Professor Mingdi Xin

BANA 273 Machine Learning for Analytics

30 November 2021

**Machine Learning Final Project Report**

**“Classifying Credible Loan Applicants”**

**Team 23:**

Kate Zagrebneva

Prashasti Sharma

My Han Mai

Noor Zia

Samira Shafiei

**Table of Contents**

[**Executive Summary**](#_yc7v76q1xfpi) **1**

[**Business Idea**](#_aonoaard5p41) **2**

[**Data Summary**](#_m99147837eny) **3**

[**Data Preprocessing**](#_sfslkz32eefo) **9**

[**Selected Machine Learning Techniques**](#_704bipbggbfb) **12**

[Naive Bayes](#_uu9qcgd5ebmg) 12

[Decision Tree](#_q3lug6keldvg) 25

[Random Forest](#_ojxrtx8a2sl3) 31

[**Analysing Results**](#_icypcdd4q0he) **37**

# 

# 

# **Executive Summary**

In this project we focused on analyzing loan approval data and figuring out which model to use in order to correctly classify people who are or are not likely to be approved for a loan. We also wanted to know what is the most important factor that influences applicant credibility.

We got the dataset that we used to build our models through Kaggle. It has twelve attributes and a loan approval class variable that is binary with values of Y and N. Deciding to capture the model result without preprocessing, we ran Naive Bayes, Decision Tree, Random Forest and Logistic Regression on WEKA to later compare the accuracy results and effectiveness of preprocessing. While the dataset was relatively clean, we still had to do some preprocessing before running all the models for optimal results. Preprocessing steps included replacing null values with mean/mode, encoding and normalizing data as well as removing outliers.

For each of the models out of Naive Bayes, Logistic Regression, Decision Tree and Random Forest we ran the model on slightly different preprocessed data to see which preprocessing steps work the best and identify the best fitting model for our dataset and project. Comparing the results, and especially focusing on the accuracy percentage of the models, we identify multiple models that worked well. Out of all of them, we looked into the differences and uniqueness of the models to see which model was best suited towards classifying our desired variable. The conclusion was made that the Logistic Regression model with outliers and no feature elimination or k-folds cross validation works best to identify who would get approved for the loan. Accuracy in descending order was Logistic Regression, Naive Bayes, Random Forest, and lastly, Decision Tree.

# 

# **Business Idea**

Classifying credible loan applicants is an important part of today's world, especially for banks, mortgage lenders, and other financial institutions. Loan officers require data analysis in order to determine which loan applications are safe and which pose a risk to their company profits. For example, the main portion of financial insitutions’s assets comes directly from profits earned from the loans it distributes.The primary objective in many financial environments is to invest their assets in situations where they can be assured of profit or being paid back. Therefore, it is extremely important to analyze and classify the applicant's information and assign them to one of two categories: applicant with good credit or applicant with bad credit.

Today many financial companies such as banks, mortgage providers, loan companies, et cetera approve loans after an aggressive process of verification and validation but there is no surety whether the chosen applicant is the most dependable applicant out of all available applicants. Through this project we hope to make more accurate predictions on whether or not a particular applicant is credible through the validation of attributes by applying machine learning techniques. Loan Prediction is very integral to the activities of lenders and also greatly impacts loan applicants.

The aim of this project is to provide a quick, immediate and easy way to identify and classify the credibility of the loan applicants who are most likely to get a loan approval. It can provide many benefits to the lenders and applicants who are looking to apply for loans. Through this project we can make decisions about whether certain attributes are affecting the credibility of the loan applicant. Other than enabling the lending officers to identify the best applicants, this project can also be useful to loan applicants. Identified impactful factors can be helpful for new customers to consider when they approach lending firms for a loan and can help them prepare and boost their potential as an applicant.

## 

# **Data Summary**

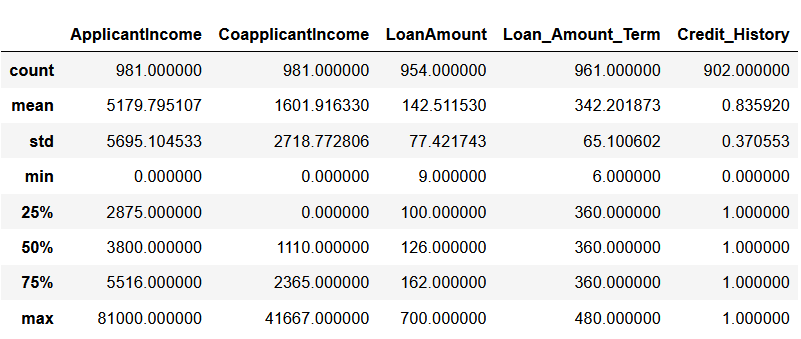
The data we are using for this project is pulled from Vikas Ukani’s ‘Loan Eligible Dataset’ off of the free datasets provided on Kaggle. The data is collected from a financial institution’s home loan application forms in order to verify the applicant’s eligibility to be approved for a loan.

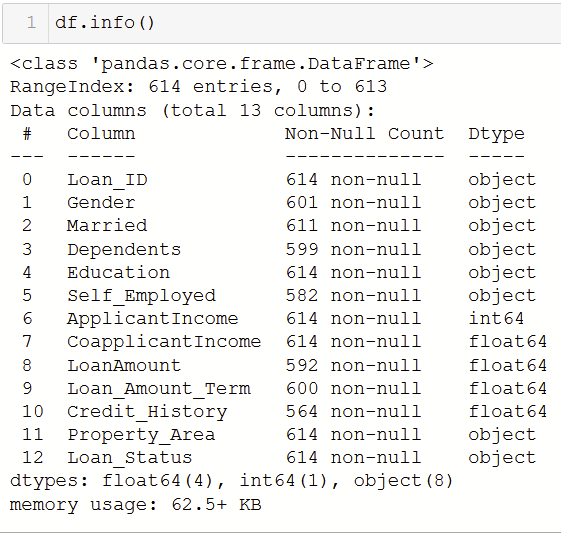
The application form requests information on gender, marriage status, dependents, education level, status of employment (self-employed or not), income level, co-applicant income level, amount of loan requested, length of loan, credit history of applicant, and area in which the property is located (urban, semi-urban, rural). It also provides applicants with a unique ID number that corresponds with the loan application. The loan amount is in units of thousands of US dollars and the loan length is in units of months.

The overall data set consists of 614 customers. The goal of using this dataset is to quickly classify customers’ loan credibility to save company resources by having a reliable applicant selection model to prevent any losses from approving the wrong applicants.

Variables:

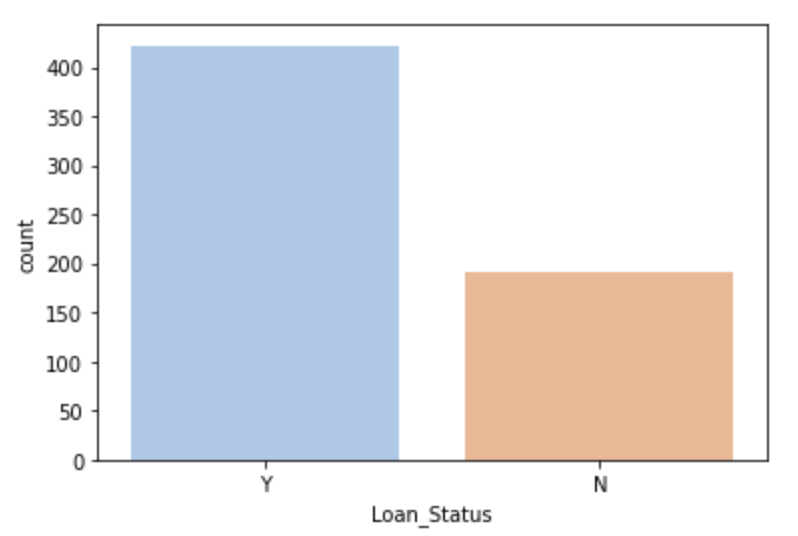
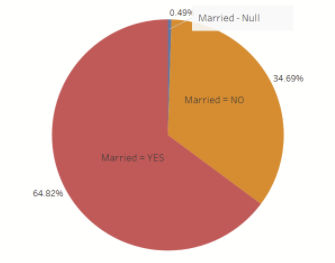
| Loan ID | Applicant Income |
| --- | --- |
| Gender (F/M) | Co-Applicant Income |
| Married (Y/N) | Loan Amount |
| Dependents (0,1,2,..) | Loan Amount - Term (12- 480) |
| Education (Graduate/Not Graduate) | Credit History (Past loan history: 0,1,..) |
| Self-Employed (Y/N) | Property Area (Urban/Semi Urban) |

Descriptive statistics of the numerical variables within the dataset:  


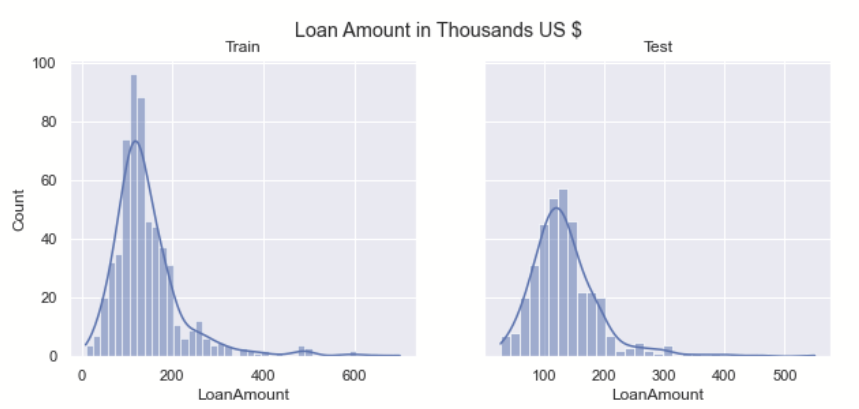
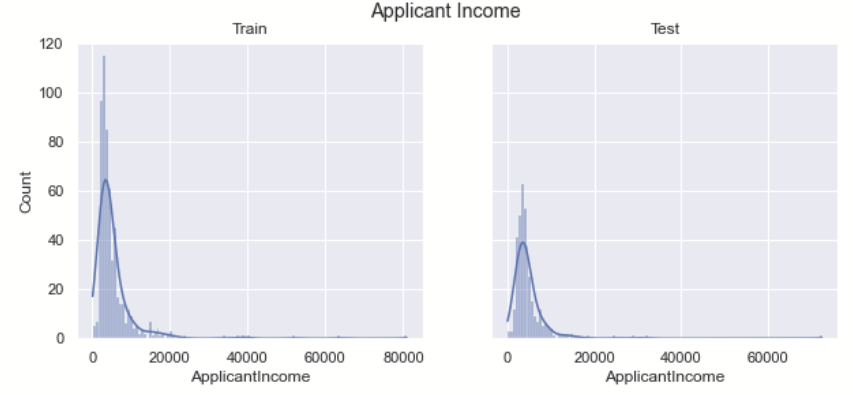
Data types of the attributes are as follows:  


