PROJECT REPORT –

SALARY PREDICTION FOR GRADUATES

# Prediction of Salary for Graduates who have taken AMCAT exam

Under the guidance of –

**Mr. Shashank Shirude**

**Sr. AI Engineer**

**bizAmica Software Pvt. Ltd.**

**TABLE OF CONTENTS**

1. **Introduction 3**
2. **Data Description 4**
3. **Exploratory Data Analysis 6**
4. **Data Cleaning. 15**

**INTRODUCTION**

**AMCAT Exam**

AMCAT or Aspiring Minds Computer Adaptive Test evaluates candidates on the basis of their core skills which includes reasoning skills, quantitative aptitude, English and technical skills. The technical module is different for each candidate and depends on the course or the subject they opt for. It is a standardized test of job skills. The test includes cognitive, domain and personality assessments.

**AIM OF THE PROJECT**

Every year lakhs of engineering graduates apply for jobs. AMCAT is an exam which many companies take into account while recruiting. So this project is aimed at predicting annual salaries of engineering graduates who take the AMCAT exam.

The main focus is to interpret the factors determining salaries. This could help students analyse the recruiting trends of companies and focus on the valued skills.

**PROBLEM STATEMENT**

* Given a new student profile, can we predict whether he will get a good salary package or average one using his historic data.
* Can we understand what factors in the labor market determine one’s salary? Is it just one’s skills or there are other factors which influence the return in the labor market? What signals and biases enter the labor market?

**DATA DESCRIPTION**

**DATA SET**

The entire data is collected from Aspiring Minds’ Employment Outcomes 2015. The dataset contains various information about a set of engineering candidates and their employment outcomes. For every candidate, the data contains both the profile information along with their employment outcome information. Candidate Profile Information includes:

Scores on Aspiring Minds’ AMCAT – a standardized test of job skills. The test includes cognitive, domain and personality assessments

* Personal information like gender, date of birth, etc.
* Pre-university information like high school grades, high school location
* University information like GPA, college major, college reputation proxy.
* Demographic information like location of college, candidates’ permanent location

Employment Outcome Information includes:

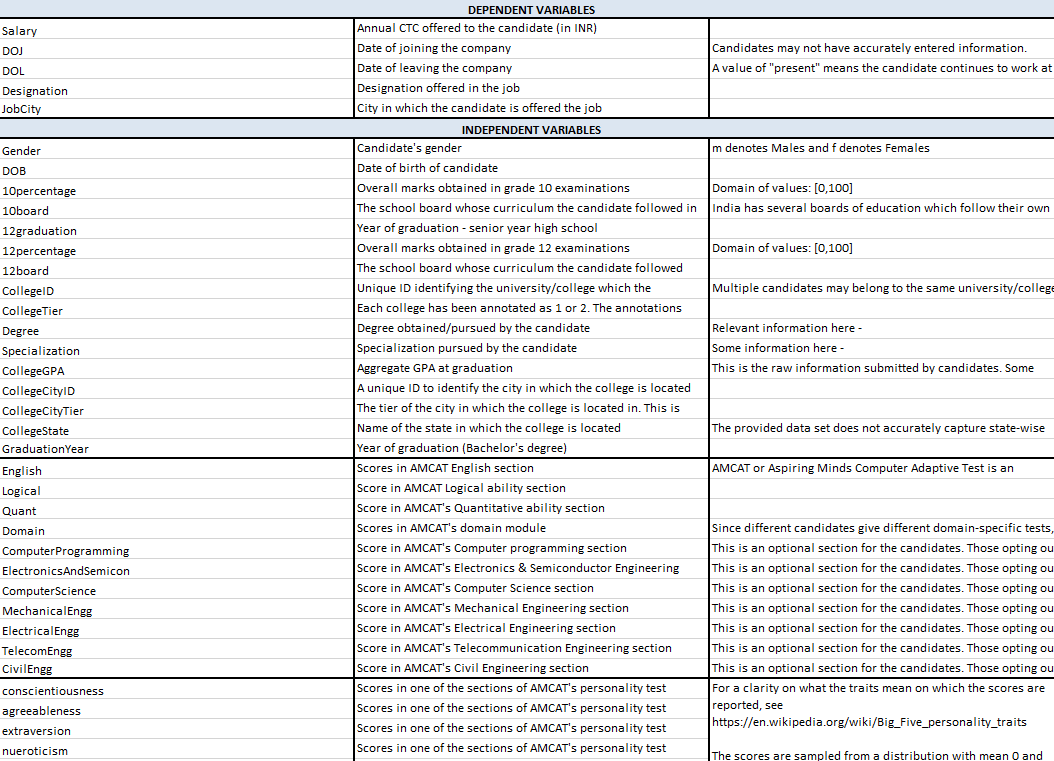
* First job annual salary
* First job title
* First job location

Random AMCAT takers were surveyed via email wherein they provided information on the dependent variables in this dataset – the jobs they are in and their corresponding annual salaries. Corresponding independent information about the candidates was recorded at the time of them taking AMCAT.

Dataset Source: <http://research.aspiringminds.com/resources/#ameo>

**VARIABLES CONSIDERED FOR ANALYSIS**

The dataset consists of 38 variables and records of 3998 graduates. Of the 38 variables 5 variables are the target variables.



