# Data622 - Group2 - Homework3

Zachary Palmore, Kevin Potter, Amit Kapoor, Adam Gersowitz, Paul Perez

10/2/2021

## Contents

# Overview

In this project we will develop models that allow us to predict whether a loan is approved given certain indicators. Models will include linear discriminant analysis, K-nearest neighbor, decision trees, and random forest algorithms and we will assess which performs best at predicting loan approval status through performance statistics.

# Approach

For this project, we begin with data exploration to understand the relationships our target variable 'Loan_Status' will have with our variables and the variables' relationships to each other. This allows us to determine the steps necessary to set up for model development. Once we have an understanding of these variables we use that knowledge to prepare the data. We handle missing values, subset, train and split the data 75/25 so that we may better extract information when modeling. Then, we build the models and predict with the testing dataset. We focus on prediction accuracy when assessing the models but consider a host of performance statistics and real-world applications to determine which model is best.

We will use 'r' for data modeling. All packages used for data exploration, visualization, preparation and modeling are listed in Code Appendix.

# Data Exploration

## Data Characteristics

There are 614 observations of 12 variables. Each observation is an applicant's application for a loan with its corresponding variables of interest. Below is the description of the variables of interest in the data set.

| VARIABLE NAME | DESCRIPTION |
|---|---|
| Loan_ID | Unique Loan ID |
| Gender | Male/ Female |
| Married | Applicant married (Y/N) |
| Dependents | Number of dependents |
| Education | Applicant Education (Graduate/ Undergraduate) |
| Self_Employed | Self employed (Y/N) |
| ApplicantIncome | Applicant income |
| CoapplicantIncome | Coapplicant income |
| LoanAmount | Loan amount in thousands |
| Loan_Amount_Term | Term of loan in months |
| Credit_History | credit history meets guidelines |
| Property_Area | Urban/ Semi Urban/ Rural |
| Loan_Status | Loan approved (Y/N) |

There are four numeric variables represented by loan amount, loan amount term, applicant and co-applicant income. Several of these variables appear to be factors with specific levels but are not coded as such. For example, Gender, Married, Dependents, Education, Self_Employed, Property_Area, Credit_History, and Loan_Status are character strings. We will need to fix this if we are to make use of them.

