Predicting Bank Loan Status

Viveksinh Solanki

Ronald Fernandes

Yatri Kalathia

Course: FE-582

Sem: Spring 2020

Stevens Institute of Technology

Abstract

To help financial institutions decide on whom to give loan, we chose to work on this project. We obtained publicly available dataset from kaggle. We have done comprehensive analysis of the dataset to provide valuable insights in the Exploratory Data Analysis section. Next, we utilized a few machine learning models to create our final classification model, which can predict whether a loan borrower will default or not based on loan borrower's details.

Introduction

Financial institutions like banks incur significant losses due to default of loans. This has led to tightening up of loan underwriting and increased loan rejection rates. The need for the better credit risk scoring model is also raised by there institutions. This warrants a study to estimate the determinants of loan default. So, it is very important for any bank to check the background of its customer before lending a loan. Based on customer's background, bank takes the decision whether to lend a loan or not to that customer. Doing so will ensure that clients capable of repayment are not rejected and important determinants can be identified which can be used for minimizing the default rates.

We are interested in finding out the probable parameters which might lead to loan defaults. Hence, we have chosen to work on this problem. We aim to identify the probable parameters by applying various data analysis methods and machine learning models. Our final model will be able to predict whether the customer will default a loan or not based on given parameters.

Data and Methods

Data Description

Data Source: Bank Loan status - Kaggle

This dataset is obtained from Kaggle. There are total 100,000 records with 19 columns(features) having numerical as well as categorical features. We are trying to classify Loan Status.

Loan ID	String denoting the ID of loan
Customer ID	String denoting the ID of customer
Loan Status	Boolean with following values: Charged Off - Defaulted Fully Paid - Paid
Current Loan Amount	Amount of the Loan
Term	Boolean with following values: 1) Short Term 2) Long Term
Credit Score	Credit score of the customer
Annual Income	Annual Income of the customer
Years in current job	Customer's total number of years in current job
Home Ownership	Specifies the ownership of home in which customer is living: 1) Have Mortgage 2) Home Mortgage 3) Own 4) Rent
Purpose	Purpose of the loan
Monthly Debt	Monthly debt of the customer

Years of Credit History	# of years of customer's credit history
Months since last delinquent	# of months since last delinquent
Number of Open Accounts	# of open accounts of customer
Number of Credit Problems	# of credit problems of customer
Current Credit Balance	Customer's current credit balance
Maximum Open Credit	Customer's maximum preapproved credit
Bankruptcies	# of bankruptcies customers went through till now
Tax Liens	# of tax liens of customer till now

Data Cleaning and Preprocessing

It was a trial and error process. Started with traditional Cleaning Methods that included:

- Removing "Months Since last Delinquent" column
  Checked for the summary of the data. It showed that "Months since last Delinquent" column had more than 50% null values. It was better to remove the entire column.
- Removing Nulls
  Considered only complete cases initially i.e removed all records even with one null value in a record. There were many strings that were null but were not directly stated. These strings include: 'n/a', blanks (''), the string 'NA' etc.
- Eliminating Duplicates:
  There were many duplicate entries which were removed using Loan.id
- Credit Score:

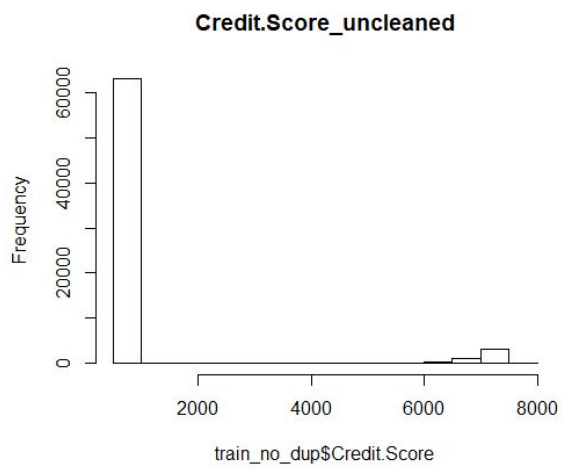
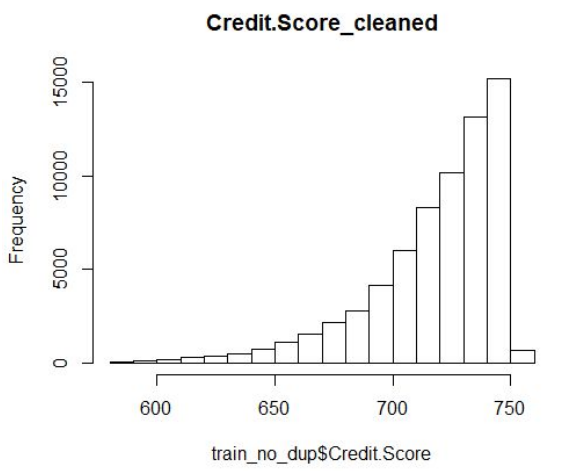
Credit Score contained values between the range 5000-8000. We compared the actual range most of the data had which was 500-800. These values may have been recorded as 10 times the actual value. Hence, we divided these values by a factor of 10.

