values. Note that in this case, the values are not restricted to a particular range.



Feature Selection

# Lasso Regression

Using Linear Regression with L1 regularization is called Lasso Regularization. There are lots of other methods of regularization as well like Ridge regression or you can create your own custom regularization method which suits your training method and outdoes the effect of outliers. Now coming to feature selection part using Lasso Regression.

Let us take another look at the Lasso Regression solution. We have different features or variables in our data which we denote by x1, x2, …, xn. And our hypothesis function for Lasso Regression(or Linear Regression) looks like:

Image for post

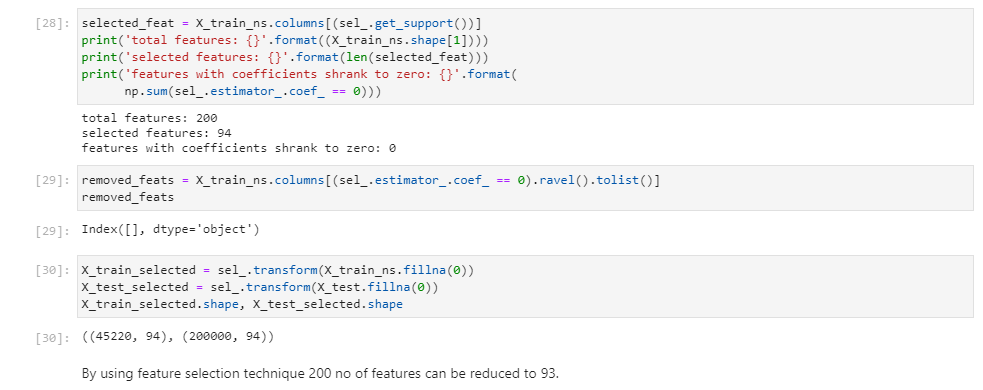
Linear Regression

The learnable or trainable parameters in our models are θ0, θ1, …, θn.

After training some values will be assigned to θ parameters. If we observe the solution the larger the values of θ the larger effect it will have on the solution and vice versa.

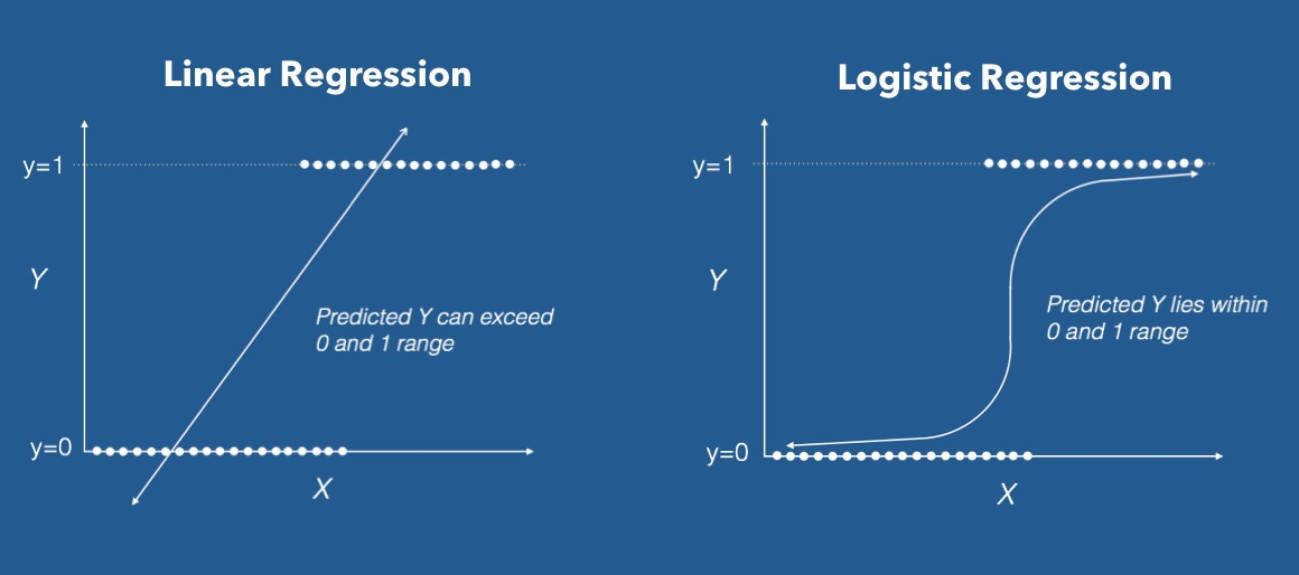
Thus with this intuition, we can decide upon some threshold value and we can keep all those variables for which the corresponding θ values are larger than the threshold value and discard the others. The larger the threshold values the smaller the number of parameters we will keep and vice versa.

