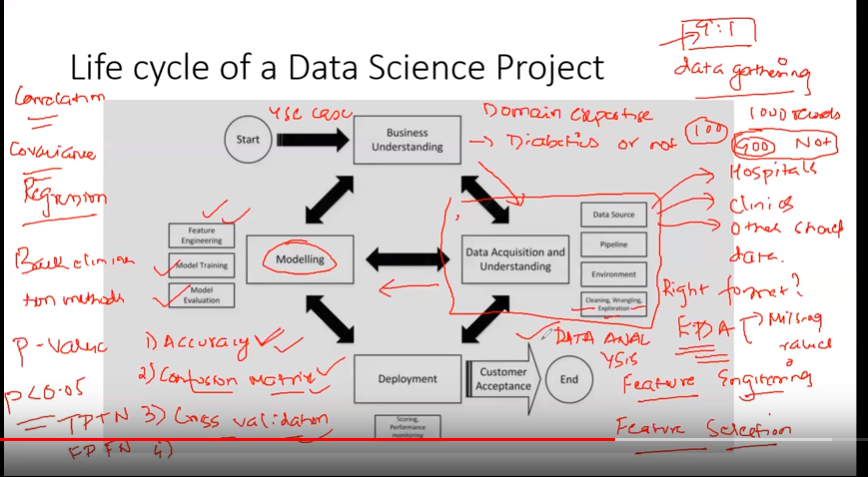
# **1)Complete Life Cycle of a Data Science Project**:

1. **Collect data**
2. **Data preprocessing** (to get data in the right format)
   1. : *Feature engineering (EDA, Data cleaning, Data Wrangling)*
   2. *: Feature selection (avoid features which are not needed and won’t help in model accuracy)*
3. **Check balance vs imbalance Data**?
4. **Data Modeling** (Use below techniques to choose the right algorithm)
   1. *Accuracy*
   2. *Confusion Metrics*
   3. *Cross-Validation (best)*
5. **Model Deployment:** We deploy the model in the form of API(web services) as our model is like a black box (take input and gives output) using Flask or AWSS.
6. If Everything goes fine then our model is good, otherwise, we will start it from starts(we can collect more data and all). So it is a continuous process - to follow the same cycle again and again.

****

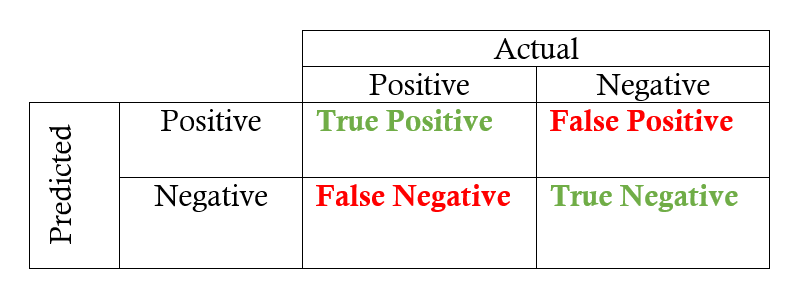
**----------------------------------------------------------------------------------------------------------------**

# **2)Confusion matrix, Precision, Recall**

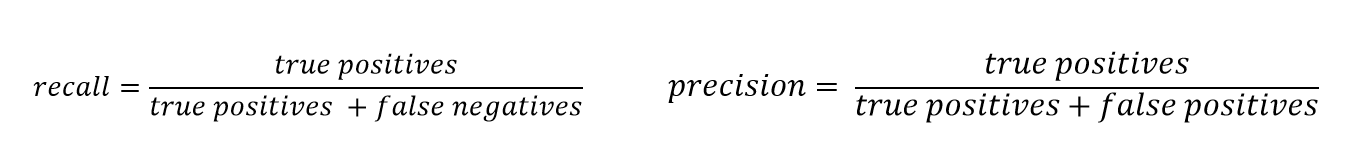
* **TP, TN, FP, FN**
* **TPR, TNR, FPR, FNR**

**Confusion matrix:** will help you to see the accuracy

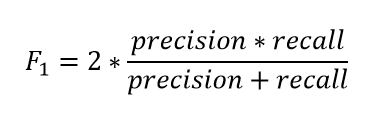
* **Confusion Matrix**: a table showing correct predictions and types of incorrect predictions.



* **Precision:** the number of true positives divided by all positive predictions. Precision is also called Positive Predictive Value. It is a measure of a classifier’s exactness. Low precision indicates a high number of false positives.
* **Recall:** the number of true positives divided by the number of positive values in the test data. The recall is also called Sensitivity or the True Positive Rate. It is a measure of a classifier’s completeness. Low recall indicates a high number of false negatives.



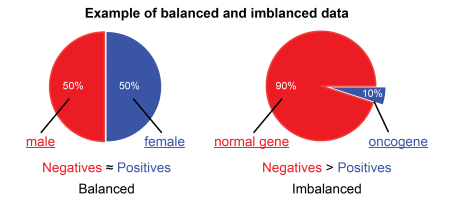
* **F1-Score**: the weighted average of precision and recall.