**TARGET VARIABLE**

**Salary Class**

Salary class of the engineering graduates which was offered to them on joining of their first job is the target variable . It is a categorical variable and following are the categories and salary ranges for respective categories :

* ﻿Low – Ranges between35000 and 260000
* Good – Ranges between 265000 and 485000
* Average – Ranges between 490000 and 1500000
* High – Ranges between 1745000 and 4000000

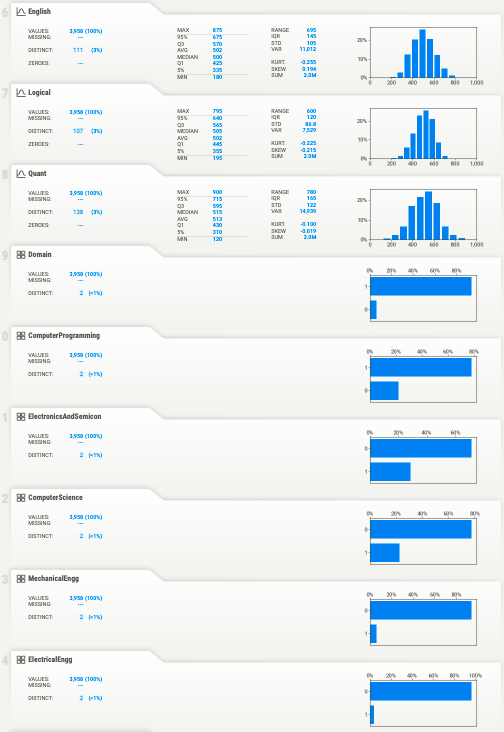
**EXPLORATORY DATA ANALYTICS**

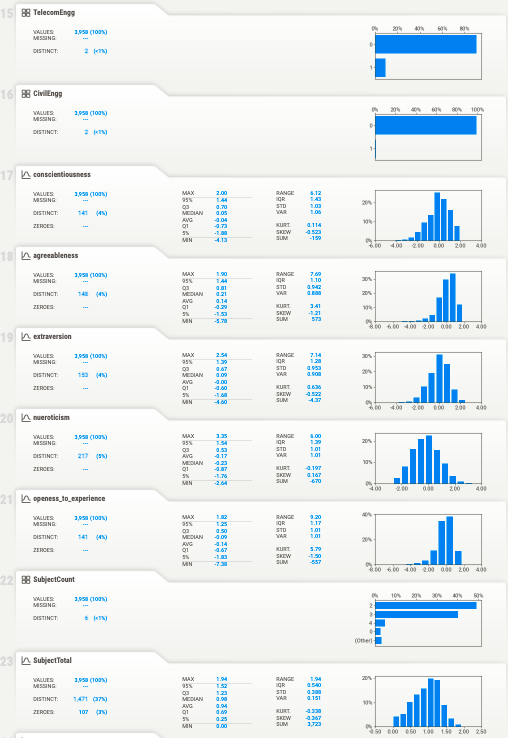
**INTRODUCTION**

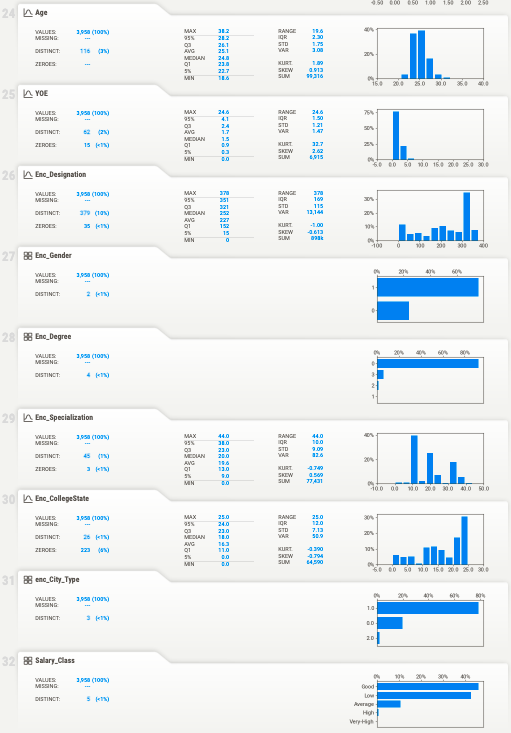
EDA is a general approach to exploring datasets by means of simple summary statistics and graphic visualizations in order to gain a deeper understanding of the data.