The selected features can then be used to train your models. Of course, there are other things as well to improve the dataset like normalizing the values and handling the missing data before proceeding to train the model and there are other ways of feature selection. But I liked this idea of using regularization for feature selection. This is a nice way of reducing dimensionality by removing not so important features.

# Logistic Regression

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.



Linear Regression VS Logistic Regression Graph| Image: Data Camp

We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function, this cost function can be defined as the ‘**Sigmoid function**’ or also known as the ‘logistic function’ instead of a linear function.

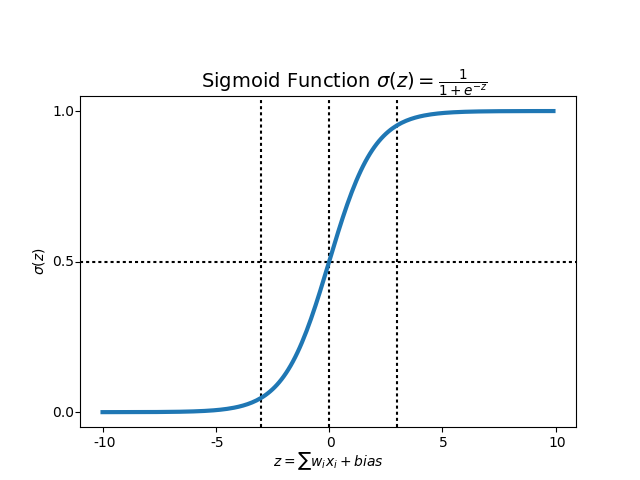
The hypothesis of logistic regression tends it to limit the cost function between 0 and 1. Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.

Image for post

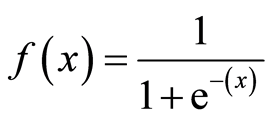
Logistic regression hypothesis expectation

## What is the Sigmoid Function?

In order to map predicted values to probabilities, we use the Sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.



Sigmoid Function Graph



Formula of a sigmoid function | Image: Analytics India Magazine

# ****Hypothesis Representation****

When using linear regression we used a formula of the hypothesis i.e.

*hΘ(x) = β₀ + β₁X*

For logistic regression we are going to modify it a little bit i.e.

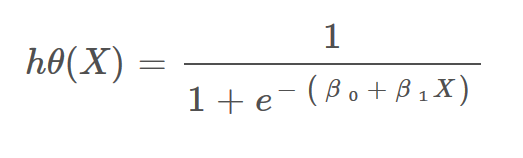
*σ(Z) = σ(β₀ + β₁X)*

We have expected that our hypothesis will give values between 0 and 1.

Z = β₀ + β₁X

hΘ(x) = sigmoid(Z)

i.e. hΘ(x) = 1/(1 + e^-(β₀ + β₁X)

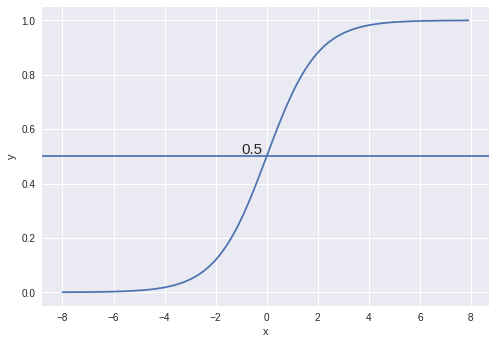


The Hypothesis of logistic regression

## Decision Boundary

We expect our classifier to give us a set of outputs or classes based on probability when we pass the inputs through a prediction function and returns a probability score between 0 and 1.