## Data summary

Below is a summary of the loan approval dataset. For this process we have already adjusted the data types to their proper forms. This summarizing function quantifies each variable in a manner consistent with their types. We notice the levels of each factor in the 'Stats/Values' column, the frequency of valid (non-missing) observations per level of our factors, and the quantity and percent missing alongside them. We review these statistics to identify any issues with each variable.

```
## Data Frame Summary
## loan_data
## Dimensions: 614 x 12
## Duplicates: 0
##
## +----+------------------+------------------------------+----------------------+----------+---------+
## | No | Variable         | Stats / Values               | Freqs (% of Valid)   | Valid    | Missing |
## +====+==================+==============================+======================+==========+=========+
## | 1  | Gender           | 1. Female                    | 112 (18.6%)          | 601      | 13      |
## |    | [factor]         | 2. Male                      | 489 (81.4%)          | (97.9%)  | (2.1%)  |
## +----+------------------+------------------------------+----------------------+----------+---------+
## | 2  | Married          | 1. No                        | 213 (34.9%)          | 611      | 3       |
## |    | [factor]         | 2. Yes                       | 398 (65.1%)          | (99.5%)  | (0.5%)  |
## +----+------------------+------------------------------+----------------------+----------+---------+
## | 3  | Dependents       | 1. 0                         | 345 (57.6%)          | 599      | 15      |
## |    | [factor]         | 2. 1                         | 102 (17.0%)          | (97.6%)  | (2.4%)  |
## |    |                  | 3. 2                         | 101 (16.9%)          |          |         |
## |    |                  | 4. 3+                        |  51 ( 8.5%)          |          |         |
## +----+------------------+------------------------------+----------------------+----------+---------+
## | 4  | Education        | 1. Graduate                  | 480 (78.2%)          | 614      | 0       |
## |    | [factor]         | 2. Not Graduate              | 134 (21.8%)          | (100.0%) | (0.0%)  |
## +----+------------------+------------------------------+----------------------+----------+---------+
## | 5  | Self_Employed    | 1. No                        | 500 (85.9%)          | 582      | 32      |
## |    | [factor]         | 2. Yes                       |  82 (14.1%)          | (94.8%)  | (5.2%)  |
## +----+------------------+------------------------------+----------------------+----------+---------+
## | 6  | ApplicantIncome  | Mean (sd) : 5403.5 (6109)    | 505 distinct values  | 614      | 0       |
## |    | [integer]        | min < med < max:             |                      | (100.0%) | (0.0%)  |
## |    |                  | 150 < 3812.5 < 81000         |                      |          |         |
## |    |                  | IQR (CV) : 2917.5 (1.1)      |                      |          |         |
## +----+------------------+------------------------------+----------------------+----------+---------+
## | 7  | CoapplicantIncome| Mean (sd) : 1621.2 (2926.2)  | 287 distinct values  | 614      | 0       |
## |    | [numeric]        | min < med < max:             |                      | (100.0%) | (0.0%)  |
## |    |                  | 0 < 1188.5 < 41667           |                      |          |         |
## |    |                  | IQR (CV) : 2297.2 (1.8)      |                      |          |         |
## +----+------------------+------------------------------+----------------------+----------+---------+
## | 8  | LoanAmount       | Mean (sd) : 146.4 (85.6)     | 203 distinct values  | 592      | 22      |
## |    | [integer]        | min < med < max:             |                      | (96.4%)  | (3.6%)  |
## |    |                  | 9 < 128 < 700                |                      |          |         |
## |    |                  | IQR (CV) : 68 (0.6)          |                      |          |         |
## +----+------------------+------------------------------+----------------------+----------+---------+
## | 9  | Loan_Amount_Term | Mean (sd) : 342 (65.1)       | 12 :   1 ( 0.2%)     | 600      | 14      |
## |    | [integer]        | min < med < max:             | 36 :   2 ( 0.3%)     | (97.7%)  | (2.3%)  |
## |    |                  | 12 < 360 < 480               | 60 :   2 ( 0.3%)     |          |         |
## |    |                  | IQR (CV) : 0 (0.2)           | 84 :   4 ( 0.7%)     |          |         |
## |    |                  |                              | 120 :   3 ( 0.5%)    |          |         |
## |    |                  |                              | 180 :  44 ( 7.3%)    |          |         |
```

```
## |    |                   |                              | 240 :   4 ( 0.7%)   |          |         |
## |    |                   |                              | 300 :  13 ( 2.2%)   |          |         |
## |    |                   |                              | 360 : 512 (85.3%)   |          |         |
## |    |                   |                              | 480 :  15 ( 2.5%)   |          |         |
## +----+-------------------+------------------------------+---------------------+----------+---------+
## | 10 | Credit_History    | 1. 0                         |  89 (15.8%)         | 564      | 50      |
## |    | [factor]          | 2. 1                         | 475 (84.2%)         | (91.9%)  | (8.1%)  |
## +----+-------------------+------------------------------+---------------------+----------+---------+
## | 11 | Property_Area     | 1. Rural                     | 179 (29.2%)         | 614      | 0       |
## |    | [factor]          | 2. Semiurban                 | 233 (37.9%)         | (100.0%) | (0.0%)  |
## |    |                   | 3. Urban                     | 202 (32.9%)         |          |         |
## +----+-------------------+------------------------------+---------------------+----------+---------+
## | 12 | Loan_Status       | 1. N                         | 192 (31.3%)         | 614      | 0       |
## |    | [factor]          | 2. Y                         | 422 (68.7%)         | (100.0%) | (0.0%)  |
## +----+-------------------+------------------------------+---------------------+----------+---------+
```

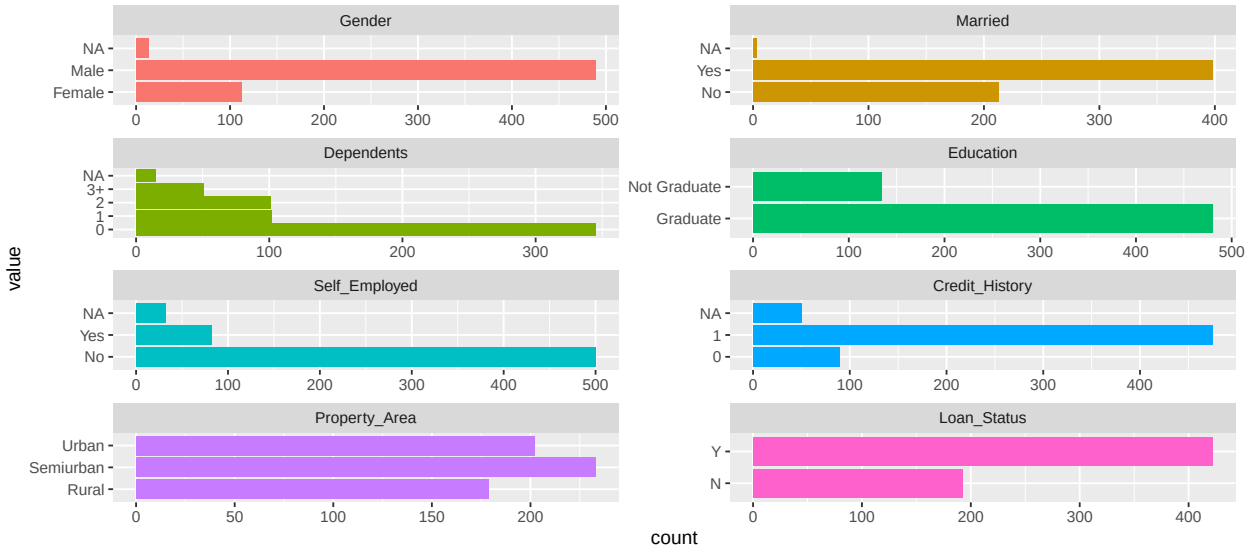
There are 7 columns that have missing values. The proportion of values for several columns shows significant differences and skew. For example, 97.9% of this dataset contains males applicants based on observations of the Gender variable, 99.5% of applicants are married people given the Married variable, and over 90% of our observations have longer Credit_History. Due to the disproportionate levels within the variables we should expect the data is not representative of a larger population unless that population happens to have similar proportions.