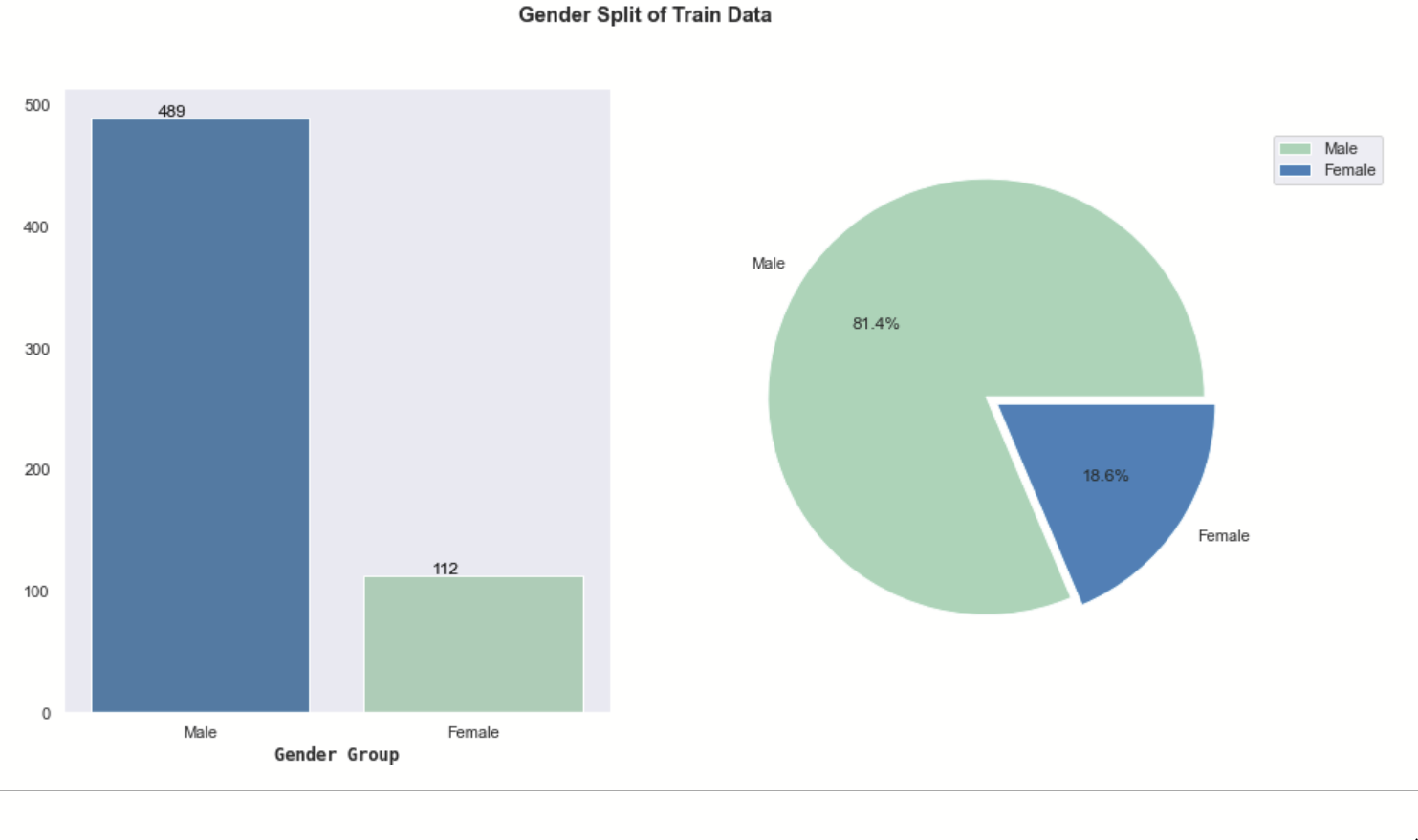
**Visualization of Un-preprocessed Dataset**