For Example, We have 2 classes, let’s take them like cats and dogs(1 — dog , 0 — cats). We basically decide with a threshold value above which we classify values into Class 1 and of the value goes below the threshold then we classify it in Class 2.



Example

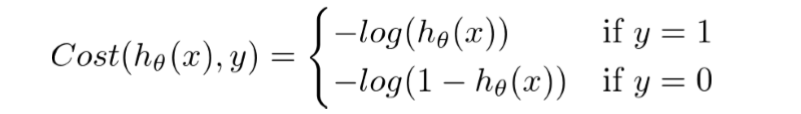
As shown in the above graph we have chosen the threshold as 0.5, if the prediction function returned a value of 0.7 then we would classify this observation as Class 1. If our prediction returned a value of 0.2 then we would classify the observation as Class 2.

# Cost Function

For logistic regression, the Cost function is defined as:

*−log(*hθ*(*x*)) if y = 1*

*−log(1−*hθ*(*x*)) if y = 0*



Cost function of Logistic Regression

**Decision Tree Classifier**

A tree has many analogies in real life, and turns out that it has influenced a wide area of **machine learning**, covering both **classification and regression**. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, its also widely used in machine learning.

How can an algorithm be represented as a tree?

For this let’s consider a very basic example that uses titanic data set for predicting whether a passenger will survive or not. Below model uses 3 features/attributes/columns from the data set, namely sex, age and sibsp (number of spouses or children along).

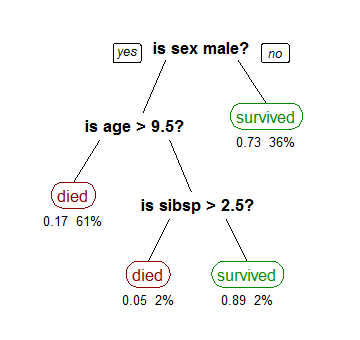


Image taken from wikipedia

*A decision tree is drawn upside down with its root at the top.* In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn’t split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.

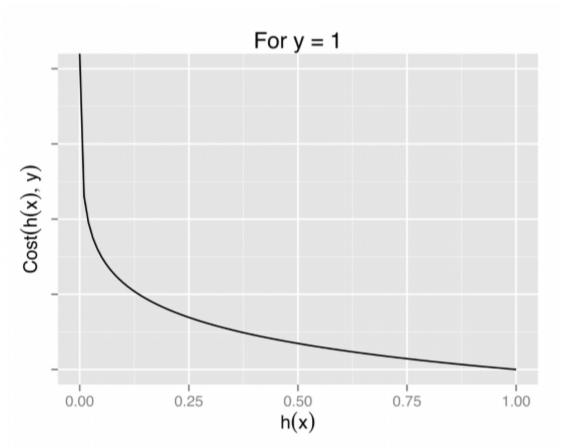
Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can’t ignore the simplicity of this algorithm. The feature importance isclear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

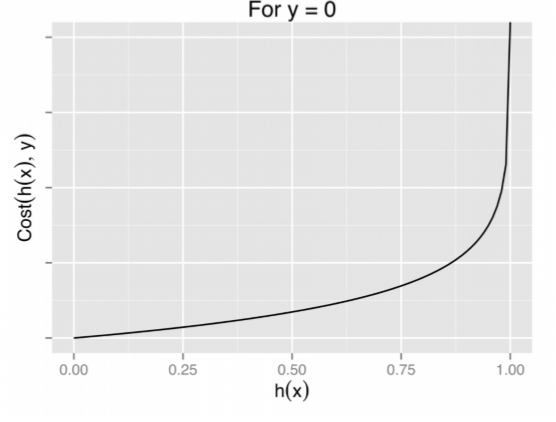
So, what is actually going on in the background? Growing a tree involves deciding on which features to choose and whatconditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, we will need to trim it down for it to look beautiful. Lets start with a common technique used for splitting.