**Sweetviz report of the dataset.**

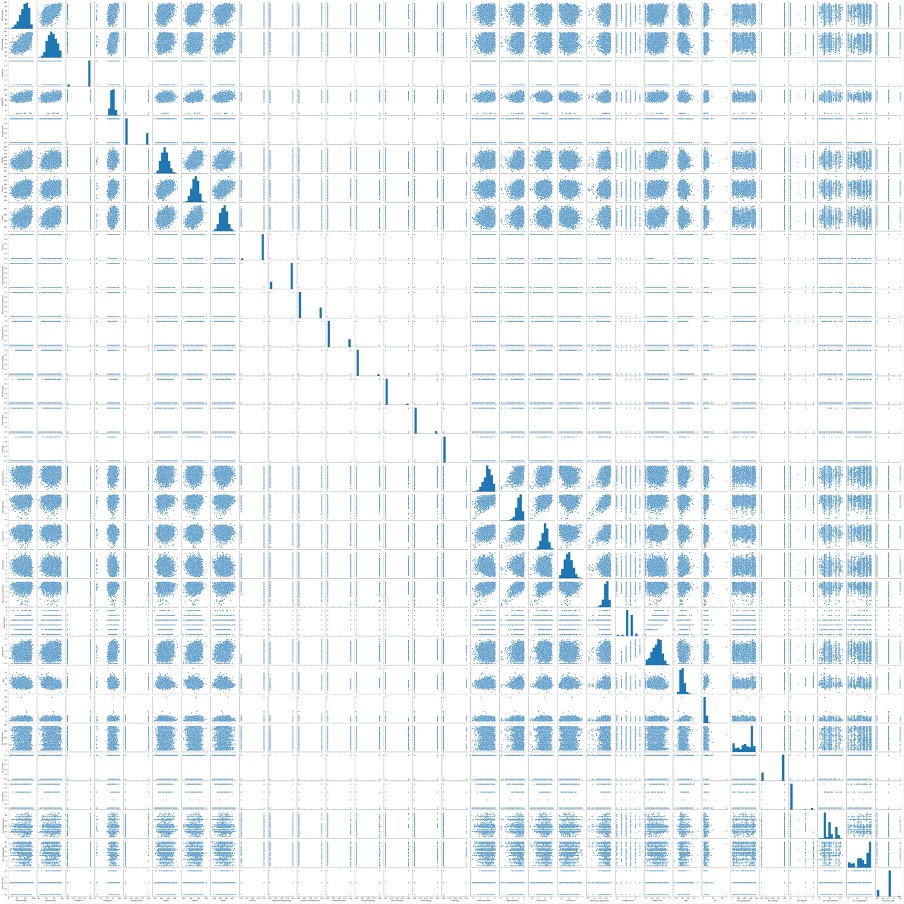
****

****

****

****

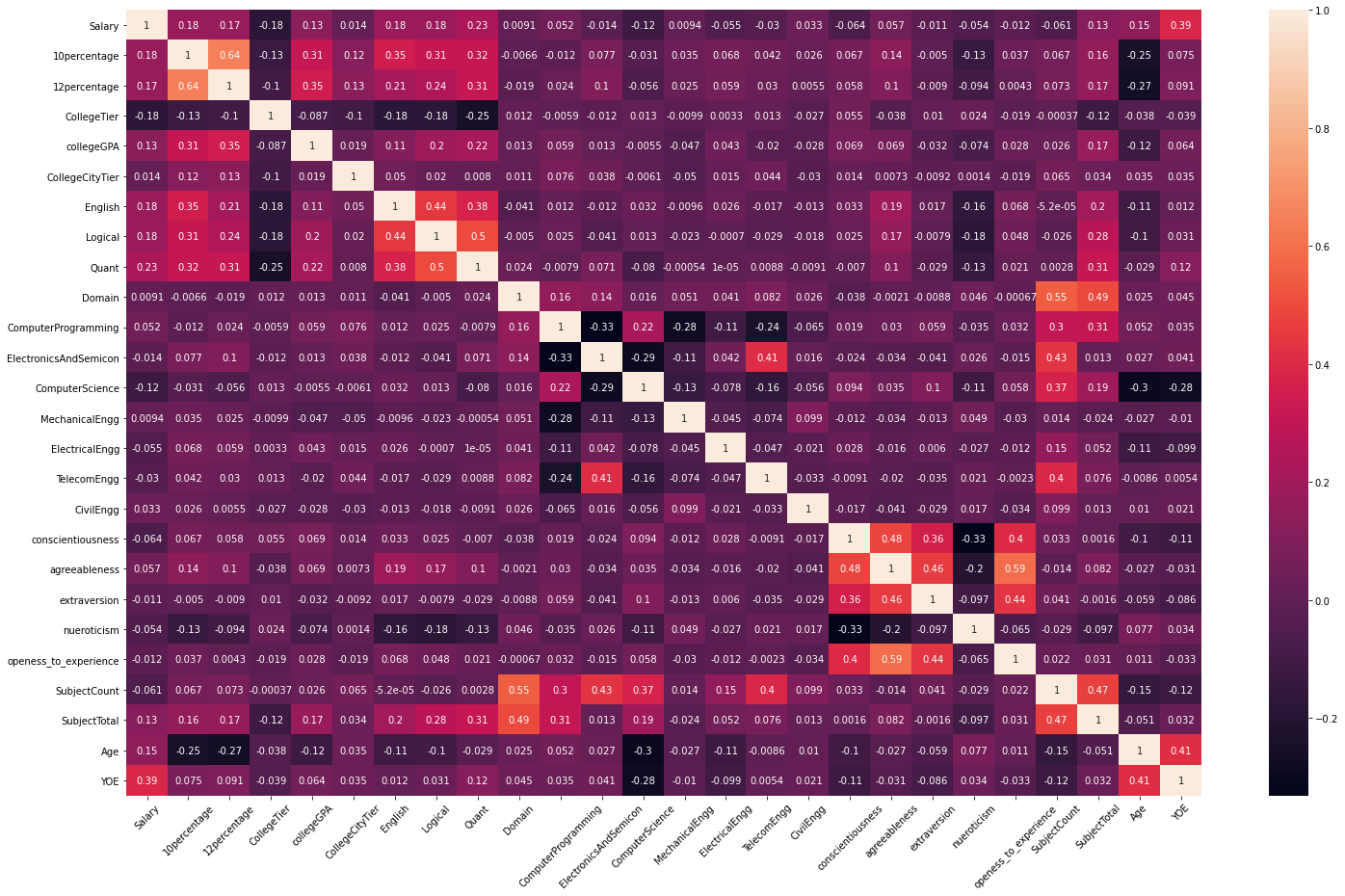
**PAIRPLOT OF THE DATASET**

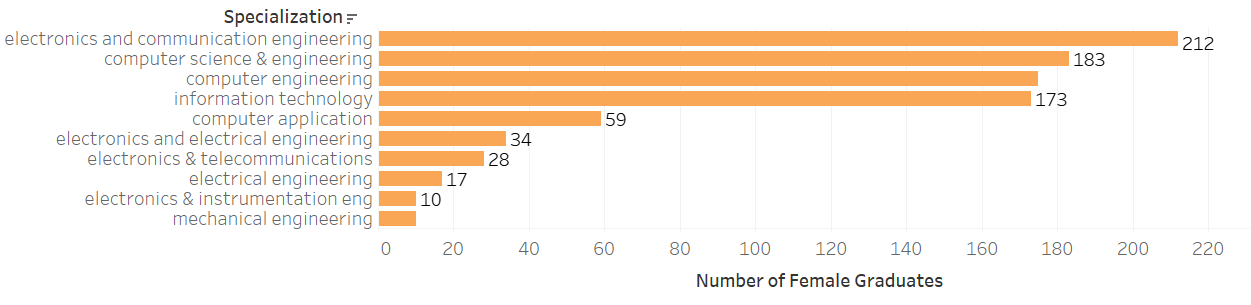


From the pair plot we can see that there is no attributes are showing significant correlation with salary.