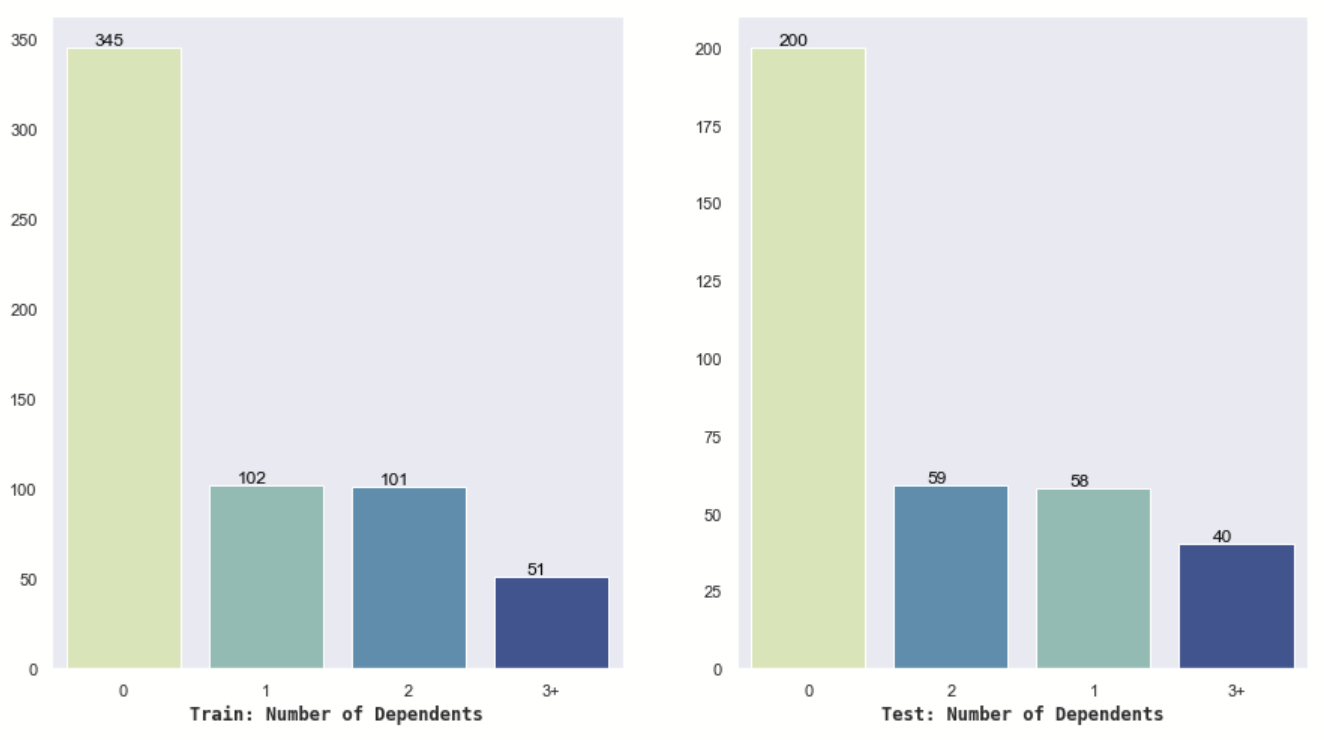
**Loan Status Marital Status**

**** 

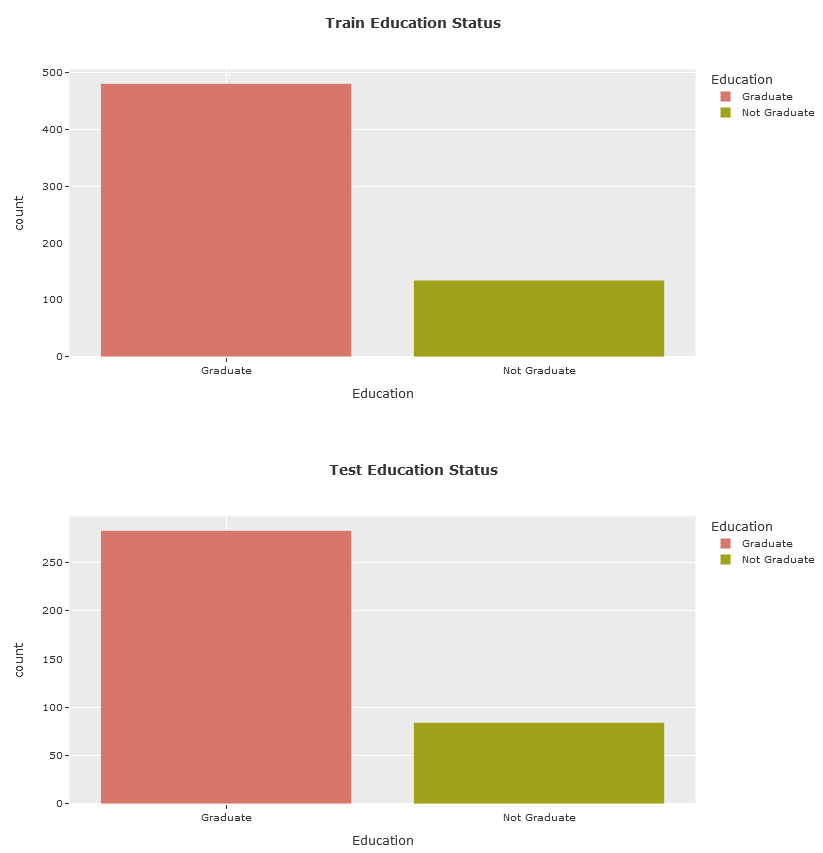
**Loan Amount Applicant Income**

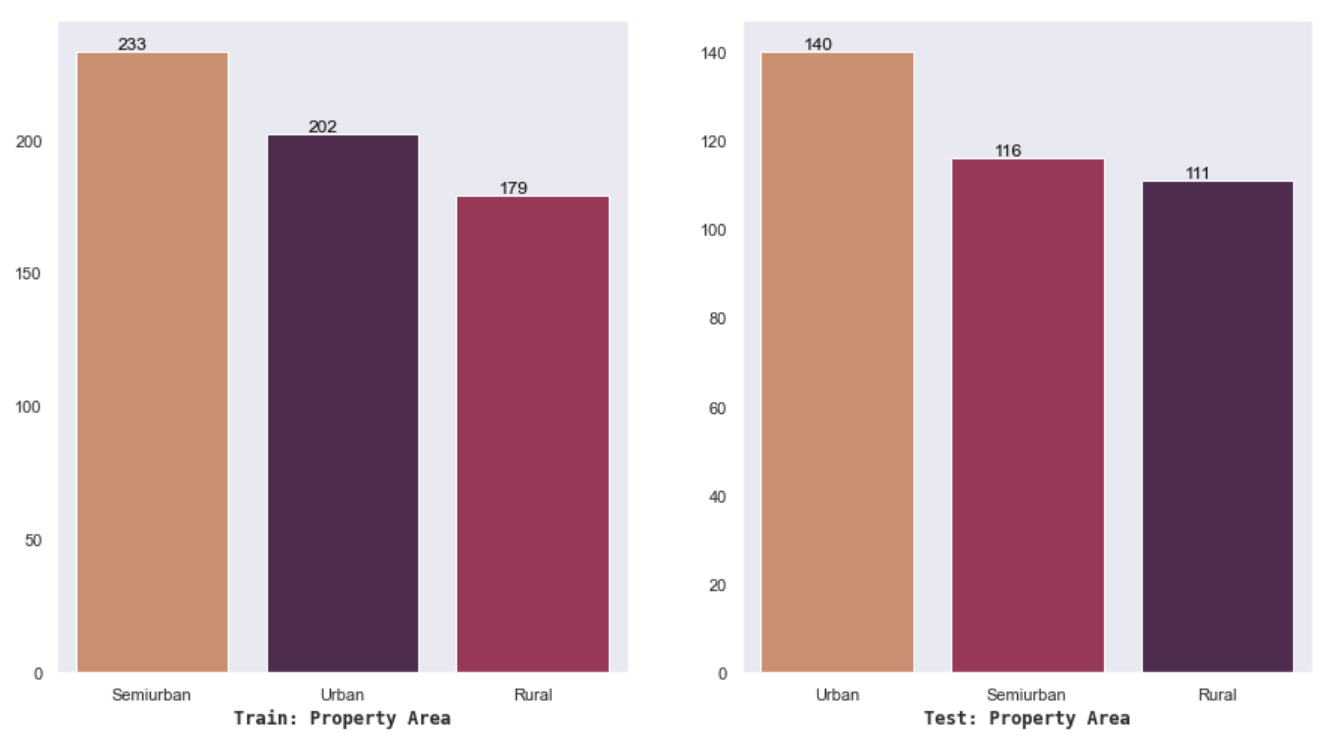


**Gender Dependents**



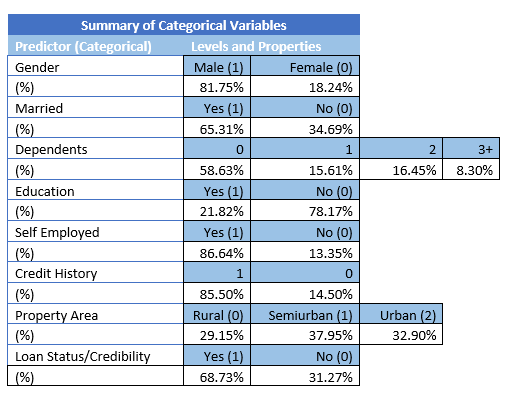
**Education**



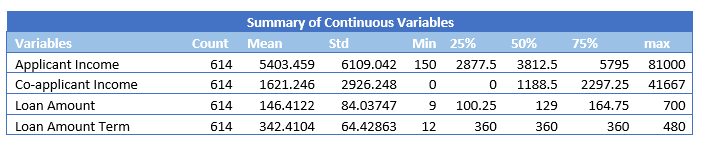
**Property Area**

**Main Observations of Data:**

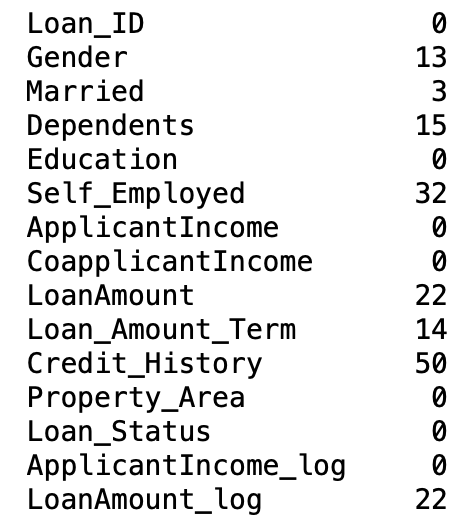
1. A number of applicants are without a credit history.
2. There are a number of null values indicating that quite a few applicants submitted an application without completing all the required information.
3. The variables, Loan Amount and Income, have outliers that skews the entire dataset to the right.
4. The general applicant type is a married, male, college-graduate with no children.

**Summary of Categorical variables:**

**Summary of Continuous variables:**



**Number of nulls in each column:**



# **Data Preprocessing**

Mainly four preliminary data processing steps were undertaken for cleaning and correcting the data for the purpose of analysis. These remained common throughout all the models:

1. Checking for duplicate instances
2. Dropping Loan\_ID column  
   Since the Loan\_ID is a unique consecutive number for each loan and doesn’t bring us any useful information, we decided to drop the column. Including this attribute would only make the model performance worse regardless of the type of the model.
3. Handling missing values  
   Since our dataset was small in size, we decided to keep as many data points as possible. Instead of removing the instances that had missing values, we used mean values as replacement for continuous variables and mode values as replacement for categorical variables
4. Label Encoding - Conversion of categorical data into numerical data

An essential part of our project was to understand the impact of outliers on the results. Our goal was to check how the treatment of outliers would affect the model’s accuracy. For treatment of outliers, we used two different techniques:

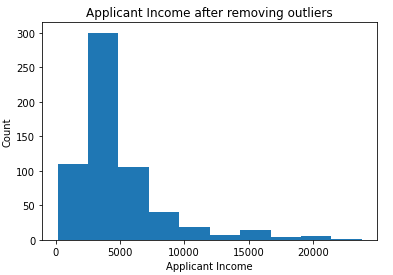
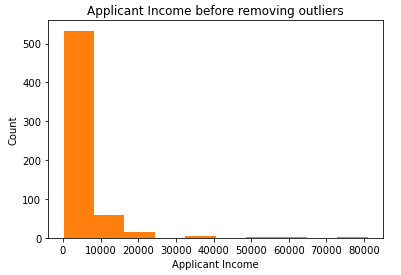
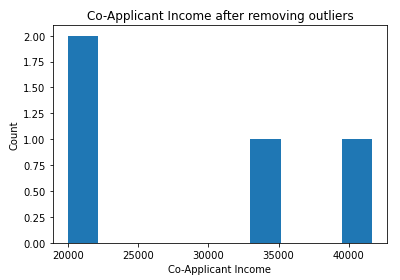
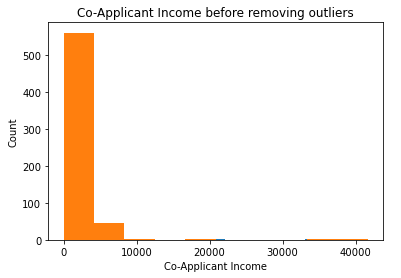
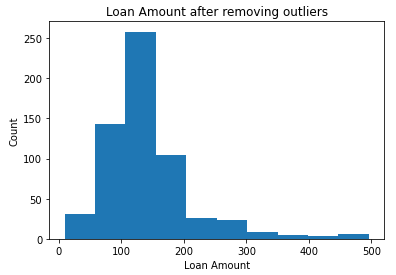
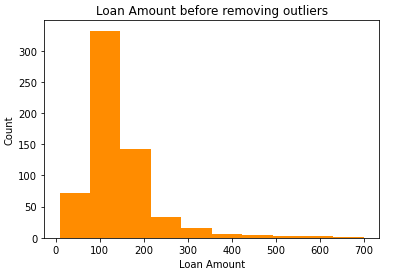
1. Identifying outlying customer data points using binning and dropping the extreme instances
2. Performing normalization using logarithmic function on skewed variables

For all the four types of modelling (Naive Bayes, Logistic Regression, Decision Tree, Random Forest), we have segmented analysis into 3 subparts -

1. With outliers
2. Without outliers
3. Using binning - dropping outlying data points
4. Using normalization - logarithmic function

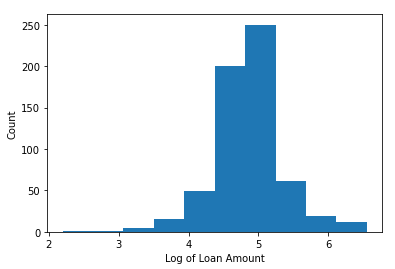
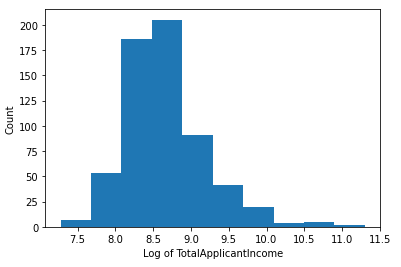
For the first subpart, the data was untreated for outliers and was used for analysis after performing the common pre-processing steps.

For the second subpart, we first visualized all the columns and saw that three columns - ApplicantIncome, CoApplicantIncome and Loan\_Amount - are skewed to the right and have maximum outliers. As part of exploration, we ran the model with different numbers of bins. Finally, we found the optimal number of the bins and removed the bins that had caused the data to skew.

1. Remove outlier customers with outlying ApplicantIncome (above $30000) - count 7  
   
2. Remove outlier customers with outlying CoaaplicantIncome (above $15000) - count 4  
   
3. Remove outlier customers with outlying loan amount (above $500) - count 5  
   

In total, there were a count of 14 unique customer entries that were outliers. We decided to drop these outlying instances.

For the final subpart, the second technique used to deal with outliers was normalization using logarithmic function. Again, we intended to perform normalization on the three continuous variable columns discussed previously - ApplicantIncome, CoApplicantIncome and Loan\_Amount. The challenge faced here was that a large number of customers had CoApplicantIncome stated as zero. This prevented us from using the log function directly on the variable. To address the challenge, we decided to introduce feature reduction and created a new variable named TotalIncome which was the sum of Applicant Income and CoApplicantIncome (TotalIncome = ApplicantIncome + CoApplicantIncome), then took a log on the new variable to address the skewed distribution.