**Recursive Binary Splitting**

*In this procedure all the features are considered and different split points are tried and tested using a cost function. The split with the best cost (or lowest cost) is selected.*

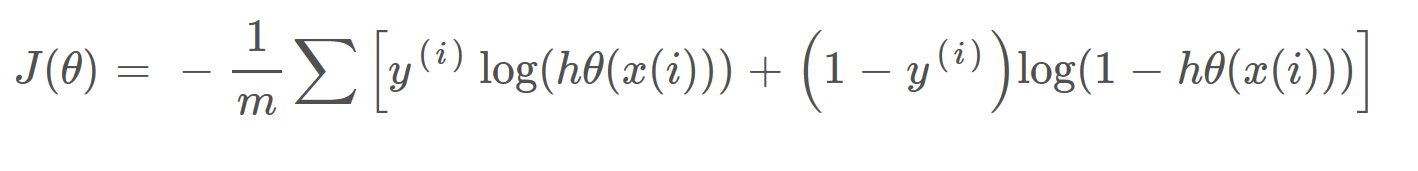
Consider the earlier example of tree learned from titanic dataset. In the first split or the root, all attributes/features are considered and the training data is divided into groups based on this split. We have 3 features, so will have 3 candidate splits. Now we will *calculate how much*accuracy *each split will cost us, using a function*. *The split that costs least is chosen*, which in our example is sex of the passenger. This *algorithm is recursive in**nature* as the groups formed can be sub-divided using same strategy. Due to this procedure, this algorithm is also known as the greedy algorithm, as we have an excessive desire of lowering the cost. This makes the root node as best predictor/classifier**.**





Graph of logistic regression

The above two functions can be compressed into a single function i.e.