**HEATMAP OF ATTRIBUTES**

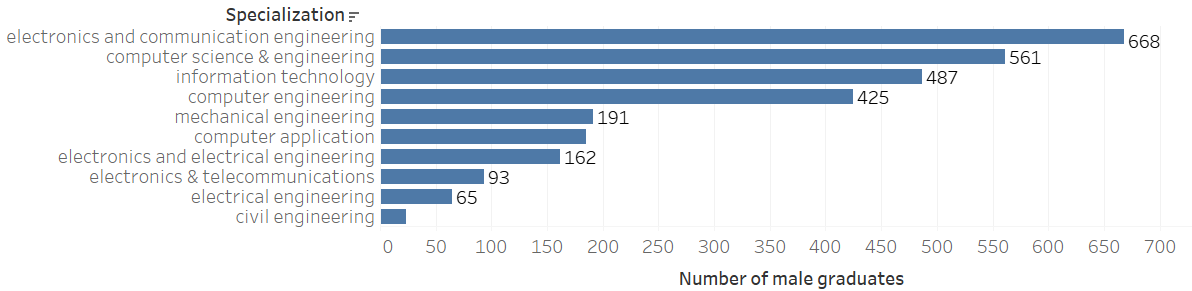
The heatmap shows the magnitude of correlation between attributes



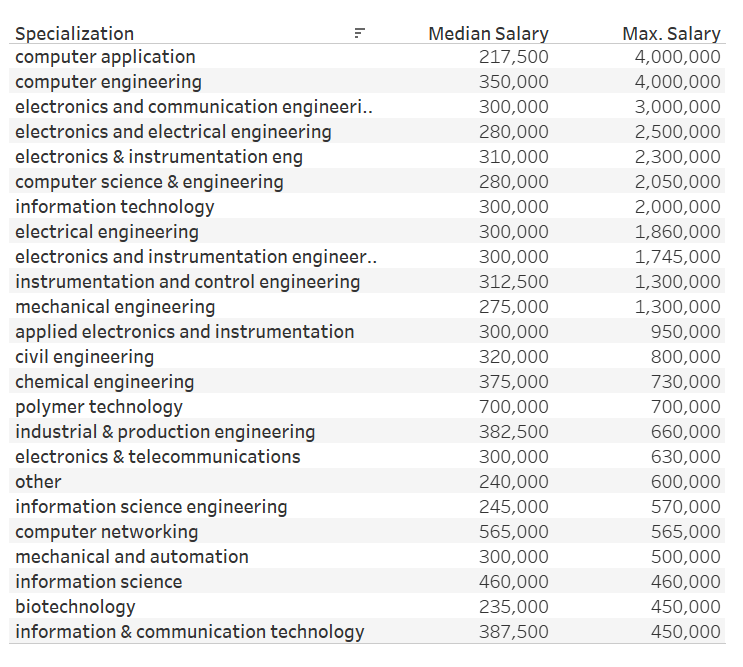
**POPULAR FIELD SPECIALISATIONS AMONG GRADUATES** 

Top 10 specializations opted by women along with their counts

Top 10 specializations opted by men along with their counts

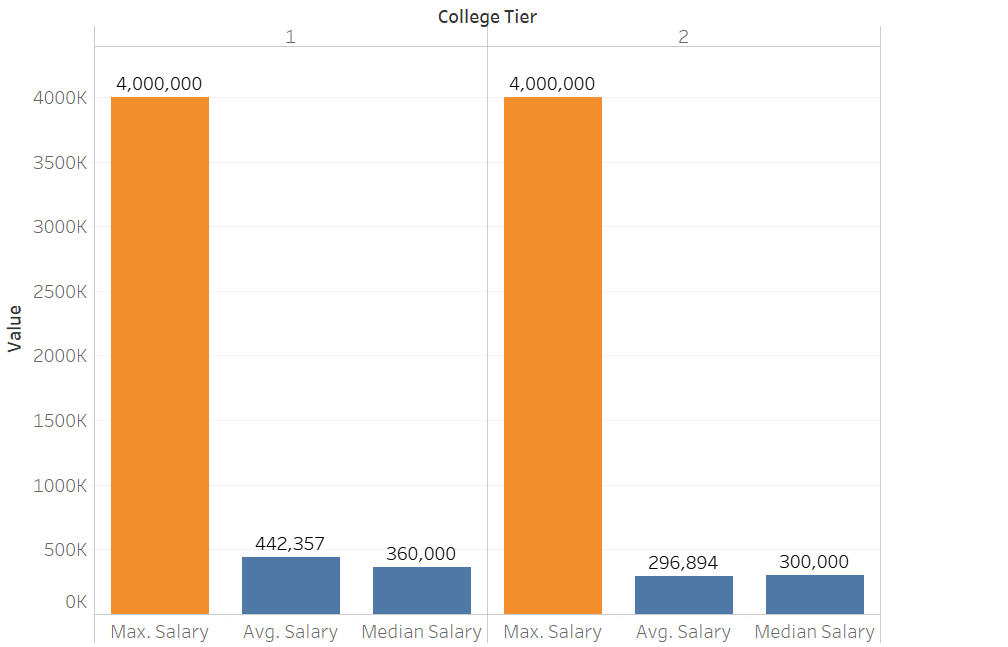


So both male and female are interested in same specializations having electronics and communication, computer science & engineering as most popular specializations.

 **SPECIALIZATIONS AND EFFECT ON SALARY**

Computer application , computer engineering and ECE are the top specializations which can land graduates with very high salary packages. The median salaries are of medium order as the number of enrolled students in these specializations is quite high.

**COLLEGE TIER AND EFFECT ON SALARY**



The college tiers seem to have a moderate effect on the salaries . Tough for both categories the maximum salaries are same , the college tier is having effect on the average salaries.

**DATA CLEANING**

**COLUMNS DROPPED**

* **﻿**Unnamed: 0
* ID
* CollegeID
* CollegeCityID
* 10board
* 12board

**FEATURE ENGINEERING**

* SubjectCount - Contains data about the count of the elective subjects.
* SubjectTotal – Contains data about total marks of the elective subjects.
* Age – Calculated using the DOB feature.
* YOE (Year Of Experience) – Calculated the difference between DOJ (Date Of Joining) and DOL (Date Of Leaving)
* Salary\_Class – Contains data after classifying the salary ranges using clusters formed by applying KMeans Unsupervised Learning Algorithm on ‘Salary’ feature.
* City\_Type – Classified the JobCity data into ‘Metropolitan’, ‘Non-Metropolitan’ and ‘Abroad’, based on the cities mentioned and used these classes to make this feature. JobCity had missing values in it, and it is an important feature in salary prediction, so we made the feature City\_Type to treat the missing values using FancyImpute.