Since we are dealing with classification problem i.e the dependent variable is a binary Y/N variable, we decided to work with the below four models -

1. Naive Bayes
2. Logistic Regression
3. Decision Tree
4. Random Forest

Within these models, we introduced different data preprocessing techniques to test the accuracy of each model and analysed their impact on the model variations. These were -

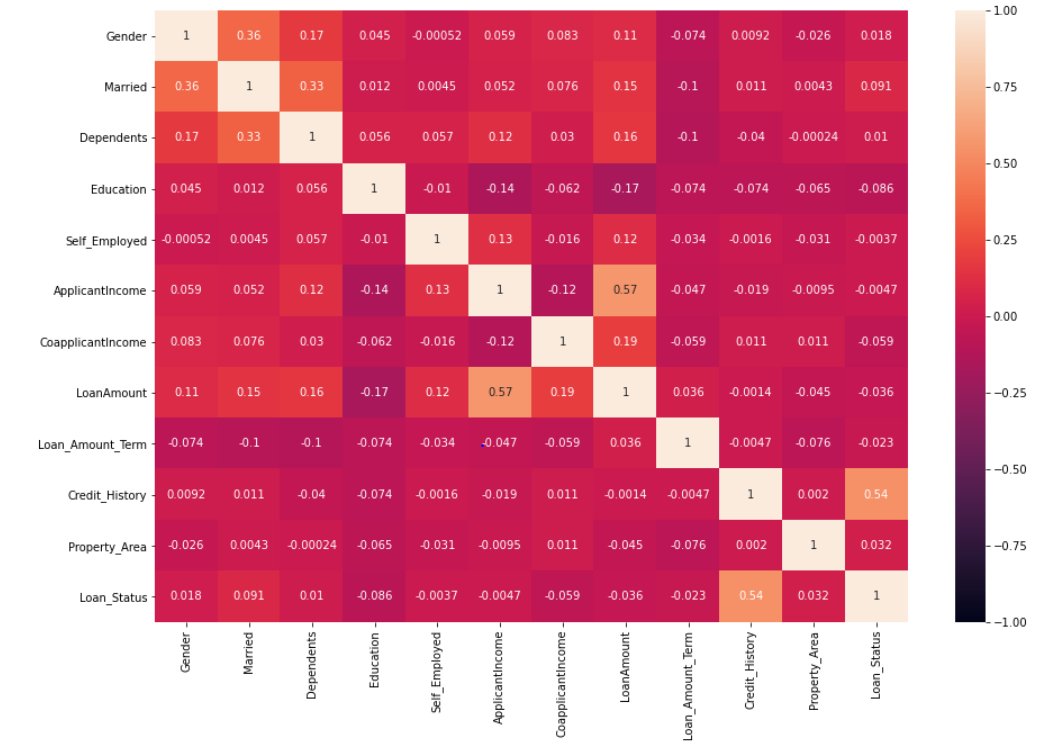
1. Naive Bayes - Feature Scaling
2. Logistic Regression - Recursive Feature Elimination, Stratified K-Fold Cross Validation
3. Decision Tree - Feature selection using Gini Impurity Index
4. Random Forest - Stratified K-Fold Cross Validation

# **Selected Machine Learning Techniques**

## **Naive Bayes**

Naive Bayes is a probabilistic classifier and is one of the most powerful and simple classification techniques to make fast predictions. Naive Bayes performs better than other classifiers when the variables are independent. The classifier works under the assumption that the predictor variables are unrelated to each other in the dataset. To make sure we have chosen the right technique and to have a robust analysis we examined the correlation between the variables in the dataset.

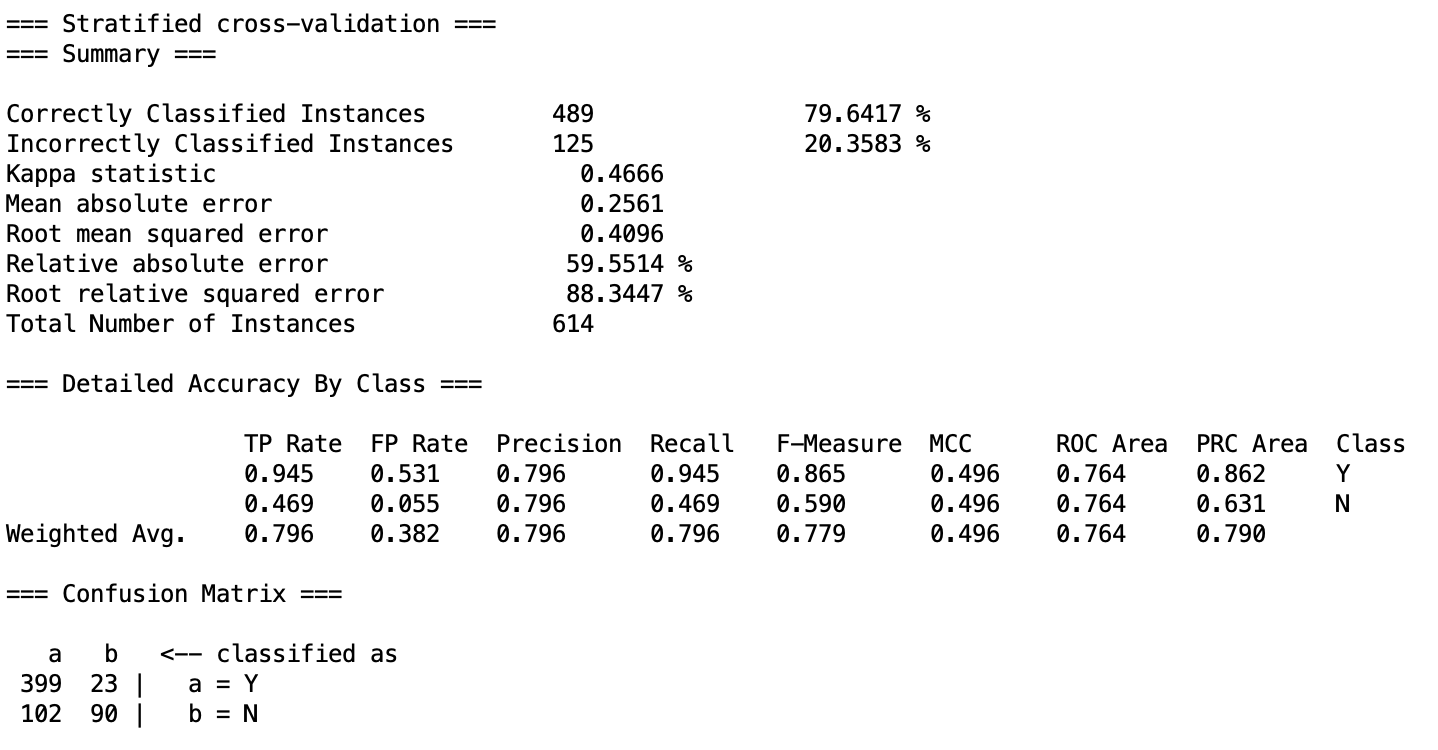
We analyzed the correlations between variables in Python and also created a heatmap visualization to be able to explain the correlation between the attributes. According to the below plot, the dataset’s attributes have either very weak correlations or almost no correlation between themselves. So, the first assumption needed to use the Naive Bayes technique is fulfilled, which is that the predictors are independent.



Moreover, the provided dataset is structured and supervised, and some of the attributes are categorical. We chose the Naive Bayes technique because it works better with categorical variables compared to numerical variables. Subsequently, we changed the numerical variables to categorical to improve the model’s accuracy

**Weka Implementation:**

To start, we used Weka to run an initial analysis on the raw dataset. The objective was to get a quick understanding of the dataset and the model accuracy. Before running Naive Bayes classifier, we removed the irrelevant variable - Loan\_ID.



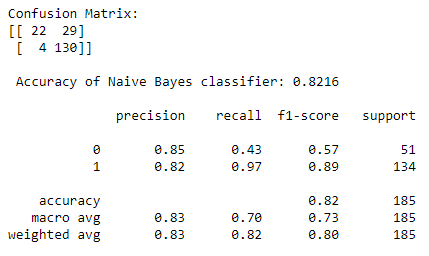
Even though the accuracy of Naive Bayes model without any pre-processing came out to be 79.64%, we can infer from the confusion matrix that the model doesn’t do a great job of classifying loan applicants as credible or not.

Our next step was to run analysis on Python since it lets us, while performing the modelling, to introduce different techniques, cross-validate at every stage, and build aesthetic visualizations.

**Python Implementation:**

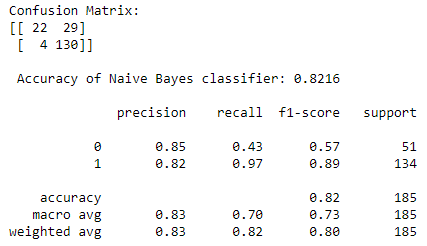
There were a total of six Naive Bayes models that we ran in order to test which preprocessing steps would work best for the model and improve its accuracy.

To begin, we first ran the pre-processed data without removing outliers and without feature scaling on Naive Bayes. Results below:



We can see that the model does a decent job of correctly classifying the customers to whom loan should not be provided (i.e. Loan\_Status = N) and gives an accuracy of 82.16%.

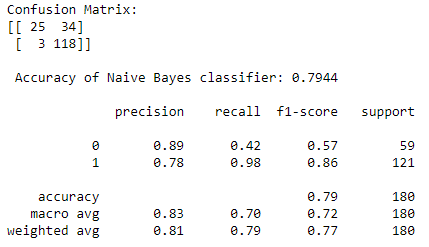
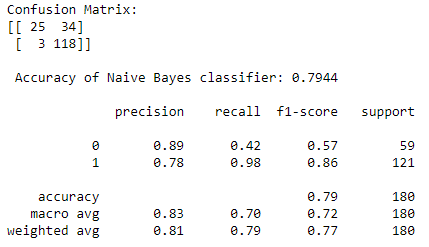
Next, we wanted to check whether feature scaling had any significant impact on the model accuracy. Next we decided to introduce feature scaling in the model and check if the results varied.



The results remained unchanged. We can imply that feature scaling had no effect on the dataset and model accuracy.