**----------------------------------------------------------------------------------------------------------------**

# **3)Balanced vs Imbalanced Dataset and how to handle Imbalanced Dataset**

Ref: ([**https://medium.com/analytics-vidhya/what-is-balance-and-imbalance-dataset-89e8d7f46bc5**](https://medium.com/analytics-vidhya/what-is-balance-and-imbalance-dataset-89e8d7f46bc5)**)**

****

## **Problem with an Imbalanced Datasets**

Let’s say you are working in a leading tech company and company is giving you a task to train a model on detecting fraud detection. But here’s the catch. The fraud transaction is relatively rare;

So you start to train your model and get over 95% accuracy.

You feel good and present your model in front of company CEO and Share Holders.

When they give inputs to your model so your model is predicting “Not a Fraud Transaction” every time.

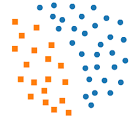
This is clearly a problem because many machine learning algorithms are designed to maximize overall accuracy.

Now, what happens?? You get 95% accuracy but your model in predicting wrong every time??

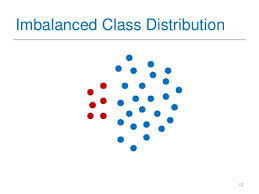
Let’s find out why?

## **What are Balanced and Imbalanced Datasets?**

**Balanced Dataset:** — Let’s take a simple example if in our data set we have positive values which are approximately same as negative values. Then we can say our dataset in balance



**Imbalanced Dataset:** — If there is a very high difference between positive values and negative values. Then we can say our dataset in Imbalance Dataset.



## **Techniques to Convert Imbalanced Dataset into Balanced Dataset**

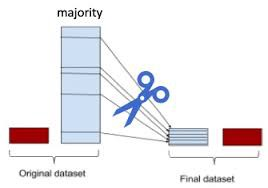
Imbalanced data is not always a bad thing, and in real data sets, there is always some degree of imbalance.

That said, there should not be any big impact on your model performance if the level of imbalance is relatively low.

Now, let’s cover a few techniques to solve the class imbalance problem.

1. **Under-sampling (Down Sampling):**

Unlike oversampling, this technique balances the imbalance dataset by reducing the size of the class which is in abundance. But Since it is removing observations from the original data set, it might discard useful information.

****

**Advantages**

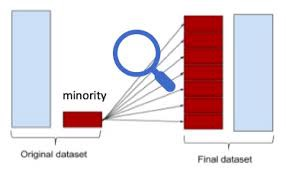
* Run-time can be improved by decreasing the amount of training dataset.
* Helps in solving the memory problems

**Disadvantages**

* Losing some critical information

**2)Over-sampling (Up Sampling):**

* This technique is used to modify the unequal data classes to create balanced datasets. When the quantity of data is insufficient, the oversampling method tries to balance by incrementing the size of rare samples.
* Over-sampling increases the number of minority class members in the training set. The advantage of over-sampling is that no information from the original training set is lost, as all observations from the minority and majority classes are kept. On the other hand, it is prone to overfitting.



**Advantages**

* No loss of information
* Mitigate overfitting caused by oversampling.

**Disadvantages**

* Overfitting

**3) Use Tree-Based Algorithms:**

The final tactic we'll consider is using **tree-based algorithms**. Decision trees often perform well on imbalanced datasets because their hierarchical structure allows them to learn signals from both classes.

In modern applied machine learning, tree **ensembles** (Random Forests, Gradient Boosted Trees, etc.) almost always outperform singular decision trees.

**Note: Well, tree ensembles have become very popular because they perform extremely well on many real-world problems**

**----------------------------------------------------------------------------------------------------------------**

**4)Select Best Model using cross-validation in Python**

**Jupyter Notebook:**(**model\_selections\_K\_Fold\_cross-validation(ModelBoost).ipynb)**

**----------------------------------------------------------------------------------------------------------------**

**5)Principal Component Analysis:**

**Another google doc for this**: (Create Pipeline, PCA, MinMaxScaler, StandardScaler using Sklearn: <https://docs.google.com/document/d/149h8ByEAMhLzGV56KcegJv23MJj1D9Gl-65x0Y_SOoc/edit>)

**Note: As the number of dimensions increases, it is a curse for our model accuracy because:**

**Your accuracy will be impacted based upon your dimensions (more dimension, less accuracy)**

**Jupyter Notebook: (Principal Component Analysis(code).ipynb)**

* Here in the **Jupyter** notebook, the total number of dimensions we have is 30.
* We need to reduce this to 2 dimensions.
* Here in 2 dimensions, we will be having after PCA:

1) WDBC-Malignant

2) WDBC-Benign

**----------------------------------------------------------------------------------------------------------------**

**6) TPR, FPR, FNR, TNR, Confusion Metric**

**----------------------------------------------------------------------------------------------------------------**

**7) Precision, Recall and F1-Score:** These we usually use in information retrieval.

**----------------------------------------------------------------------------------------------------------------**

**8) GridSearchCV-** Select the best hyperparameter for any classification Model**(in order to increase the accuracy of the model)**

**Jupyter Notebook: (GridSearchCV(Code).ipynb)**

**----------------------------------------------------------------------------------------------------------------**

**9) RandomizedSearchCV -** Select the best hyperparameter for any classification:

* **GridsearchCV is usually very slow in comparison to this one.**
* With small data sets and lots of resources, Grid Search will produce accurate results. However, with large data sets, the high dimensions will greatly slow down computation time and be very costly.
* In this instance, it is advised to use Randomized Search since the number of iterations is explicitly defined by the data scientist.