**ROWS DROPPED**

After calculating YOE, it was observed that there were some negative values, and since these records were almost 1% of the total records so we decided to drop the records.

**MISSING VALUE TREATMENT**

City\_Type contains **449 missing values**, we used FancyImpute to impute the missing values. We used KNN algorithm to impute the missing values.

**FancyImpute**

fancyimpute is a library for missing data imputation algorithms. Fancyimpute use machine learning algorithm to impute missing values. Fancyimpute uses all the column to impute the missing values. There are two ways missing data can be imputed using Fancyimpute

1. KNN or K-Nearest Neighbour
2. MICE or Multiple Imputation by Chained Equation

**K-Nearest Neighbour**

To fill out the missing values KNN finds out the similar data points among all the features. Then it took the average of all the points to fill in the missing values.

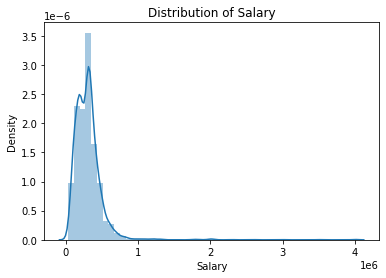
**Multiple Imputation by Chained Equation**

MICE uses multiple imputation instead of single imputation which results in statistical uncertainty. MICE perform multiple regression over the sample data and take averages of them.

**ARCHITECTURE**

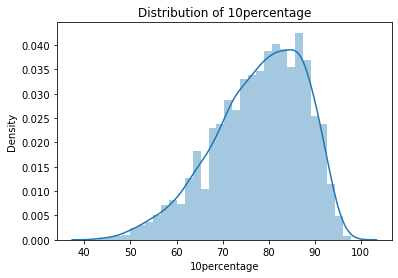
**DISTRIBUTION OF ATTRIBUTES**

1. **Salary**

****

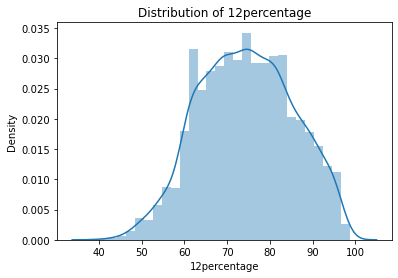
More people at the lower salary scale.

1. **Tenth percentage**

****

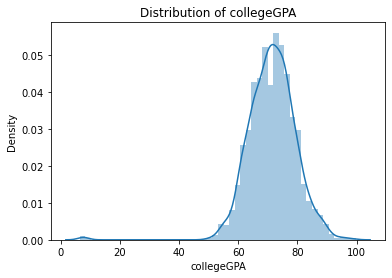
Most of the people are between 70% - 90% marks.

1. **Twelfth percentage**

****

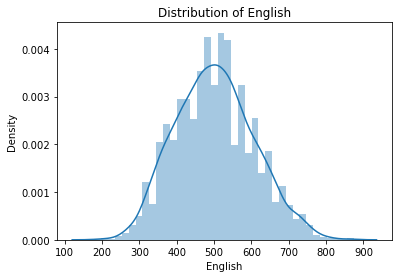
Most people are between 60% - 80% marks.

1. **College GPA**

****

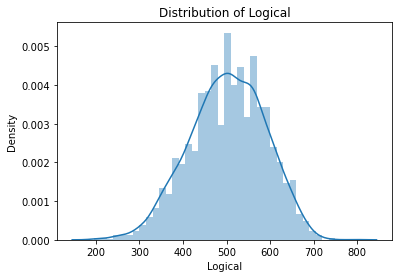
Most people are between 60 and 80 GPA.

1. **English**

****

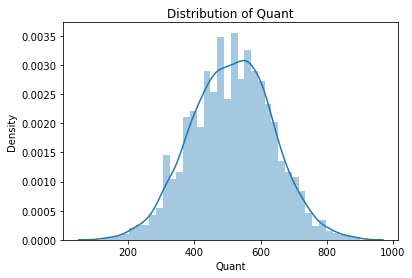
From the above plot, we can observe that English attribute distribution is normal and skewness=0.177.

1. **Logical**



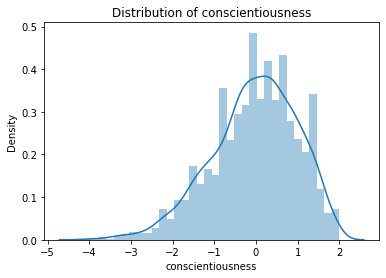
From the above plot, we can observe that Logical attribute distribution is normal and skewness= -0.206.

1. **Quant**



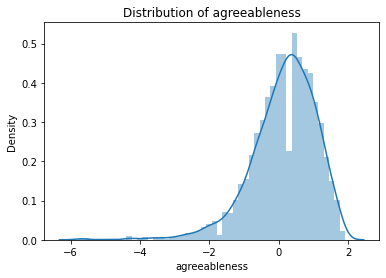
From the above plot, we can observe that Quant attribute distribution is normal and skewness= -0.0257.

1. **Conscientiousness**

****

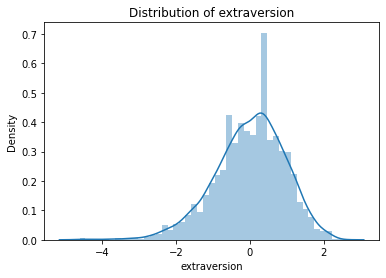
From the above plot, we can observe that Conscientiousness attribute distribution is normal and skewness= -0.503.

1. **Agreeableness**



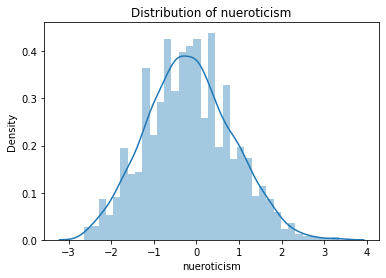
The frequency is high near zero.