Afterward, we decided to treat outliers in the dataset. Using the binning method, we decided to drop the outliers and ran the model twice - with and without feature scaling. The results were as follows:

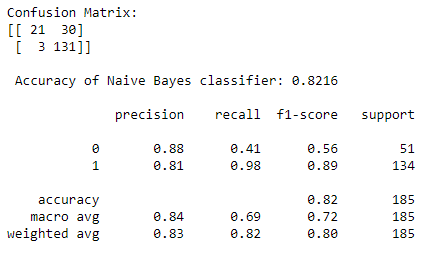
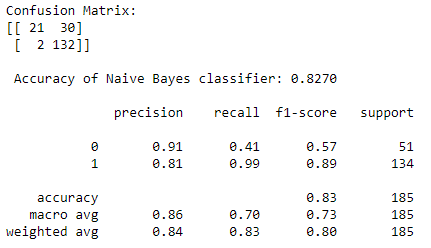
Without feature scaling: With feature scaling:



With the above results, we can easily conclude that feature scaling had no impact on the accuracy of the Naive Bayes model without outliers (using binning technique).

The second technique used to deal with outliers was normalization using logarithmic function. Post normalization, the Naive Bayes model with and without feature scaling gave the below results:

Without feature scaling: With feature scaling:



The model’s accuracy reduced from 82.70% to 82.16% upon introducing feature scaling. We concluded that feature scaling was ineffective since none of the three modeling variations showed any improvements. The overall accuracy improved up to 82.70% after removing outliers and normalizing the skewed variables using the logarithmic function.

Hence, the best performing model out of the six Naive Bayes models was the Naive Bayes model without outliers by normalization using log function.

| **Method** | **Accuracy** |
| --- | --- |
| Naïve Bayes before any preprocessing (Weka) | 79.64% |
| Naïve Bayes with Outliers (without Feature Scaling) | 82.16% |
| Naïve Bayes with Outliers (with Feature Scaling) | 82.16% |
| Naïve Bayes without Outliers (Binning) | 79.44% |
| Naïve Bayes without Outliers (Binning + Feature Scaling) | 79.44% |
| Naïve Bayes without Outliers (Normalization using log function) | 82.70% |
| Naïve Bayes without Outliers (Normalization using log + Feature Scaling) | 82.16% |

## 

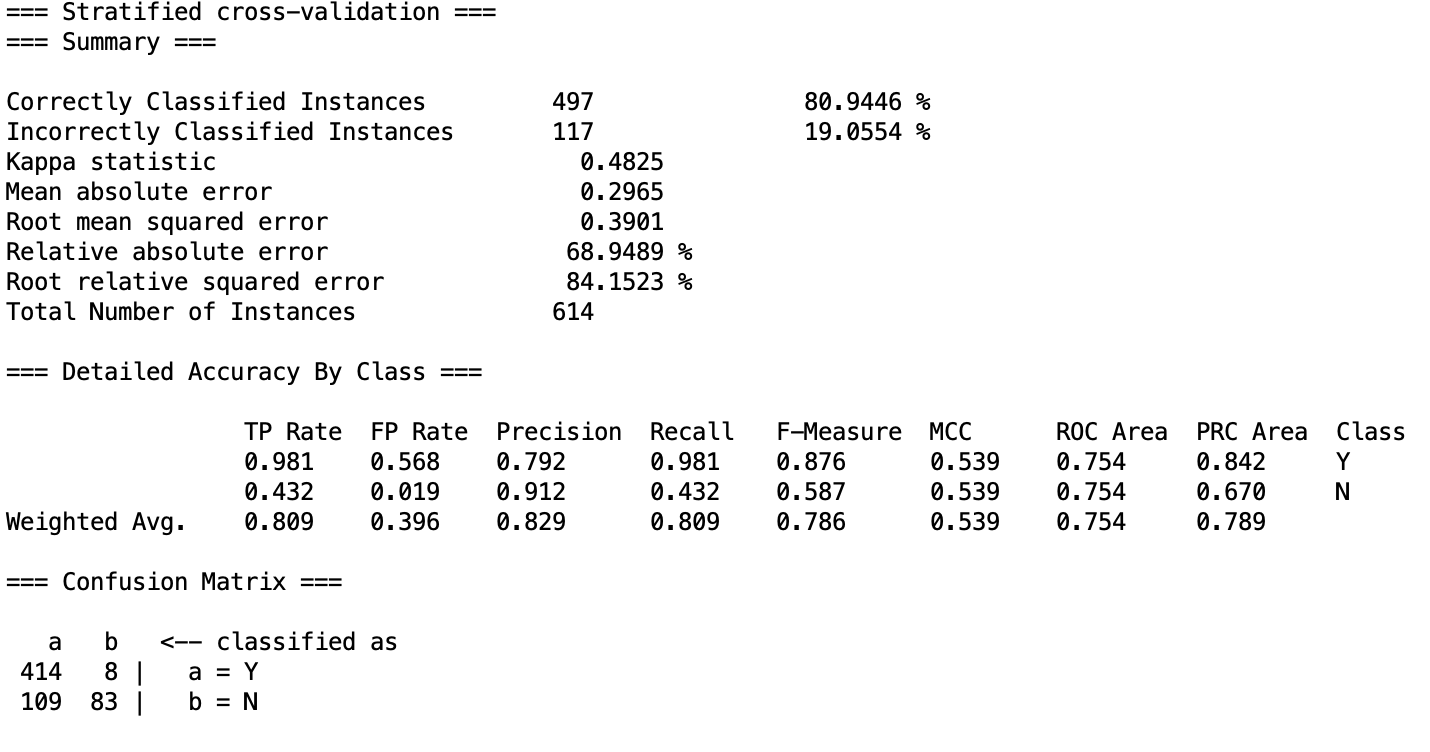
**Logistic Regression**

Another classical model used in classification problems is Logistic Regression. We chose this technique because Logistic Regression is a robust model to predict binary variables. In the dataset, the dependent variable is loan approval and it conveniently is binary (approve: yes =1, approve: No=0). The model is an estimation of logit function and is simply a log of odds in favour of the event.

Furthermore, Another assumption of the logistic regression which is made by the provided dataset is that independent variables should not be correlated. As mentioned in the Naive Bayes technique, we proved that the attributes have practically no correlation by conducting a heatmap plot (see above). In order to have a more robust model and increase the overall accuracy, we used Recursive Feature Elimination for removing the least significant features. Moreover, since our dataset is small, we incorporated K-fold cross validation to improve the robustness of our results.

**Weka Implementation:**

To assess unprocessed data, we first ran Logistic Regression in Weka. We removed LoanID and Dependents. We used ‘Logistic’ and cross-validation with 10 folds and results are as follows.

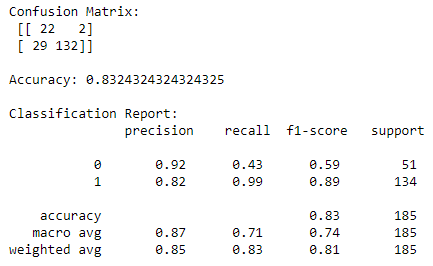


Even though the accuracy turned out to be 80.94%, the confusion matrix is a good indicator of the model’s inability to correctly classify loan applicants as credible or not credible for loan.

**Python Implementation:**

For logistic regression, we added one more testing measure compared to Naive Bayes models. In addition to testing how the models perform with and without outliers by performing the two different techniques of removing the outliers by binning and running normalization using the log function, we introduced Recursive Feature Elimination technique to check if dropping of least significant variables would improve the model’s performance.

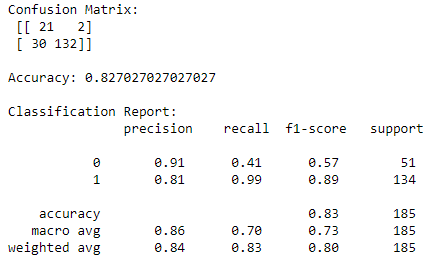
First, we run a model with basic pre-processing but without addressing the outliers. The preprocessing includes treatment of null values, dropping irrelevant columns (such as Loan\_ID), and encoding categorical to numerical variables. The results are as follows -



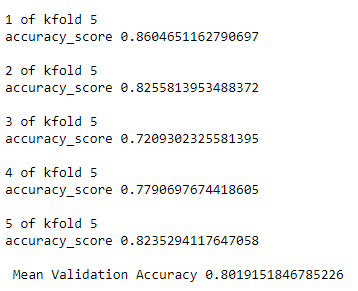
Next, we incorporated Recursive Feature Elimination (RFE) in the above model. We set the n\_features\_to\_select variable as 8 out 11 features and the below variables were ranked as least significant -



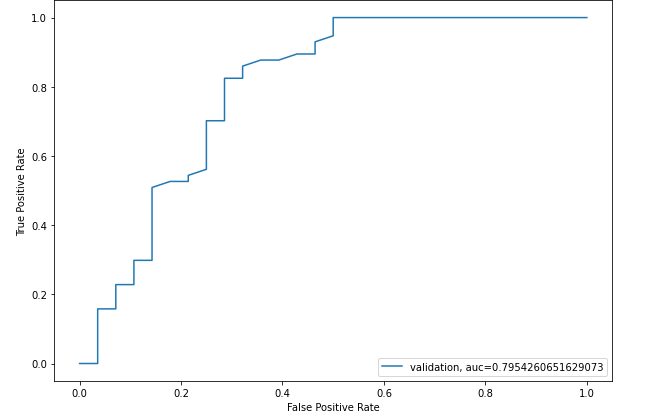
After dropping the above variables, the Logistic Regression model gave the below results -



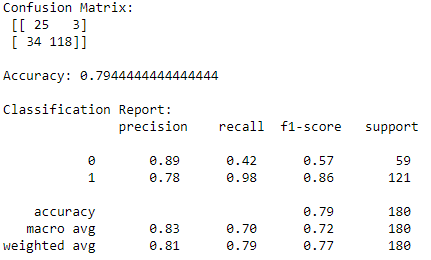
Next we introduce the K-fold cross validation in the above model to address the limitations due to a small dataset. We use Stratified K-Fold validation technique to ensure that each fold is a good representative of the whole. Here, the K is set as 5.