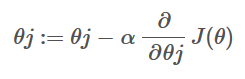


Above functions compressed into one cost function

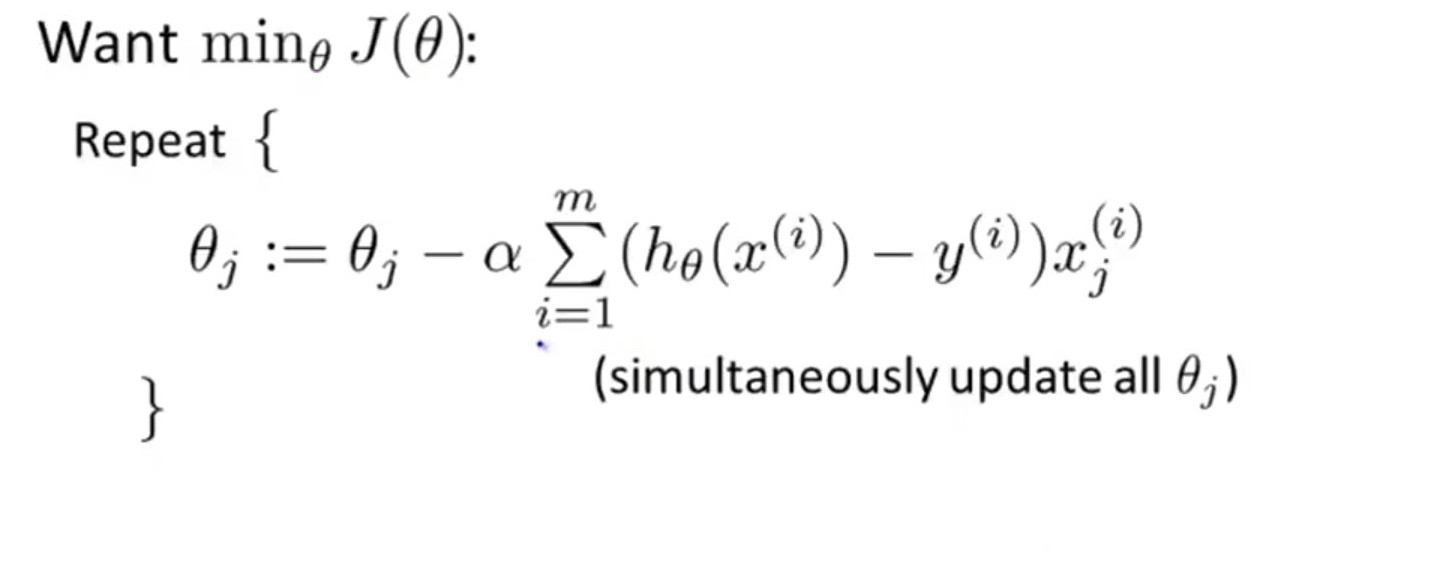
# Gradient Descent

Now the question arises, how do we reduce the cost value. Well, this can be done by using **Gradient Descent.**The main goal of Gradient descent is to **minimize the cost value.** i.e. min J(**θ**).

Now to minimize our cost function we need to run the gradient descent function on each parameter i.e.

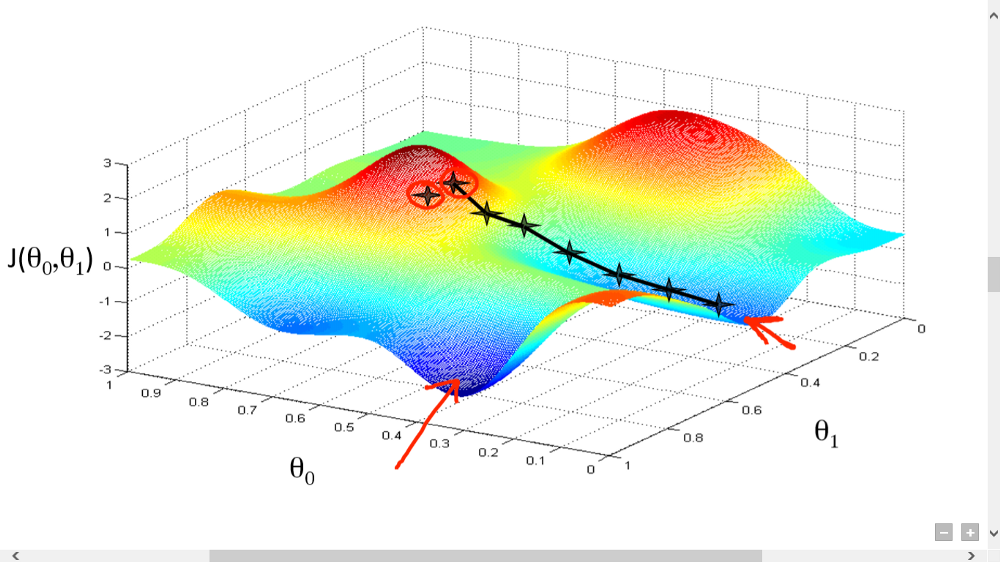


Objective: To minimize the cost function we have to run the gradient descent function on each parameter



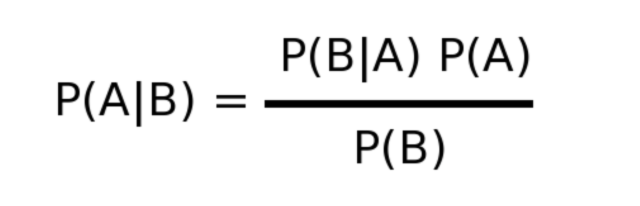
Gradient Descent Simplified | Image: Andrew Ng Course

Gradient descent has an analogy in which we have to imagine ourselves at the top of a mountain valley and left stranded and blindfolded, our objective is to reach the bottom of the hill. Feeling the slope of the terrain around you is what everyone would do. Well, this action is analogous to calculating the gradient descent, and taking a step is analogous to one iteration of the update to the parameters.