1. **Extraversion**

****

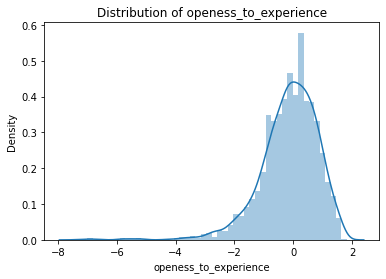
From the above plot, we can observe that Extraversion attribute distribution is not normal

1. **Neuroticism**

****

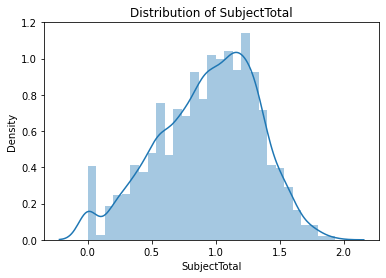
From the above plot, we can observe that Neuroticism attribute distribution is normal and skewness= 0.168 which is in acceptable range.

1. **Openness to Experience**

****

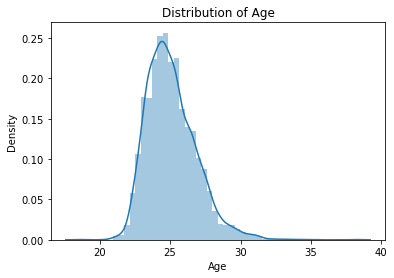
From the above plot, we can observe that Openness to Experience attribute distribution is not normal.

1. **Subject Total**

****

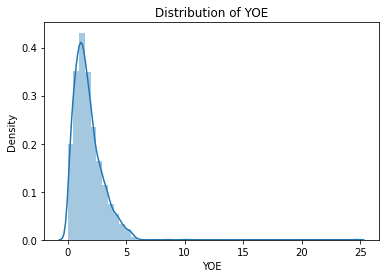
From the above plot, we can observe that Domain attribute distribution is not normal.

1. **Age**

****

Most of the people are young as high age frequency is between 20 and 30.

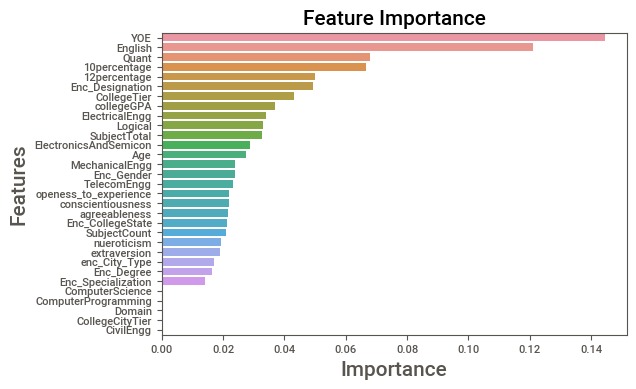
1. **YOE (Year Of Experience)**

****

Most of the people are less experience with 0-5 years of experience.

**FEATURE SELECTION**

The final features for model fitting were decided upon fitting the XGBoost model which gave us the best features.



**List of attributes which were removed were**:

* 12Graduation
* GraduationYear

**List of the final attributes selected for model fitting:**

* **﻿**10percentage
* 12percentage
* CollegeTier
* collegeGPA
* CollegeCityTier
* English
* Logical
* Quant
* Domain
* ComputerProgramming
* ElectronicsAndSemicon
* ComputerScience
* MechanicalEngg
* ElectricalEngg
* TelecomEngg
* CivilEngg
* Conscientiousness
* Agreeableness
* Extraversion
* Neuroticism
* openess\_to\_experience
* SubjectCount
* SubjectTotal
* Age
* YOE
* Enc\_Designation
* Enc\_Gender
* Enc\_Degree
* Enc\_Specialization
* Enc\_CollegeState
* enc\_City\_Type

**TENTATIVE LIST OF ALGORITHMS & INITIAL APPROACH**

Since our problem is a classification problem, we will be using the following algorithms in modelling:

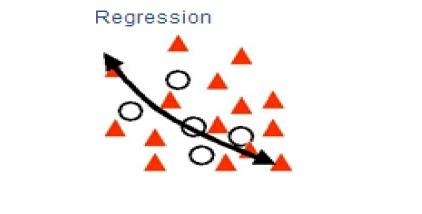
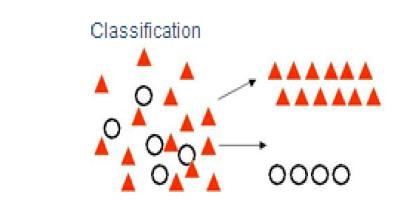
* Random Forest Classifier
* Gradient Boosting Classifier
* XGBoost Classifier

**DECISION TREE (CART)**

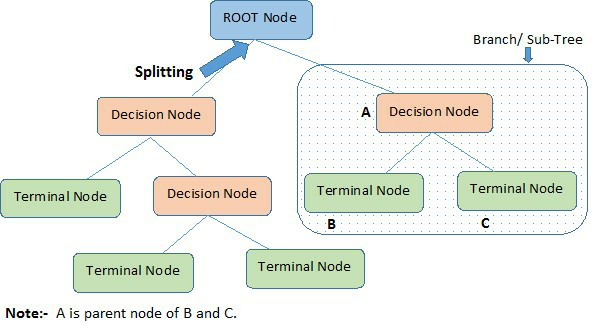
A Decision tree (CART) is a schematic, tree-shaped diagram used to determine a course of action or show a statistical probability. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

**Types of Decision Tree**

* **Classification Trees:** where the Dependent variable is categorical and the tree is used to identify the "class" within which a Dependent variable would likely fall into.
* **Regression Trees:** where the Dependent variable is continuous and tree is used to predict its value. (e.g. the price of a house, or a patient's length of stay in a hospital).



**Layout / flow of Decision Tree**



**Advantages of Decision Tree**

* Simple to understand, interpret, visualize.
* Decision trees implicitly perform variable screening or feature selection.
* Can handle both numerical and categorical data. Can also handle multi-output problems.
* Decision trees require relatively little effort from users for data preparation.
* Nonlinear relationships between parameters do not affect tree performance.