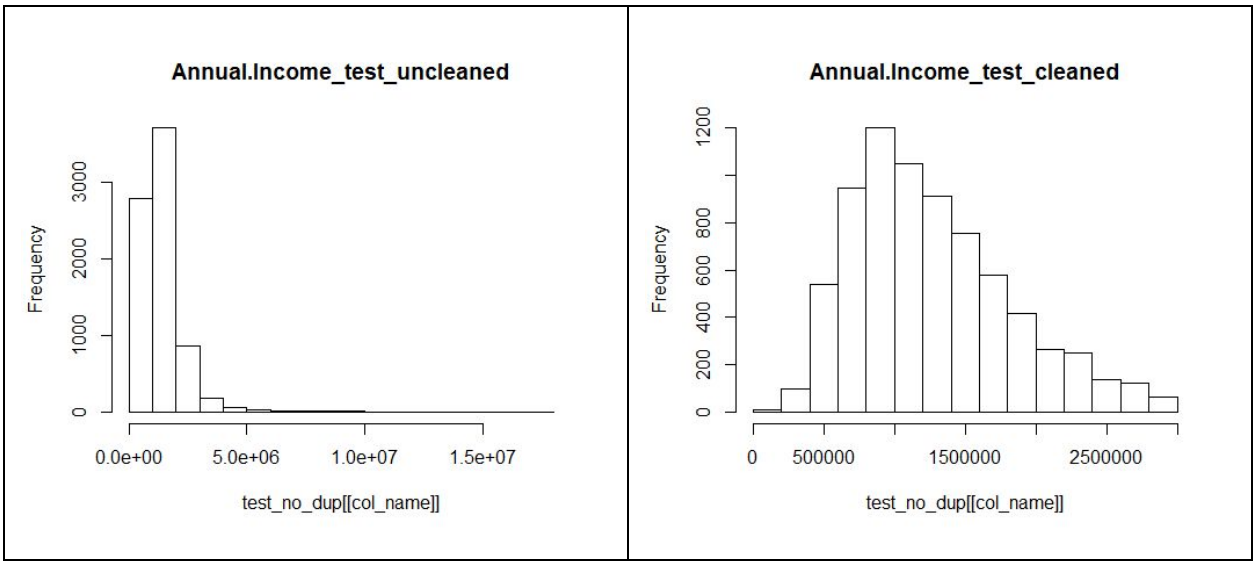
Outlier Detection

It was divided into two categories:

- Numerical Columns:
  Visible outliers were removed directly e.g Current Loan Amount 99999999 existed in many records which were eliminated. Implemented an outlier range as: 25% - 1.5*IQR to 75% + 1.5*IQR.
  Values 1.5 times the interquartile range below 25 percentile and 1.5 times the interquartile range above 75 percentile were considered outliers.
- Categorical Columns:
  Checked categorical columns for balanced samples. Even if not perfectly balanced, checked that at least it was not unbalanced to an extent that one category was too many or too few. Also, they were one-hot encoded so as the machine learning models can accurately predict its result.

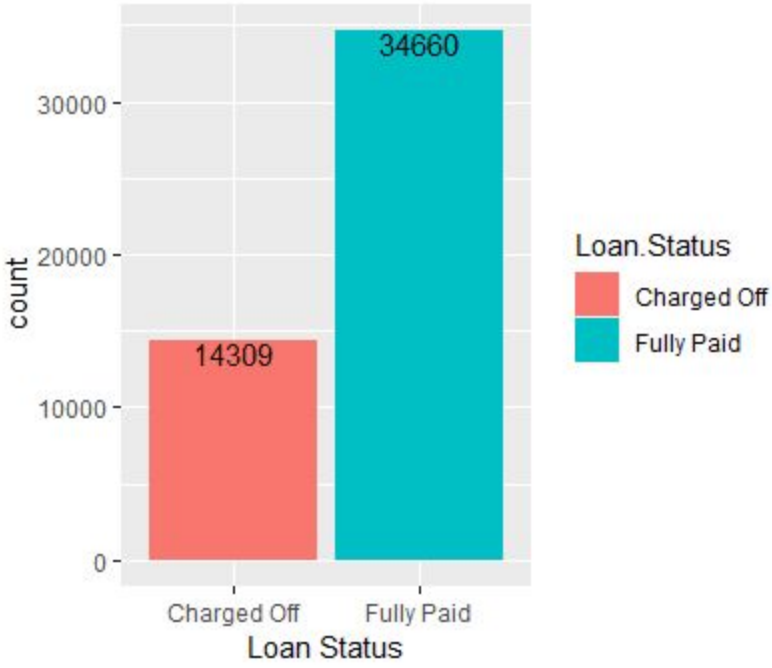
Results after pre-processing:

BEFORE CLEANING	AFTER CLEANING
	

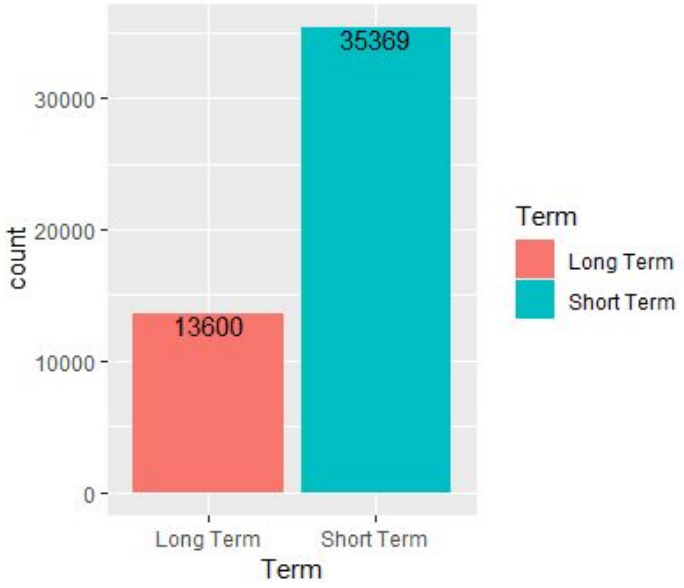


Exploratory Data Analysis

1) Counts per loan status:

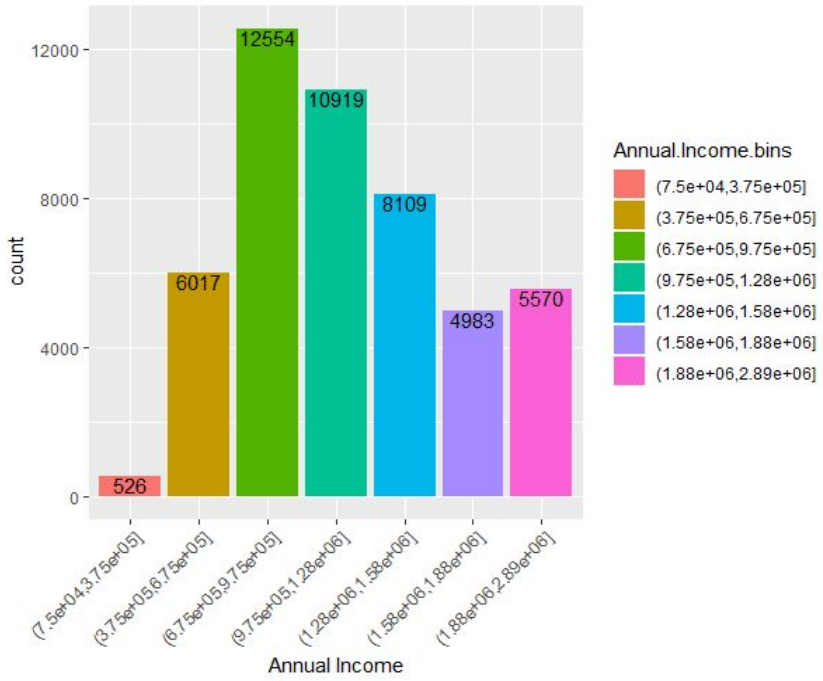


In our dataset, there are more people(34660) with ‘fully paid’ loan status than ‘charged off’ loan status. So, it is kind of an unbalanced dataset for classification task.

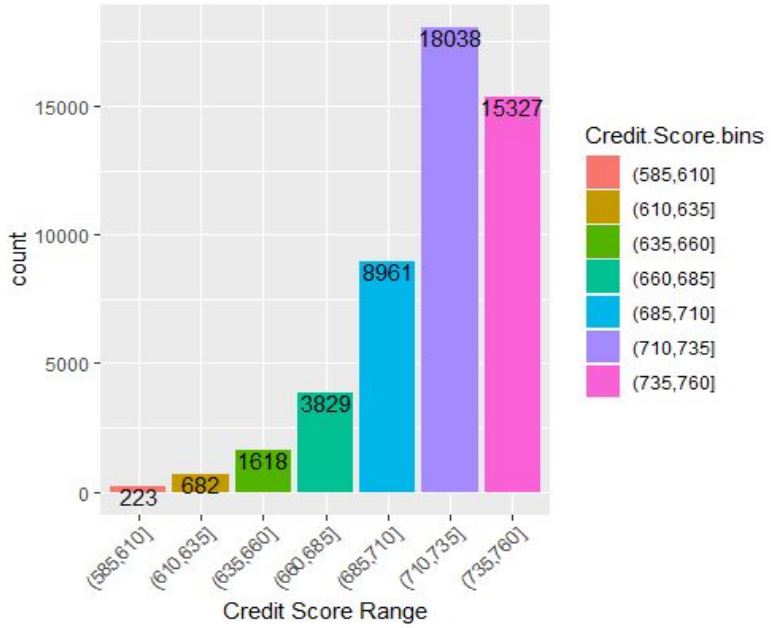


2) Counts per term:
   We can observe that people tend to borrow short term loans then long term loans.

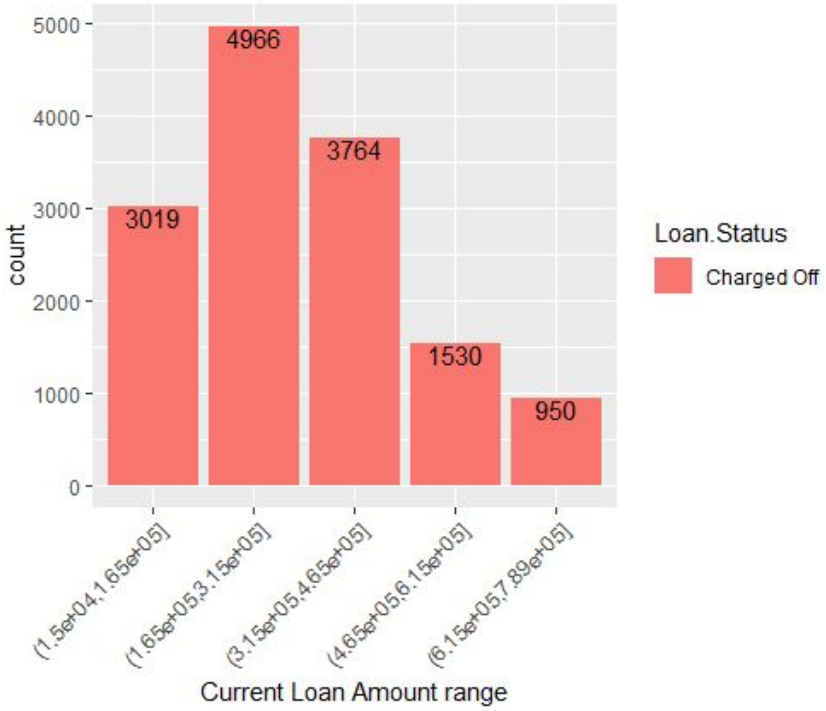
3) Counts per annual income range:



4) Counts per Credit Score range:
   Obvious observation: The higher the credit score, more the chances of getting a loan

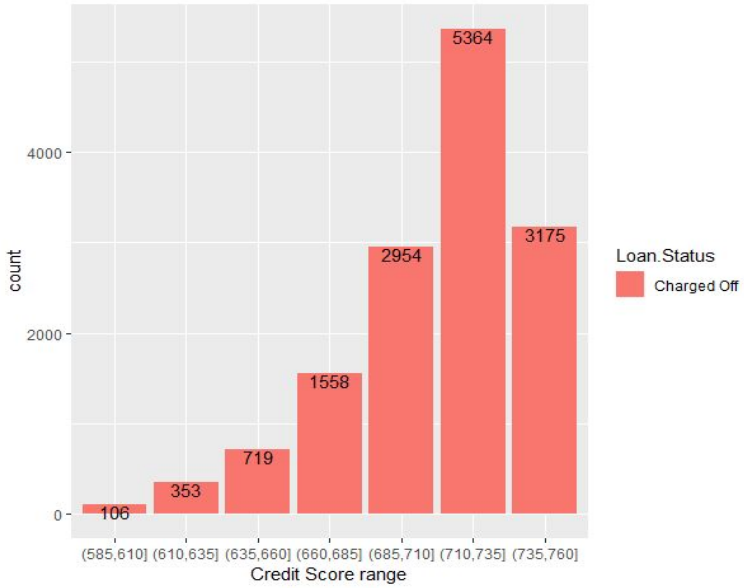


5) Current loan amount Range vs count for loan defaulters:



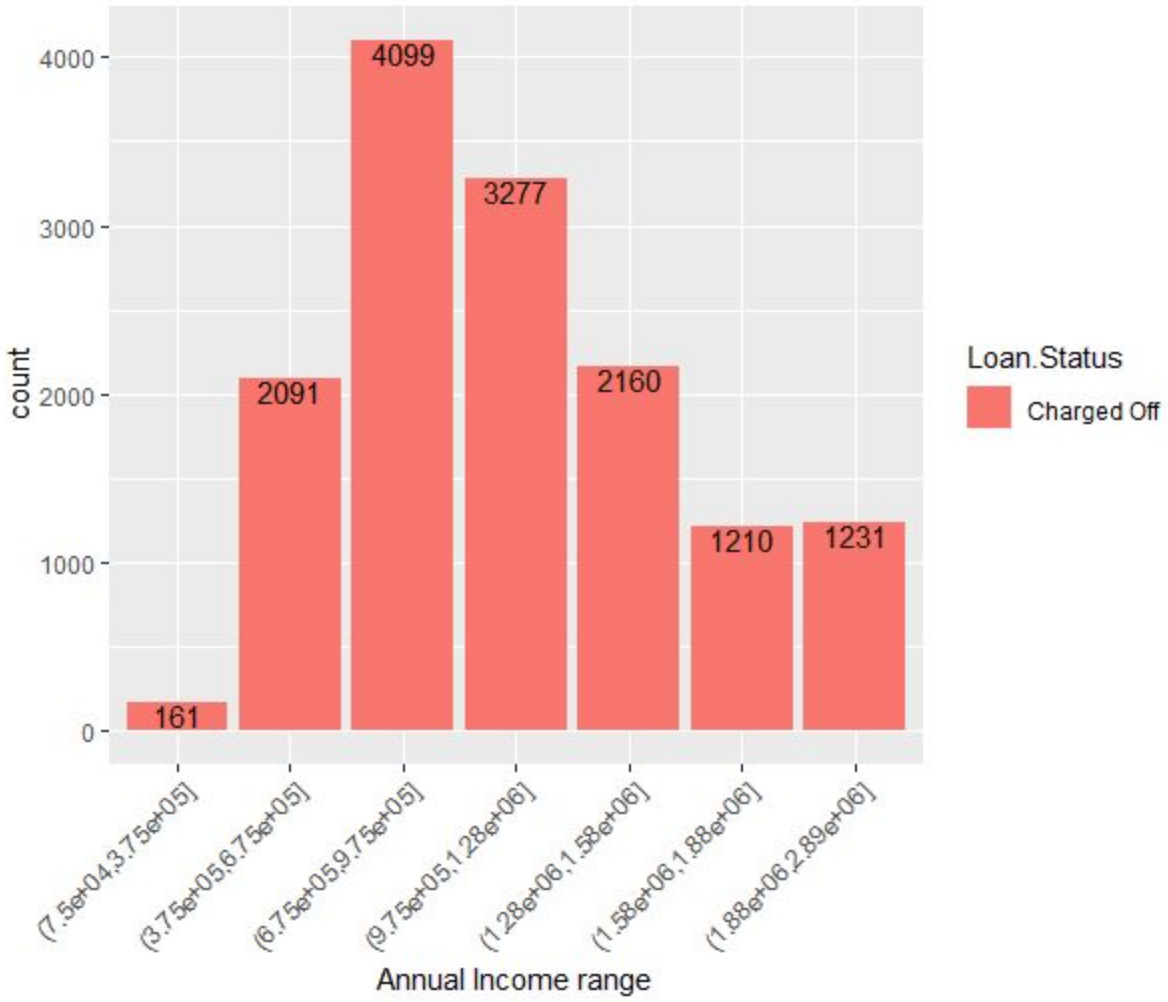
We can clearly see that ~63%(~8200) loan defaulters are in current-loan-amount range of 165k - 465k. That means higher chances of loan default in this range.

6) Credit Score Range vs count for loan defaulters:



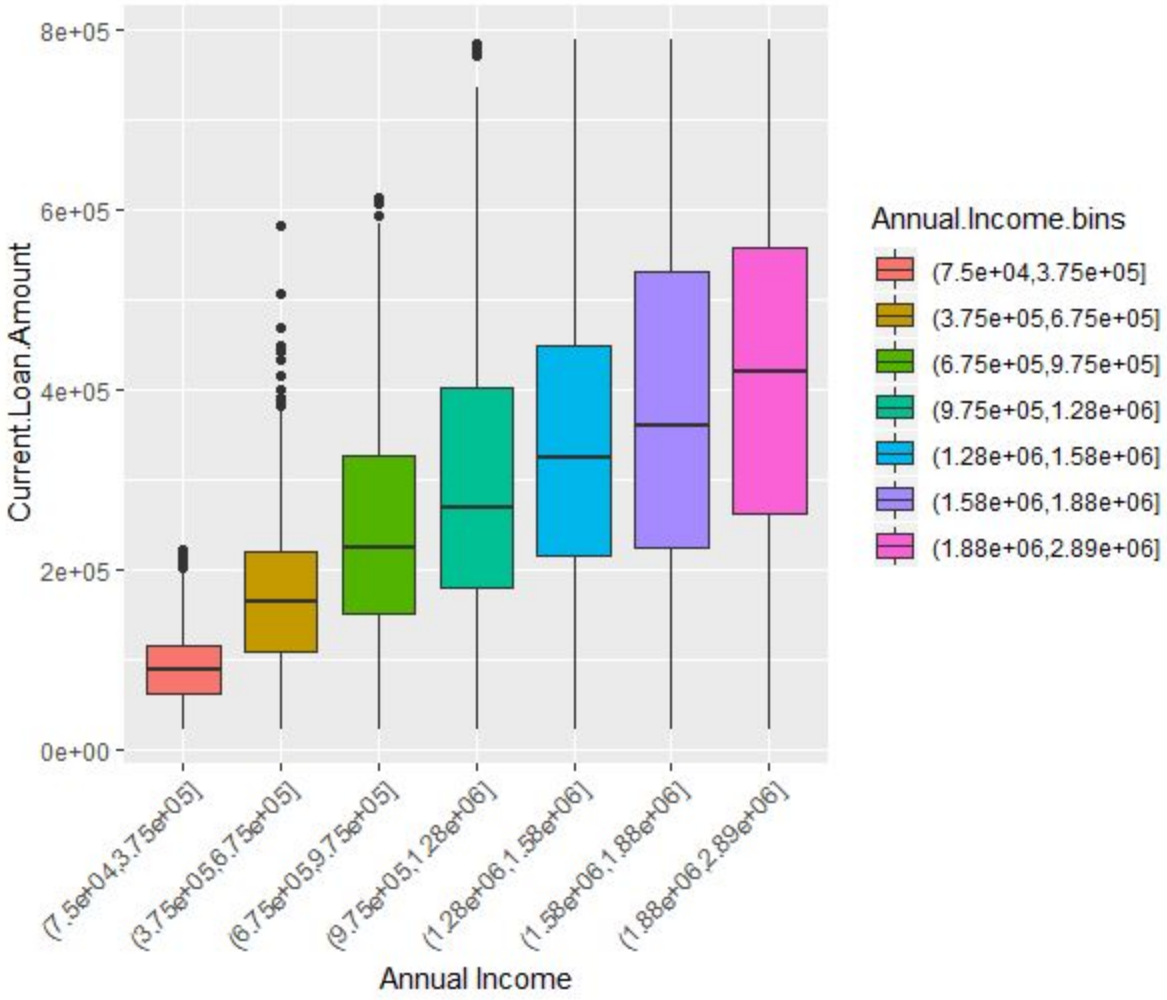
Surprisingly enough ~40% (5391) loan defaulters have credit score between 710-735.

7) Annual Income Range vs count for loan defaulters:



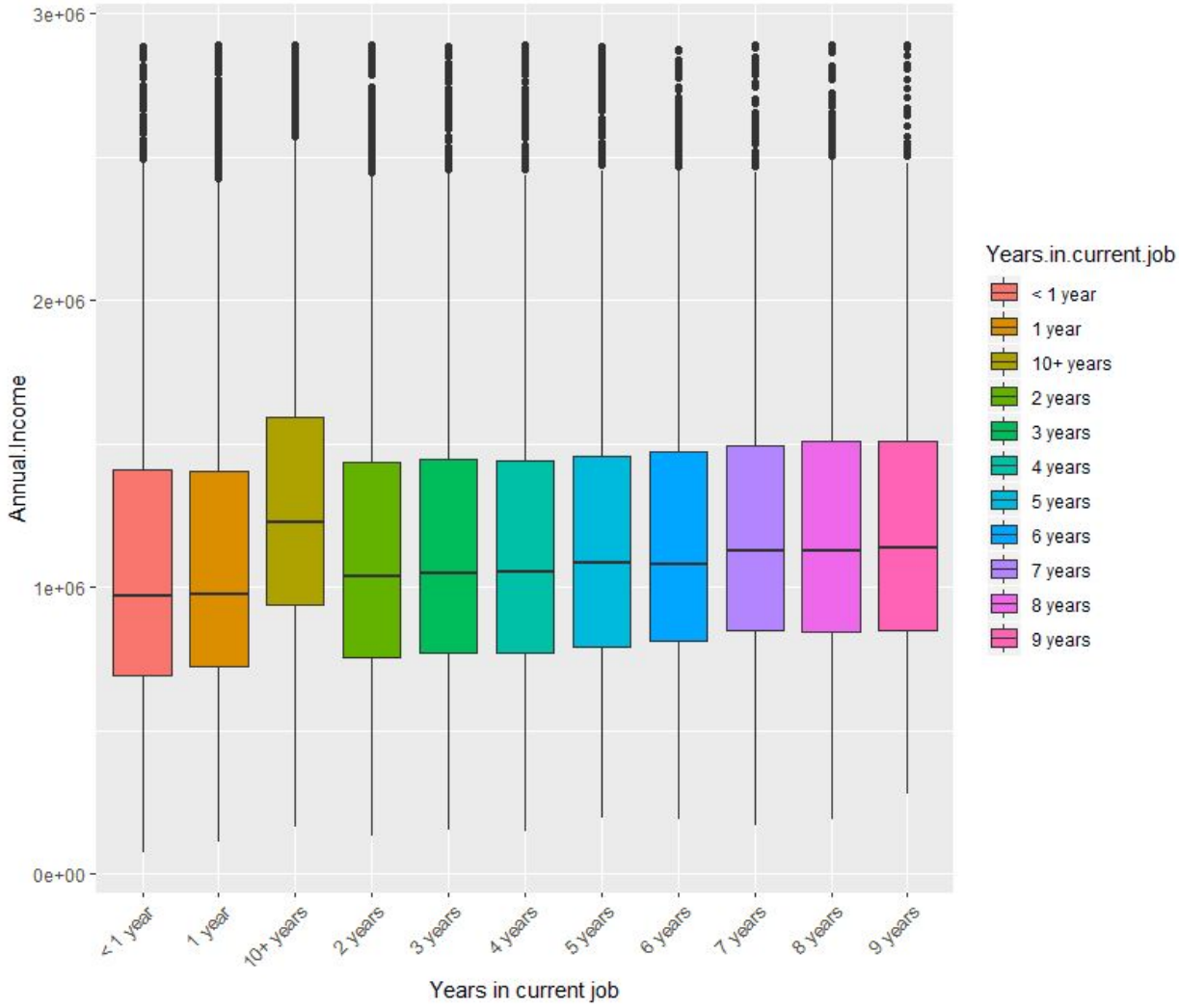
As we can see, there are ~4000 loan defaulters whose income is between 675k - 975k. There are ~3000 defaulters who have income between 975k - 1275k. Hence, almost ~54% (~7000) loan defaulters come in the income range of 675k - 1275k.