**Juputer:** **(RandomizedSearchCV(Code).ipynb)**

**----------------------------------------------------------------------------------------------------------------**

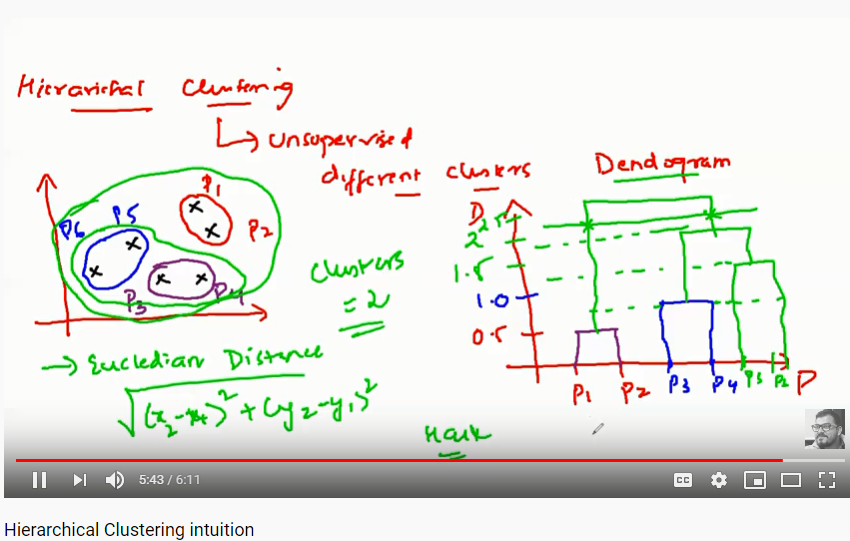
**10) KNN:** Check jupyter for it once and mention that here.

**----------------------------------------------------------------------------------------------------------------**

**11) Hierarchical Clustering Intuition:**

(Unsupervised machine learning algo)

**Jupyter Notebook:** **(Hierarchical\_Clustering.ipynb)**

****

**There are two types of hierarchical clustering:**

1. **Agglomerative** and 2) **Divisive**. In the former, data points are clustered using a bottom-up approach starting with individual data points, while in the latter top-down approach is followed where all the data points are treated as one big cluster and the clustering process involves dividing the one big cluster into several small clusters.

**Agglomerative Clustering:**

#### **Steps to Perform Hierarchical Clustering**

Following are the steps involved in agglomerative clustering:

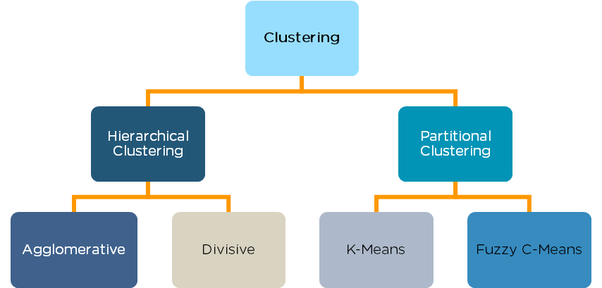
1. At the start, treat each data point as one cluster. Therefore, the number of clusters at the start will be K, while K is an integer representing the number of data points.
2. Form a cluster by joining the two closest data points resulting in K-1 clusters.
3. Form more clusters by joining the two closest clusters resulting in K-2 clusters.
4. Repeat the above three steps until one big cluster is formed.
5. Once single cluster is formed, [dendrograms](https://en.wikipedia.org/wiki/Dendrogram) are used to divide into multiple clusters depending upon the problem. We will study the concept of dendrogram in detail in an upcoming section.

There are different ways to find distance between the clusters. The distance itself can be Euclidean or Manhattan distance. Following are some of the options to measure distance between two clusters:

1. Measure the distance between the closes points of two clusters.
2. Measure the distance between the farthest points of two clusters.
3. Measure the distance between the centroids of two clusters.
4. Measure the distance between all possible combinations of points between the two clusters and take the mean.

#### **Role of Dendrograms for Hierarchical Clustering**

**Very good reference: (**[**https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/**](https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/)**)**

****

**Difference between K-Means and Hierarchical Clustering:**

1. In K means clustering we have to define the number of clusters to be created beforehand, Which is sometimes difficult to say. Whereas in Hierarchical clustering data is automatically formed into a tree shape form (dendrogram) and we can chose which trees are significant.

### **When should I go for K-Means Clustering and when for Hierarchical Clustering ?**

Well, Answer is pretty simple, if your data is small then go for Hierarchical Clustering and if it is large then go for K-Means Clustering.