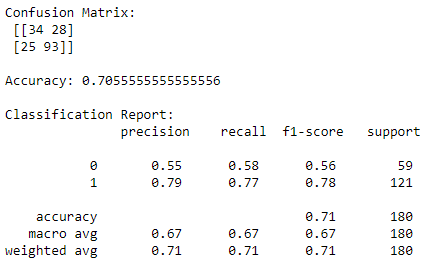
Plotting the AUC graph - with AUC value as 0.7954



Second, we attempt to remove the outliers and address skewed variables by using binning technique and by dropping the outlying customer entries (14) as described previously under Naive Bayes. Without RFE and K-Fold cross validation, the pre-processed data gave the below results -

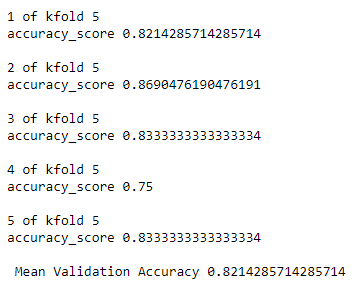


After Recursive Feature Elimination -

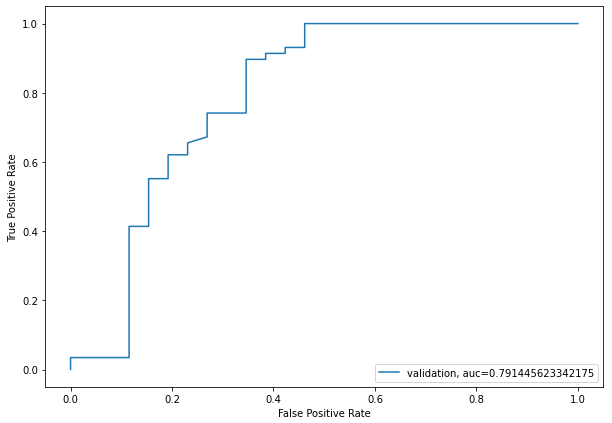


3 out of 11 features were dropped. The overall accuracy of the model was impacted in this case. Accuracy dropped from 79.44% to 70.55%.

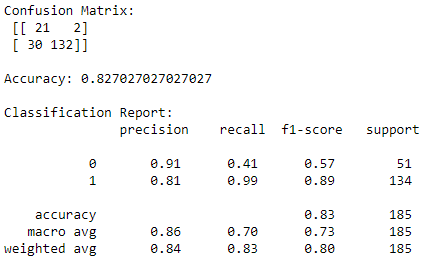
Introducing K-fold cross validation -



The overall accuracy of the model was significantly improved. The AUC plot showed AUC value as 0.7914



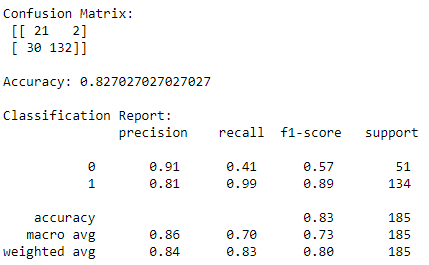
Lastly, we removed outliers using the second technique of normalization. Running the pre-processed data without RFE and without K-fold cross validation gave the below results -



Since the ApplicantIncome and CoApplicantIncome were combined to form a new variable TotalApplicantIncome, the Recursive Feature Elimination results gave two new least significant variables, i.e. -

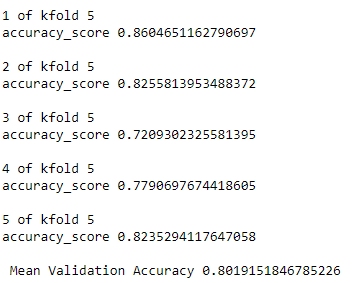


After removing the above two variables, the results were as follows -

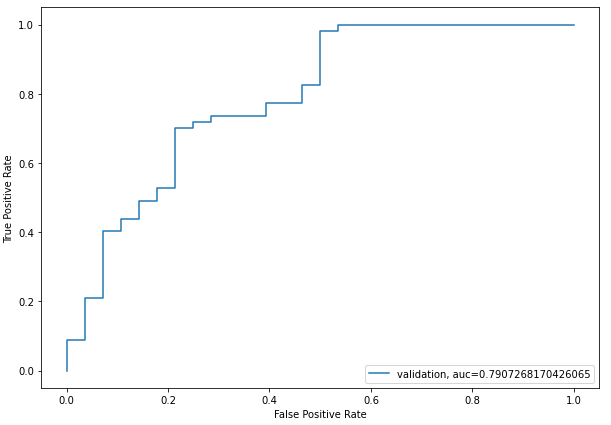


The results remained unchanged. Hence, RFE had no impact in this case.

Next, we included K-fold cross validation with K =5.



AUC plot -



Overall, the results showed that the Logistic Regression with Outliers performed the best out of the ten model variations with accuracy of 83.24%. We can infer that the outliers in the data helped the Logistic Regression model predict an accurate output. Another important thing to note is that Recursive Feature Elimination did not improve results in any of the three model variations (i.e. with outliers, without outliers (binning), without outliers (normalization)). Using this information, we can safely conclude that out of the limited features (11) in our data, all the variables were significantly contributing to the results and feature elimination didn’t play a significant role in improving the model accuracy. Lastly, the use of K-fold cross validation technique showed model improvement in only 1 out of the 3 cases (i.e. model without outliers (binning))

| **Method** | **Accuracy** |
| --- | --- |
| Logistic Regression before Preprocessing (Weka) | 80.94% |
| Logistic Regression with Outliers | 83.24% |
| Logistic Regression with Outliers + RFE | 82.70% |
| Logistic Regression with Outliers + RFE + K-fold Cross Validation | 80.19% |
| Logistic Regression without Outliers using Binning | 79.44% |
| Logistic Regression without Outliers using Binning + RFE | 70.55% |
| Logistic Regression without Outliers using Binning + RFE + K-fold CV | 82.14% |
| Logistic Regression without Outliers by normalization using log function | 82.70% |
| Logistic Regression without Outliers (normalization) + RFE | 82.70% |
| Logistic Regression without Outliers (normalization) + RFE + K-fold CV | 80.19% |

## **Decision Tree**

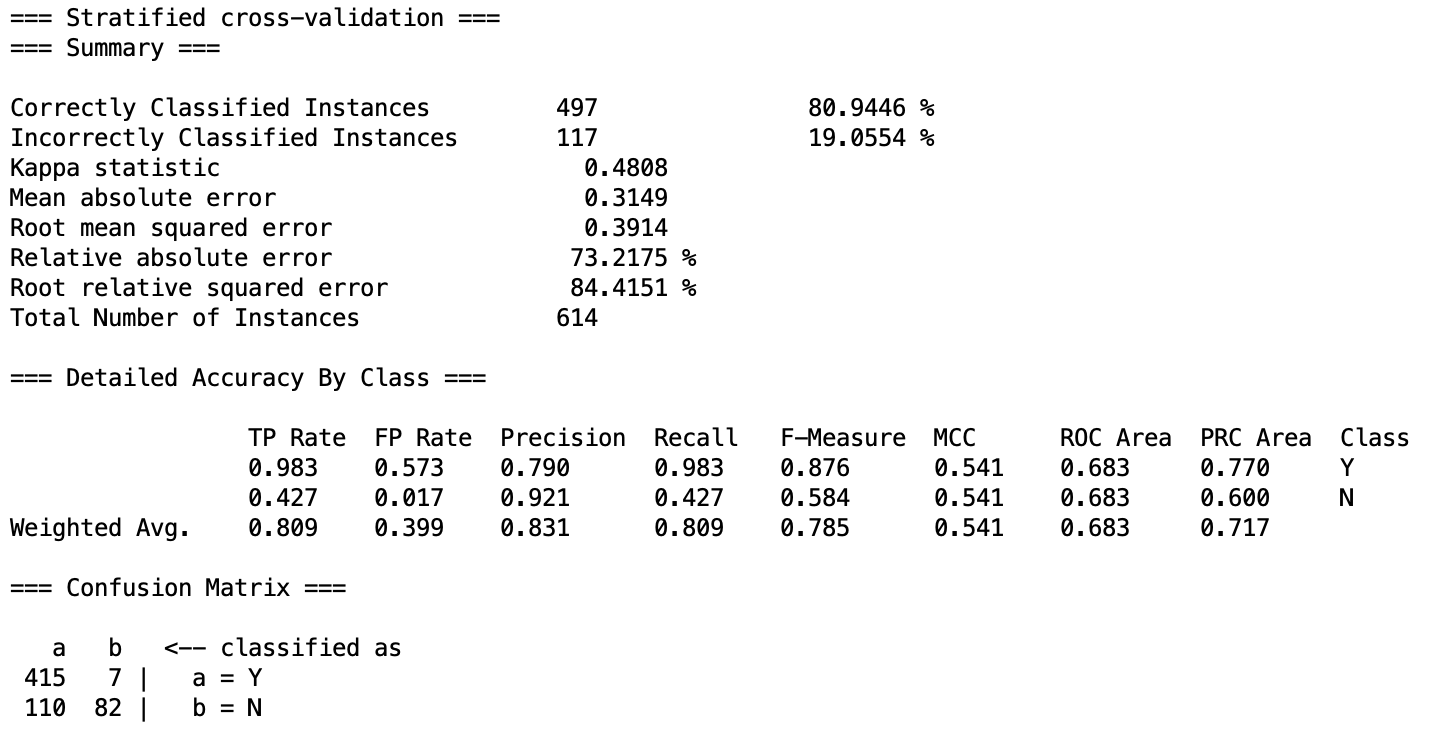
Decision Tree is one of the simplest ways to perform classification. Similar to the branches of an actual tree, the decision tree model splits the dataset into parts based on given attributes at each level. The objective of our project is to determine whether the customer with certain characteristics will qualify for a loan. For example, a customer provides information such as Gender, Marital Status, Education, etc. on a loan application form and based on the provided parameters, we decide to predict whether the loan application will be approved or not.

One of the reasons that we chose the Decision Tree technique is that our dataset contains both categorical and continuous variables and the Decision Tree algorithm can utilize both data types. Another reason is that it is easy to understand and interpret without requiring very complex and precise data. Also, Decision Tree does not require preprocessing to get output.

A key challenge with the Decision Tree algorithm is selecting the right attributes. We need to choose the most informative attributes as the root node at the different levels of the decision Tree. In order to tackle this problem, we planned to measure the information gain and entropy for each variable. However, we ultimately decided to use Gini Impurity to decide which attribute was most impactful. We set the chosen attribute as the root of the decision Tree and split the data to get the best classification and purer classes.