8) Annual Income seems to be positively correlated with Current Loan Amount:



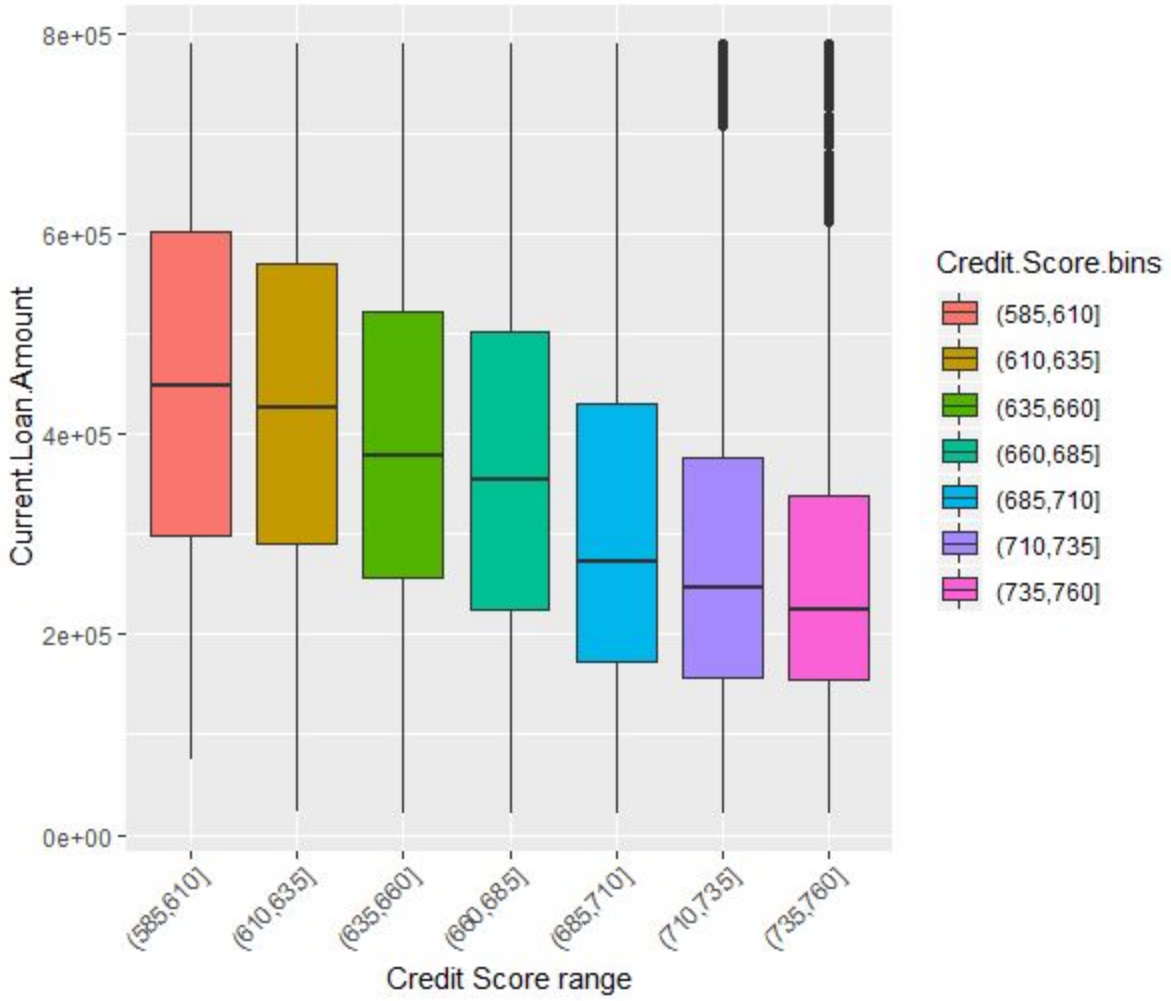
As annual income increases, the current loan amount range also increases with it.

9) Annual Income vs Years in current job:



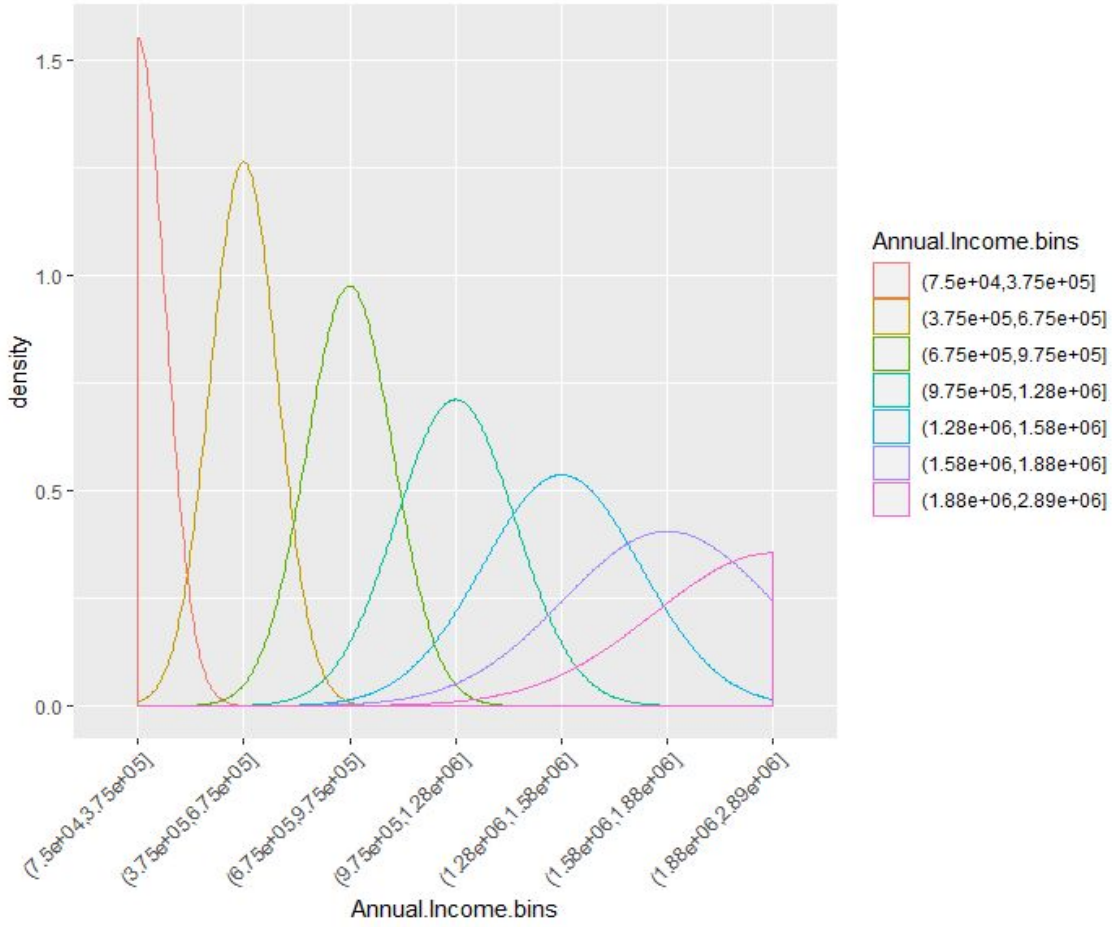
This plot shows that regardless of the number of years in a current job, the annual income of all people seems to lie between specific range. Just slightly higher annual income for people with 10+ years in current job.

10) Current loan amount vs Credit score:



Current loan amount is negatively correlated with credit score. As we can see from the plot, as credit scores increase, current loan amount decreases. Hence, people with higher credit scores don’t borrow high amount loans.

11) Density plot of annual income:



There are more people with lesser income as compared to the number of people with high income. This observation proves current inequalities in the real world as well.

Classification Methods

In this project, we implemented the following algorithms to train our data:

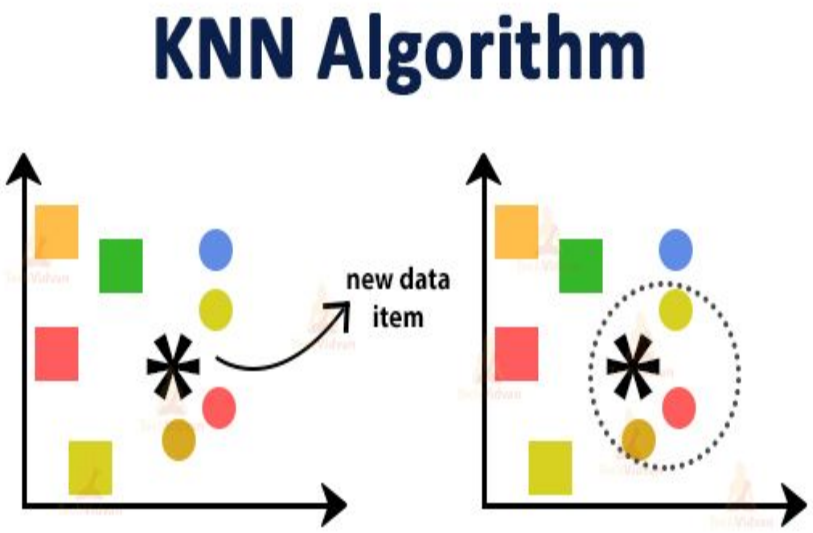
1. Naive-Bayes Classification
2. k-NN Classification
3. Logistic Regression
4. Random Forest
5. Gradient Boosting Machine

1. Naive Bayes Classification

It is a technique based on Bayes' Theorem with a "naive" assumption of conditional independence between every pair of features. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to perform quite decently especially when assumption of independence holds. Due to these reasons, we implemented this model initially since it would provide us with a benchmark to compare the other models with. When we run it against our dataset, we get an accuracy of 66%

2. k-Nearest Neighbour Algorithm

The k-Nearest Neighbor algorithm is an example of instance-based learning where training set records are first stored. Next, the classification of a new unclassified record is performed by comparing it to records in the training set it is most similar to. In simple words, the k-NN algorithm assumes that similar things exist in close proximity i.e. they are near to each other.