# The Naive Bayes Theory

The Naive Bayes Theorey in the most cases can be reduced to a formula:

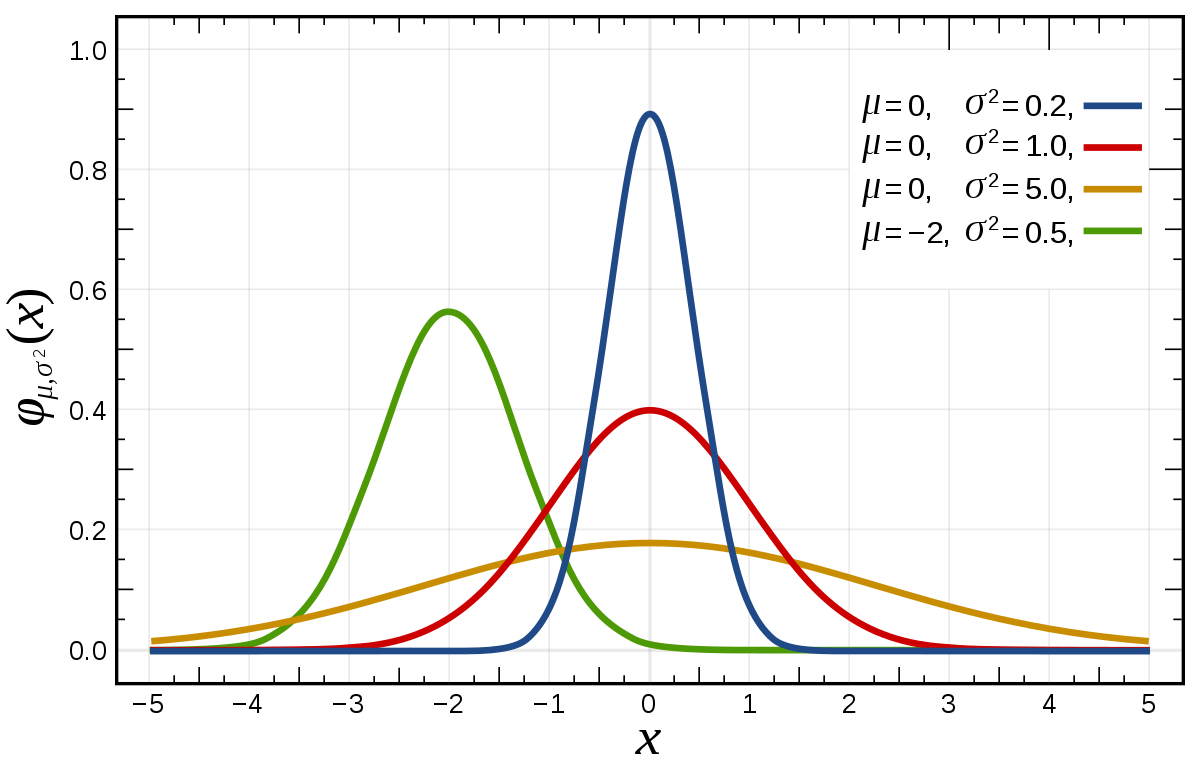


The Naive Bayes formula [source — [https://miro.medium.com/max/640/1\*7lg\_uLm8\_1fYGjxPbTrQFQ.png](https://miro.medium.com/max/640/1*7lg_uLm8_1fYGjxPbTrQFQ.png)]

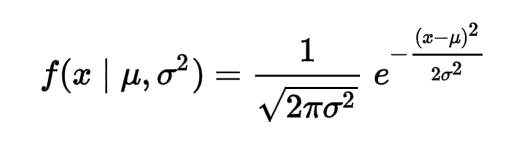
This formula means that the probability of happening of the event A knowing that event B happened already.

# What is a distribution?

Distribution, basically show how values are dispersed in series, and how frequently they appear in this series. Here is an example:



How we can see in the plot above, the Gaussian or Normal Distribution depends on 2 parameters of a series — The mean and the standard Deviation. Knowing these 2 parameters of the series we can find it’s distribution function. It has the next form:



But, why do we need this function? Very simple, the majority of data in the world is represented as continuous values, but guess what, you can’t calculate the probability of the value X to get the value v. It would be 0. Why? Technically when you divide something by infinity you get what? correct — zero.