**Weka implementation:**

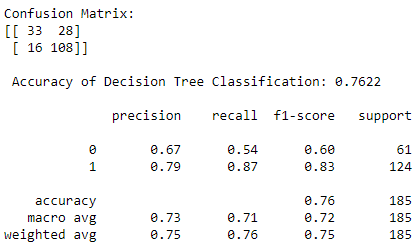
Firstly, we ran an initial un-preprocessed analysis on Weka after dropping Loan\_ID column and results are as follows:



The confusion matrix in the above result clearly shows that the model with unprocessed data isn’t doing a good job of classifying credible loan applicants.

**Python implementation:**

Next, we moved to analyzing the model and running different enhancements in Python. We ran the model after performing basic pre-processing steps - without treating for outliers and without feature scaling. The results are as follows:



As an attempt to improve model accuracy and prevent model overfitting, we tried implementing feature selection. We measure the gini impurity index of each attribute to decide which variables (with the highest gini impurity) can be dropped while performing further analysis and which variables (with lowest gini impurity) can be used to split at the parent node. We chose to work with Gini Index instead of Information Gain or entropy because calculating entropy involves use of logarithmic function and for variables with high number of zeroes it would be a difficult task to compute. Moreover, calculating the Gini Index is computationally less intensive.

Gini Impurity Index is given by -

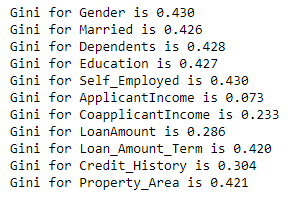


where Pi denotes the probability of an element being classified for a distinct class

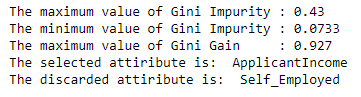
Gini Gain is given by -

****

Gini impurity index for the attributes are -

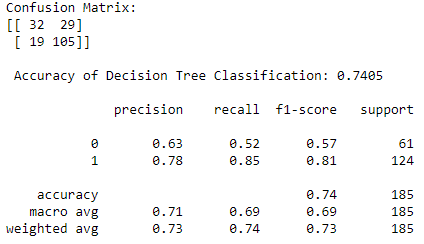
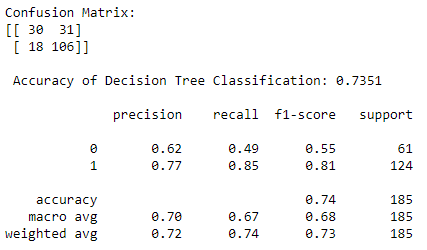


The attribute with the highest Gini Impurity is chosen to be discarded. Those with lower Gini impurity have purer classes and are selected as the parent node to split further branches.

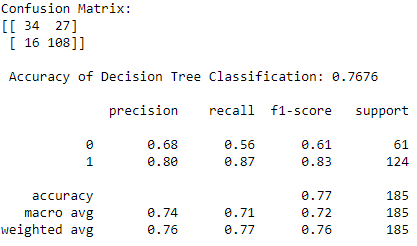
****

The attributes with the highest Gini Impurity are Self\_Employed and Gender with values of 0.430. As part of feature selection, we discard both variables and then run the analysis with the remaining attributes to see if there has been further improvements.

After removing ‘Gender’ attribute - After removing ‘Self\_Employed’ attribute -

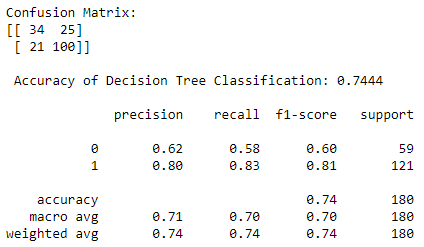


Results after removing both ‘Gender’ and ‘Self\_Employed’ attribute -



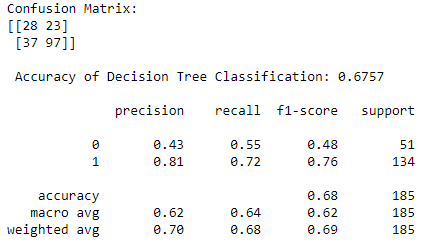
We observed that, while dropping only one variable did not improve accuracy, dropping both the variables with highest Gini Index caused the overall accuracy of the model to increase slightly from 76.22% to 76.76%. Therefore, we chose to proceed with including feature selection while performing additional pre-processing steps such as treating outliers.

The results after removing outliers using binning technique are as follows -



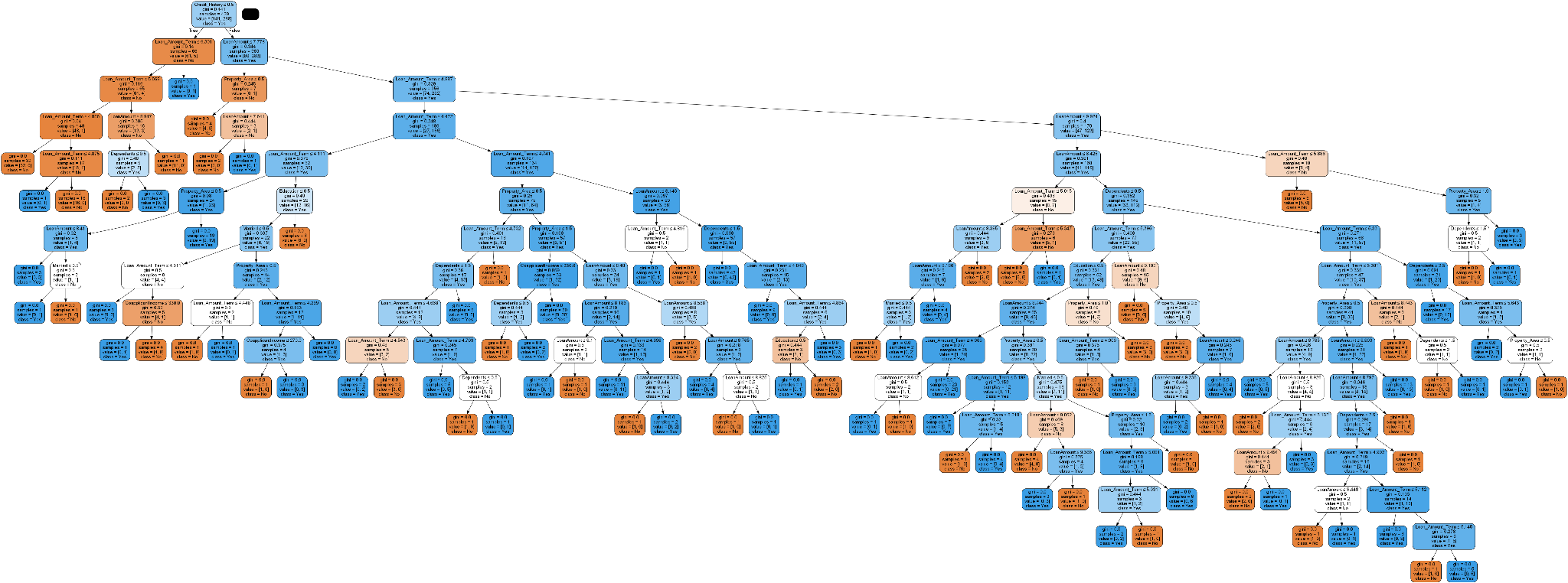
The accuracy of the model dropped from 76.76% (post feature selection) to 74.44% (after removing outliers using the binning method).

Next, we remove outliers using normalization and get the following results:



The accuracy dropped even further compared to the first technique of treating outliers. From this, we conclude that the Decision Tree model does a better job of classification of variable importance but not so when accounting for outliers.

**Decision Tree Visualization**



For Better View of Decision Tree: <https://imgur.com/n6Vbcpg>

Out of the four variations of decision tree models, it can be concluded that the Decision Tree with outliers and with feature selection based on Gini Index resulted in the highest model accuracy of 76.76%. However, surprisingly, the Decision Tree model without any pre-processing that ran on Weka has the highest accuracy overall. Even though the results with Weka showed highest accuracy, the confusion matrix results were proof that the model wasn’t doing well in classifying credibility of loan applicants. We would consider the Decision Tree (After Feature Selection) model as the best of the lot because the dataset required basic pre-processing to be done (such as addressing nulls, label encoding, etc.) for the model to be accurately delivering results. Hence, for comparative purposes we would consider the Decision Tree (After Feature Selection) model as the best choice.

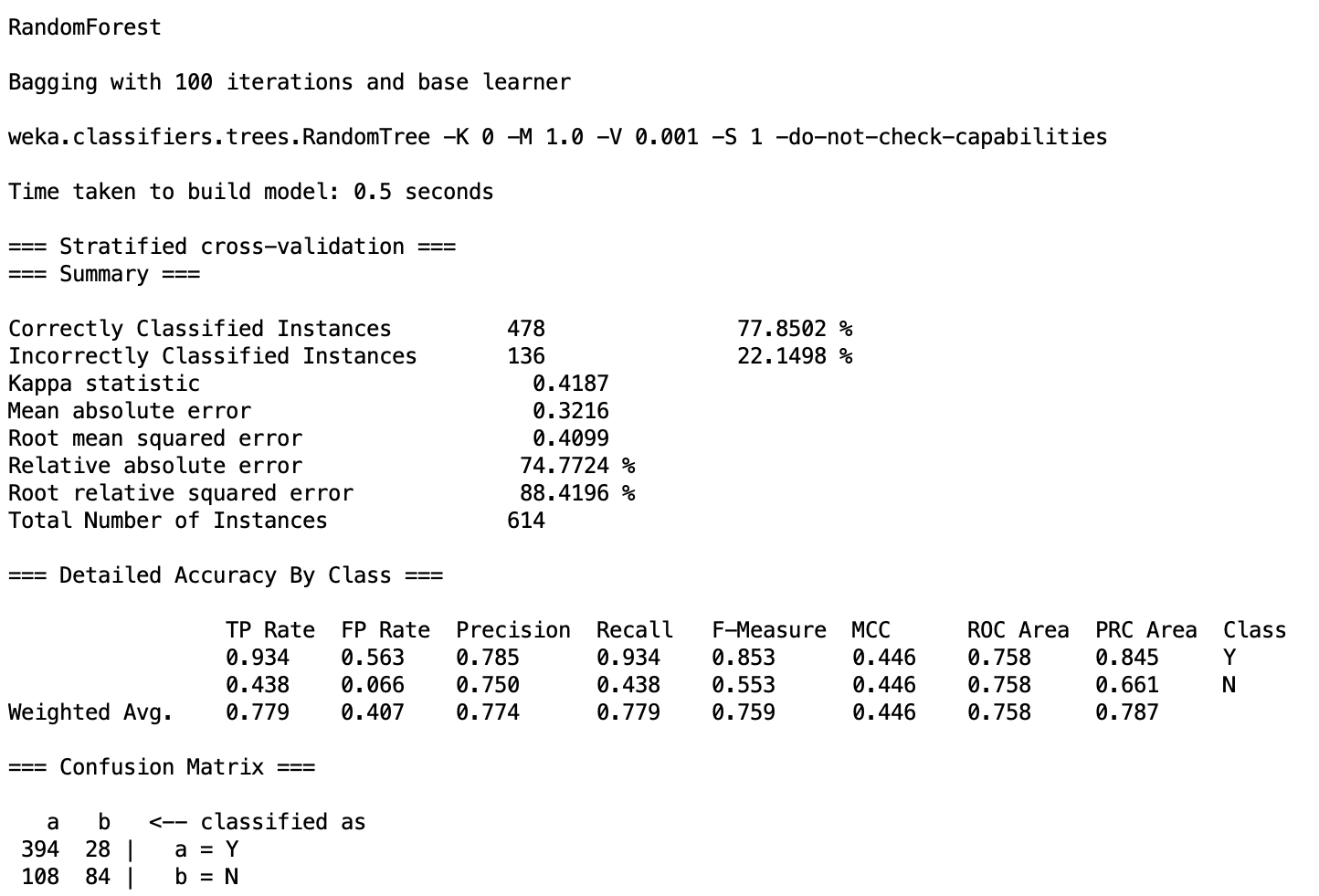
| **Method** | **Accuracy** |
| --- | --- |
| Decision Tree (No preprocessing - Weka) | 80.94% |
| Decision Tree (With Outliers) | 76.22% |
| Decision Tree (After Feature Selection) | 76.76% |
| Decision Tree (Without Outliers using binning) | 74.44% |
| Decision Tree (Without Outliers using normalization) | 67.57% |