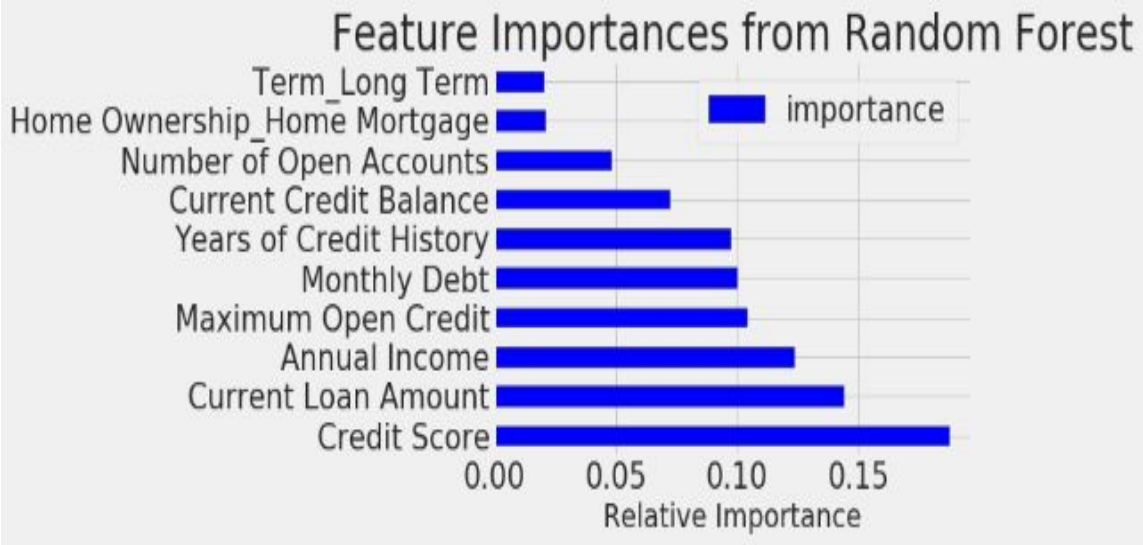
After running this algorithm against our dataset, we found the most efficient solution to be when k was chosen to be 9. Using this algorithm, we could get a model with an accuracy of 65%

3. Logistic Regression

Since, our dependent variable is categorical in nature, we also performed logistic regression. For logistic regression, we first standardized our data since all the variables were not in the same units.After that, we removed highly co-linear features of our data. We ended up considering 41 features out of 44 to train our data since 4 features were highly correlated. Using this algorithm against our data, we could get a model with 67% of accuracy.

4. Random Forest Algorithm

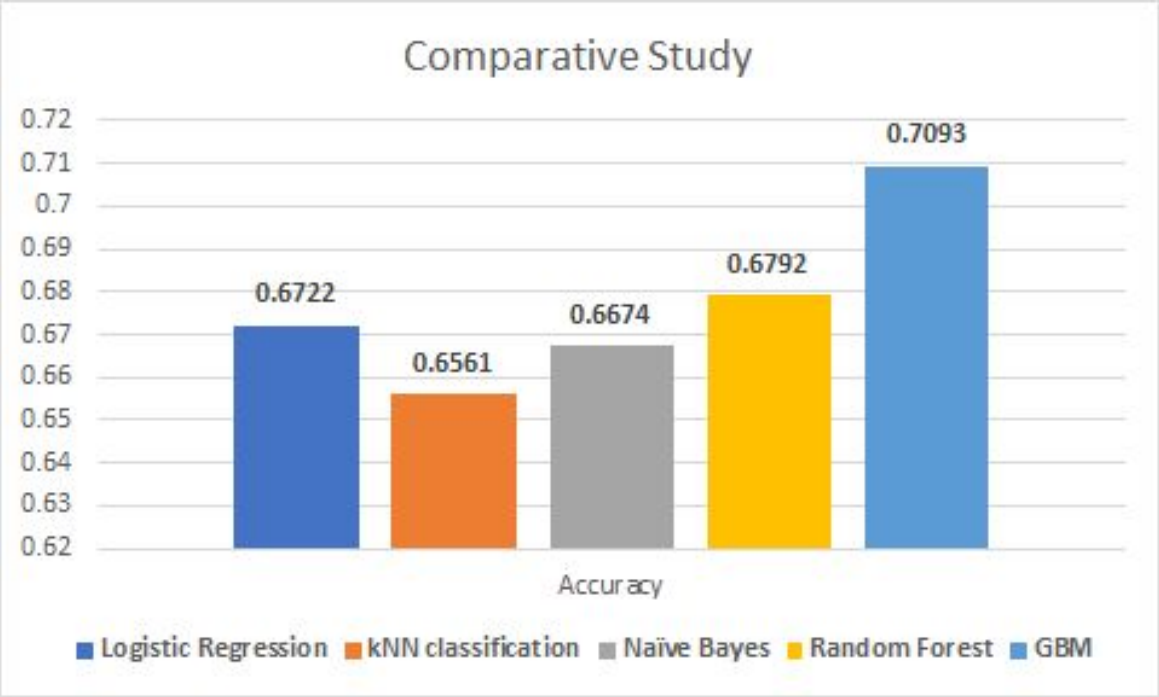
Decision trees tend to have high variance since they tend to overfit on training data. The random forest builds an ensemble of Decision Trees. The Random Forest algorithm randomly selects observations and features to build several decision trees and then averages the results. Random Forest prevents overfitting most of the time, by creating random subsets of the features and building smaller trees using these subsets. Afterwards, it combines the subtrees. Using this algorithm, we could get the list of most important features.



5. Gradient Boosting Machine

When we try to predict the target variable using any machine learning technique, the main causes of difference in actual and predicted values are noise, variance, and bias. Ensembling helps us to reduce the bias and variance. Ensembling techniques are further classified into Bagging and Boosting. **Bagging** is a simple ensembling technique in which we build many independent models and combine them using some model averaging techniques. **Boosting** is an ensemble technique in which the predictors are not made independently,but sequentially. **GBM** is an efficient and powerful algorithm for classification and regression problems. GBM is unique compared to other decision tree algorithms because it builds models sequentially with higher weights given to those cases that were poorly predicted in previous models, thus improving accuracy incrementally instead of simply taking an average of all models like a random forest algorithm would. For this reason, this algorithm gives us a model with the highest accuracy of 71%

Results



Conclusions

Here, we can conclude that we can solve the problem of predicting whether the customer will default or not, more accurately, by implementing the Gradient Boosting model in training our data.

References

1. Dataset link: https://www.kaggle.com/zaurbegiev/my-dataset
2. Plots: https://www.rdocumentation.org/packages/ggplot2/versions/3.3.0/topics/ggplot
3. https://amunategui.github.io/dummyVar-Walkthrough/
4. https://www.thoughtco.com/what-is-the-interquartile-range-rule-3126244