So, how we can solve this problem? Of course, using the Gaussian Distribution Function, illustrated above. Imputing instead of x their value from a series, the mean value of the series and its standard deviation you can find out the probability that the value x will occur. Voila.

**Model Evaluation**

Confusion matrix

Well, it is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.



It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC Curve.

Let’s understand TP, FP, FN, TN in terms of pregnancy analogy.

**True Positive:**

Interpretation: we predicted positive and it’s true.

we predicted that a woman is pregnant and she actually is.

**True Negative:**

Interpretation: we predicted negative and it’s true.

we predicted that a man is not pregnant and he actually is not.

**False Positive: (Type 1 Error)**

Interpretation: we predicted positive and it’s false.

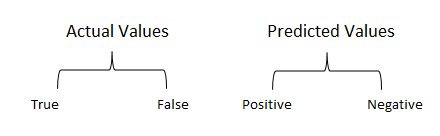
we predicted that a man is pregnant but he actually is not.

**False Negative: (Type 2 Error)**

Interpretation: we predicted negative and it’s false.

we predicted that a woman is not pregnant but she actually is.

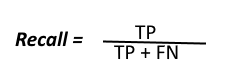
Just Remember, We describe predicted values as Positive and Negative and actual values as True and False.



**How to Calculate Confusion Matrix for a 2-class classification problem?**

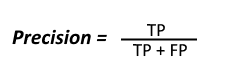
Let’s understand confusion matrix through math.

**Recall**



Out of all the positive classes, how much we predicted correctly. It should be high as possible.

**Precision**

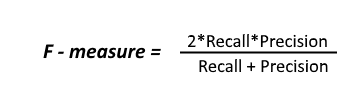


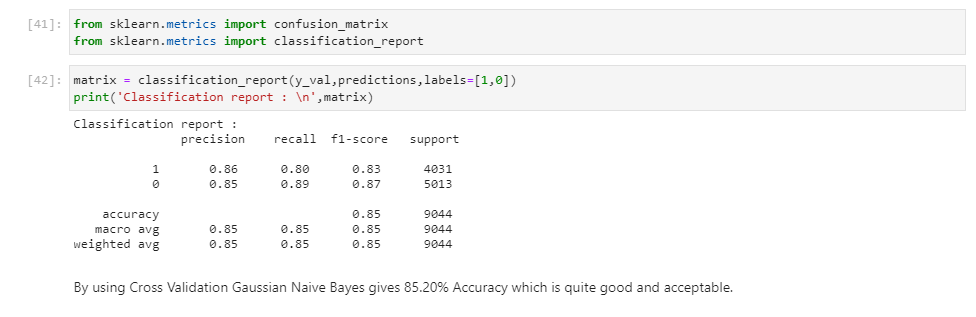
Out of all the positive classes we have predicted correctly, how many are actually positive.

and **Accuracy** will be

Out of all the classes, how much we predicted correctly, It should be high as possible.

**F-measure**



****

**Conclusion**

First I start with data preprocessing as missing value then tried standardizing the data to get a better accuracy. After Feature scaling I did feature selection using Lasso Regression in which I kept 98 features out of 200 features. After selecting feature I develop various models such as Logistic Regression, Random Forest, Decision tree, Svm, Gaussian Nave Bayes. From above all models GaussianNB gives best accuracy and result then I did cross Validation and perform matrix evaluation.

Finally I predict target values using my model in test dataset.

**Reference**

<https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>

<https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>

<https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148#:~:text=Logistic%20Regression%20is%20a%20Machine,on%20the%20concept%20of%20probability.&text=The%20hypothesis%20of%20logistic%20regression,function%20between%200%20and%201%20>.

<https://medium.com/@23.sargam/lasso-regression-for-feature-selection-8ac2287e25fa>

<https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/>