## **Random Forest**

Random Forest Algorithm can solve both types of problems i.e. classification and regression. Random Forest is an advanced classification ensemble learning algorithm which combines several weak learners to make a powerful prediction model.

The Decision Tree is a simple classification model, however, in terms of prediction accuracy, it is not as robust as other models. We chose the Random Forest classifier to improve the prediction accuracy of the Decision Tree model. After growing some individual decision trees on the training dataset, the Random Forest model combines all the predictions and produces outcomes with a more reliable accuracy. We will also use ‘feature importance’ in Random Forest to find out the most important features.

**Weka Implementation:**

Running raw unprocessed data on Weka post removing irrelevant Loan\_ID column - 

Though the accuracy of the model is 77.85%, we can see the confusion matrix and infer that the model doesn’t do a good job of classifying loan applicants as loan approved : Yes or No.

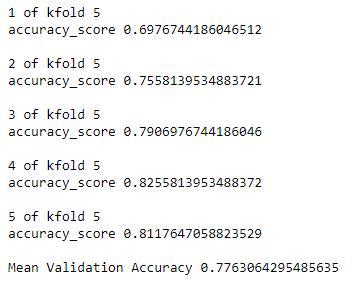
**Python Implementation:**

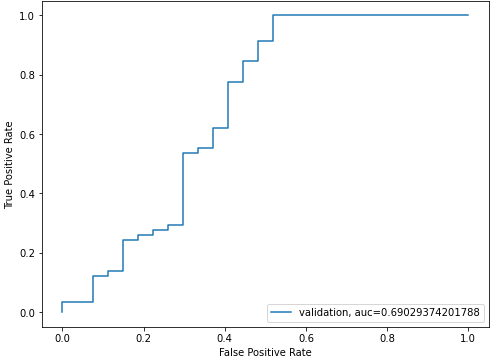
We will implement three variations of Random Forest modelling for comparative purposes -

1. With Outliers
2. Without Outliers using binning
3. Without Outliers using normalization

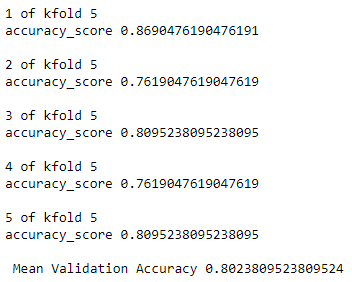
Due to the small size of the dataset, we performed K-fold Cross Validation. In each of the three variations we will implement K-Fold Cross Validation (Stratified K-Fold Cross Validation) to improve the accuracy and evaluate the results. The K value used is 5.

Firstly, model results for preprocessed data (removing duplicates, addressing null values, dropping irrelevant columns such as Loan\_ID and encoding categorical variables) are as follows:

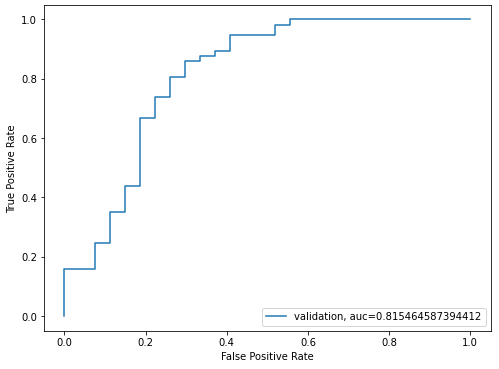


The accuracy of the K-fold Cross Validated Random Forest Classifier is 77.63%. The AUC curve is plotted below -   


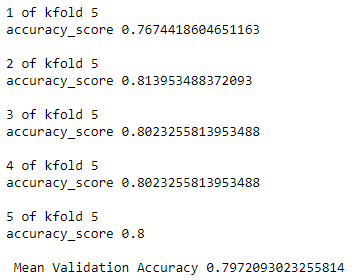
Moving forward we addressed outliers using binning and dropping outlying instances. Post implementing K-fold cross validation we get the below results -



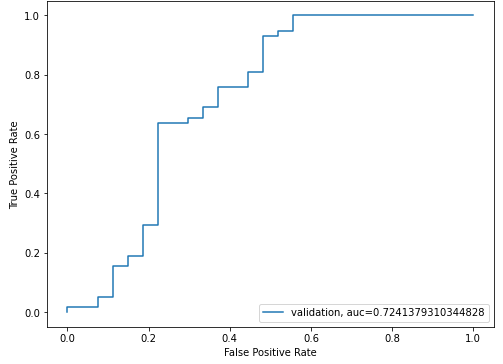
The accuracy of the model slightly improved from the previous scenario to 80.24%. The AUC plot is below -



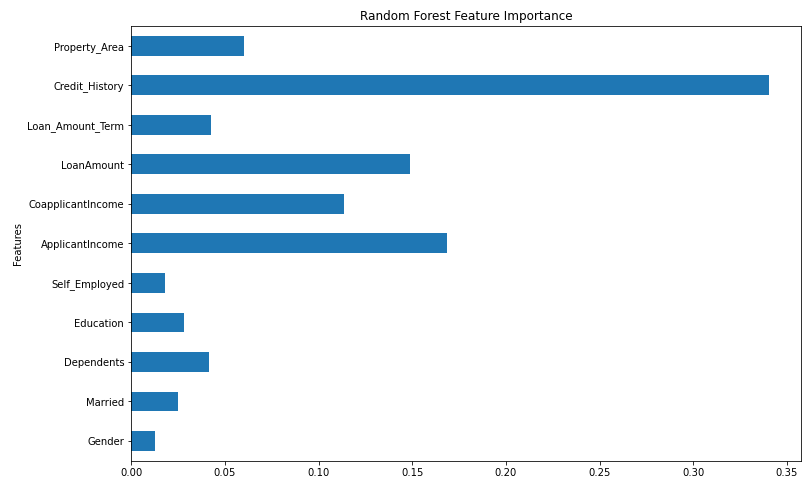
Lastly, the results for Random Forest model without outliers using normalization method and K-fold Cross Validation are presented below -



Although the accuracy dropped only by decimal point and is very close to the result given by the previous model, the AUC value is comparatively less.



Using the feature\_importance\_ of sklearn, we find out which attributes are the most important for our problem statement. The higher the feature importance value, the more important is the feature. The most important feature in determining the credibility of a loan applicant is Credit\_History. It is followed by Applicant’s Income and the Loan Amount that the candidate is seeking.



On comparing the different models, we can see both on the basis accuracy and AUC value that Random Forest Classification without Outliers does a good job of classification of credible loan applicants.

| **Method** | **Accuracy** | **AUC** |
| --- | --- | --- |
| Random Forest without Preprocessing (Weka) | 77.85% |  |
| Random Forest with Outliers | 77.63% | 0.690 |
| Random Forest without Outliers using binning | 80.23% | 0.815 |
| Random Forest without Outliers using normalization | 79.72% | 0.7241 |

# 

# 

# 

# 

# 

# 

# **Analysing Results**

We wanted to choose the best model to utilize for this dataset in order to identify who would get approved for loan while also identifying which factors are the most influential in making the approval decision. For evaluating the models, we used accuracy as a key measure. Occasionally, we have also utilised confusion matrix results as well as AUC values for analysis. All the models were built using python package sklearn.

Simply comparing the accuracy results of the top performing models in each of the Naive Bayes, Logistic Regression, Decision Tree and Random Forest models, we get the following results in the descending order:

| **Model** | **Accuracy** |
| --- | --- |
| Logistic Regression with Outliers (no RFE and no K-Fold CV) | 83.24% |
| Naïve Bayes without Outliers (Normalization using log function) | 82.70% |
| Random Forest without Outliers (binning) with K-Fold CV | 80.23% |
| Decision Tree (After Feature Selection) | 76.76% |

Comparing the highest accuracies of each of the classification models, we can safely conclude that our dataset works best with Logistic Regression. Since the dataset is smaller in size, the model didn’t require any feature elimination. Interestingly, unlike other models, Logistic Regression takes outliers into account to perform classification accurately.

As for Naive Bayes, the model works best without outliers (using normalization). The accuracy of the Naive Bayesian model is quite close to that of Logistic Regression with only a difference of 0.54%

Random Forest Classification works well without outliers (using binning) and with K-Fold Cross Validation with an accuracy of 80.23% and AUC value of 0.815. Clearly, Random Forest performs better than Decision Tree model because Random Forest is an enhanced version of Decision Tree. Decision Tree model including feature selection showed highest accuracy amongst the other variations of Decision Tree models at 76.76%.

We ran a model comparison between the four models using model\_selection by sklearn and the results are consistent with our conclusion.