**Disadvantages of Decision Tree**

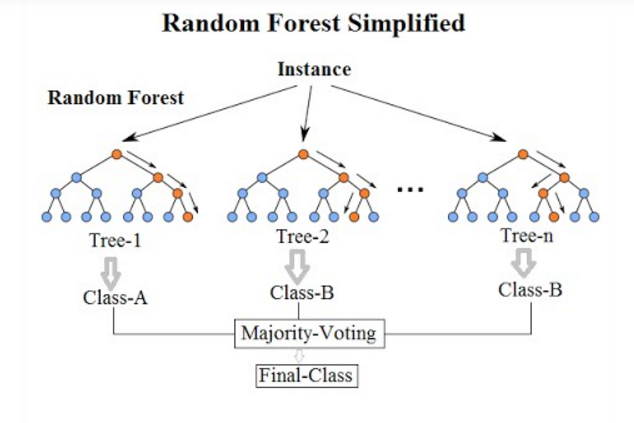
* Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting.
* Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This is called variance, which needs to be lowered by methods like bagging and boosting.
* Greedy algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement.
* Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the data set prior to fitting with the decision tree.

**RANDOM FOREST**

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

One big advantage of random forest is, that it can be used for both classification and regression problems.



Random Forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, we don’t have to combine a decision tree with a bagging classifier and can just easily use the classifier-class of Random Forest. Like I already said, with Random Forest, you can also deal with Regression tasks by using the Random Forest regressor.

Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

**Advantages of Random Forest**

* There is no need for feature normalization
* Individual decision trees can be trained in parallel
* Reduced overfitting
* Require almost no input preparation
* Performs implicit feature selection
* It’s very quick to train

**Disadvantages of Random Forest**

* No interpretability
* It requires much computational power as well as resources as it builds numerous trees to combine their outputs.
* It also requires much time for training as it combines a lot of decision trees to determine the class.
* Due to the ensemble of decision trees, it also suffers interpretability and fails to determine the significance of each variable.

**GRADIENT BOOSTING CLASSIFIER**

[Gradient boosting classifiers](https://en.wikipedia.org/wiki/Gradient_boosting) are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Gradient boosting models are becoming popular because of their effectiveness at classifying complex datasets, and have recently been used to win many [Kaggle](https://www.kaggle.com/) data science competitions.

The idea behind "gradient boosting" is to take a weak hypothesis or weak learning algorithm and make a series of tweaks to it that will improve the strength of the hypothesis/learner. This type of Hypothesis Boosting is based on the idea of [Probability Approximately Correct Learning](https://en.wikipedia.org/wiki/Probably_approximately_correct_learning) (PAC).

This PAC learning method investigates machine learning problems to interpret how complex they are, and a similar method is applied to Hypothesis Boosting.

In hypothesis boosting, you look at all the observations that the machine learning algorithm is trained on, and you leave only the observations that the machine learning method successfully classified behind, stripping out the other observations. A new weak learner is created and tested on the set of data that was poorly classified, and then just the examples that were successfully classified are kept.

### Theory Behind Gradient Boost

The Gradient Boosting Classifier depends on a [loss function](https://en.wikipedia.org/wiki/Loss_function). A custom loss function can be used, and many standardized loss functions are supported by gradient boosting classifiers, but the loss function has to be differentiable.

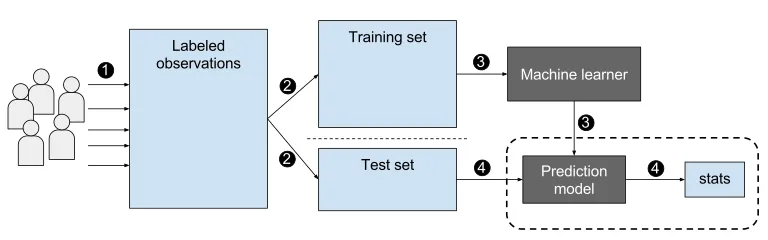
Classification algorithms frequently use logarithmic loss, while regression algorithms can use squared errors. Gradient boosting systems don't have to derive a new loss function every time the boosting algorithm is added, rather any differentiable loss function can be applied to the system.

Gradient boosting systems have two other necessary parts: a weak learner and an additive component. Gradient boosting systems use decision trees as their weak learners. Regression trees are used for the weak learners, and these regression trees output real values. Because the outputs are real values, as new learners are added into the model the output of the regression trees can be added together to correct for errors in the predictions.

The additive component of a gradient boosting model comes from the fact that trees are added to the model over time, and when this occurs the existing trees aren't manipulated, their values remain fixed.

A procedure similar to gradient descent is used to minimize the error between given parameters. This is done by taking the calculated loss and performing gradient descent to reduce that loss. Afterwards, the parameters of the tree are modified to reduce the residual loss.

The new tree's output is then appended to the output of the previous trees used in the model. This process is repeated until a previously specified number of trees is reached, or the loss is reduced below a certain threshold.



**Advantages of Gradient Boosting**

* Often provides predictive accuracy that cannot be beat.
* Lots of flexibility - can optimize on different loss functions and provides several hyperparameter tuning options that make the function fit very flexible.
* No data pre-processing required - often works great with categorical and numerical values as is.
* Handles missing data - imputation not required.

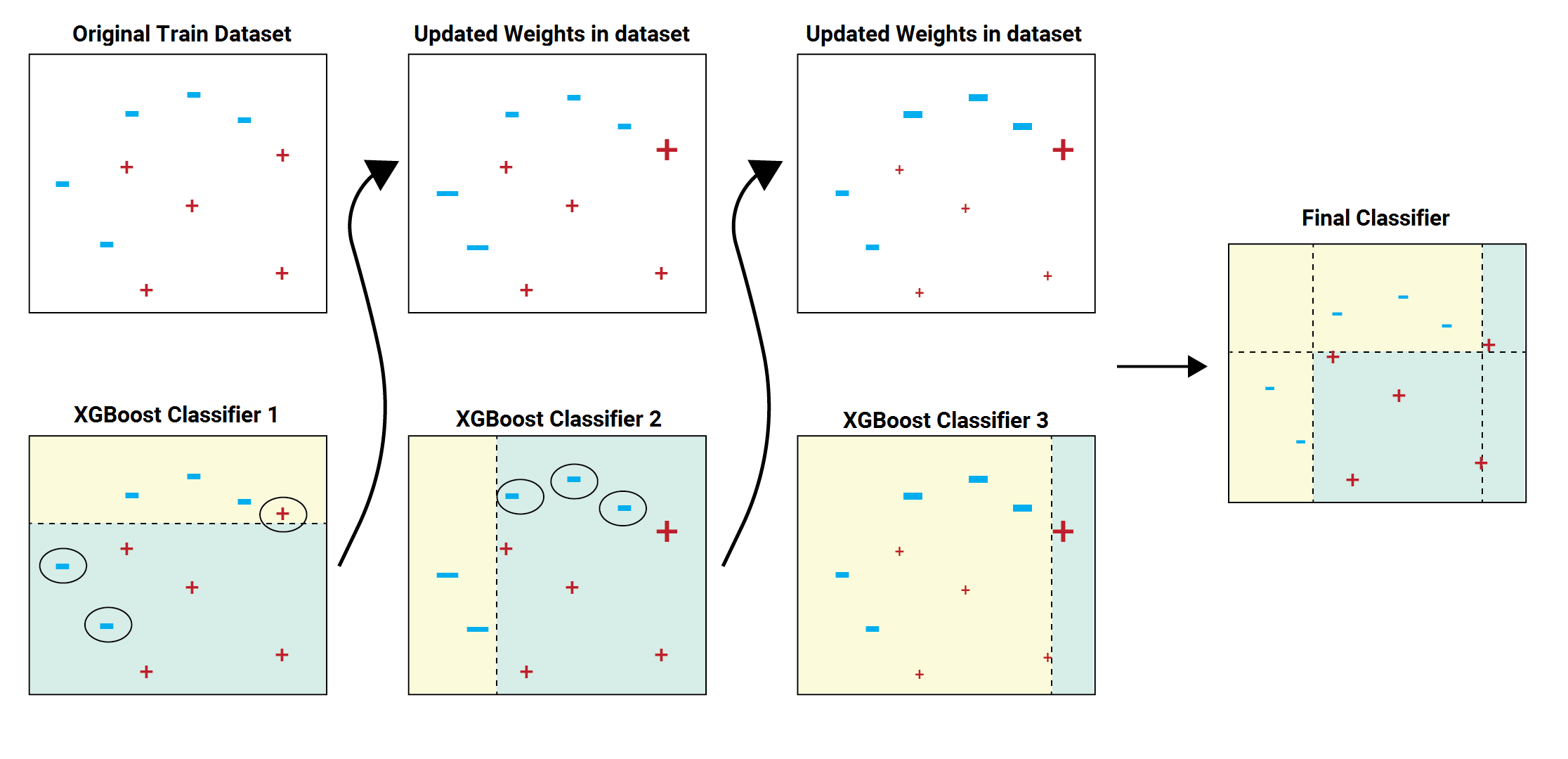
**Disadvantages of Gradient Boosting**

* GBMs will continue improving to minimize all errors. This can overemphasize outliers and cause overfitting. Must use cross-validation to neutralize.
* Computationally expensive - GBMs often require many trees (>1000) which can be time and memory exhaustive.
* The high flexibility results in many parameters that interact and influence heavily the behaviour of the approach (number of iterations, tree depth, regularization parameters, etc.). This requires a large grid search during tuning.
* Less interpretable although this is easily addressed with various tools (variable importance, partial dependence plots, LIME, etc.).

**XGBOOST**

XGBoost stands for e**X**treme **G**radient **B**oosting.

XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data.It is an implementation of gradient boosted decision trees designed for speed and performance.



The two reasons to use XGBoost are also the two goals of the project:

* Execution Speed.
* Model Performance.

The XGBoost library implements the [gradient boosting decision tree algorithm](https://en.wikipedia.org/wiki/Gradient_boosting).

This algorithm goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines.

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. A popular example is the [AdaBoost algorithm](https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/) that weights data points that are hard to predict.

Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

This approach supports both regression and classification predictive modelling problems.

**Advantages of XGBoost**

* Extremely fast (parallel computation).
* Highly efficient.
* Versatile (Can be used for classification, regression or ranking).
* Can be used to extract variable importance.
* Do not require feature engineering (missing values imputation, scaling and normalization)

**Disadvantages of XGBoost**

* Only work with numeric features.
* Leads to overfitting if hyperparameters are not tuned properly.

**MODEL FITTING & COMPARISON**

**Random Forest**

Below is the confusion matrix , classification report and accuracy scores for Random Forest classifier with criterion as entropy and max depth = 9 obtained by 5 fold cross validation(showing three results only)

precision recall f1-score support

Average 0.83 0.11 0.19 137

Good 0.62 0.76 0.68 536

High 0.00 0.00 0.00 5

Low 0.73 0.74 0.74 510

accuracy 0.67 1188

macro avg 0.55 0.40 0.40 1188

weighted avg 0.69 0.67 0.65 1188

Inference:

Random forest could only get one percent higher accuracy than Decision Trees . The difference between the precision and recall scores is relatively higher than Decision trees

**Gradient Boosting**

Below is the confusion matrix , classification report and accuracy scores for Gradient Boosting with max\_depth. = 4, loss = ‘deviance’ and criterion = ‘friedman\_mse’, obtained by 5 fold cross validation(showing three results only).

﻿ precision recall f1-score support

Average 0.62 0.20 0.30 119

Good 0.63 0.74 0.69 548

High 0.00 0.00 0.00 8

Low 0.72 0.71 0.72 513

accuracy 0.67 1188

macro avg 0.49 0.41 0.43 1188

weighted avg 0.67 0.67 0.66 1188

Inference:

Gradient Boosting gives the accuracy of 67 which is little higher than other two models.

**XGBoost**

precision recall f1-score support

Average 0.61 0.28 0.38 109

Good 0.69 0.72 0.71 580

High 0.00 0.00 0.00 5

Low 0.72 0.78 0.75 494

accuracy 0.70 1188

macro avg 0.51 0.45 0.46 1188

weighted avg 0.70 0.70 0.69 1188