Our numeric incomes variables show significant signs of skew through the differences in their mean and medians as well as their ranges. The lowest applicant income was 150, while the highest was 81000. A similar problem exists with our co-applicant income data having had individuals with 0 income on the lowest end of the range and 41667 on the highest.

However, all of the observations contained an applicant and co-applicant income. Since some applicants may not have used a co-applicant on their applications, part of this skew could be caused by the data collection process. Additionally, we are only missing 3.6% of the observations of loan amount and 2.3% for loan terms.

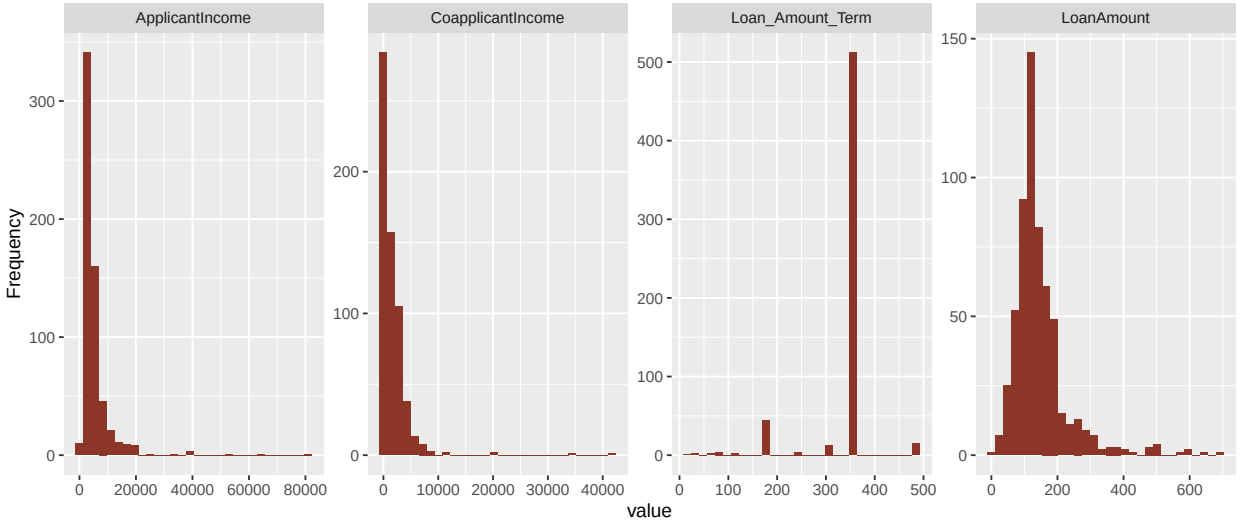
There are regular intervals and commonality in the loan term amounts which indicates we may have been able to factorize their data types. We chose instead to leave it as a discrete numeric value since it represents the term length which could be any number of days or months. We note that 85.3% percent of these applicants applied for a loan term of 360 but we are unsure if that is due to the lending institutions standard practice or if applicants requested this specific term.

For exploratory purposes, we visualize the proportions to see just how skewed and disproportionate this dataset is. We include missing values to demonstrate their influence on the dataset as well. The chart below shows the distribution of all categorical variables, which includes the factors mentioned previously.




From this chart, it is very clear we have a dataset with mostly married male graduates with no dependents, a long credit history, and who are not self-employed. There is a relatively even mix of urban, suburban, and rural applicants and a small number of missing values. Applicants tend to be accepted more often than not and there are no missing observations for our target variable 'Loan_Status' nor the applicant's property area or education. These are all of our categorical variables.

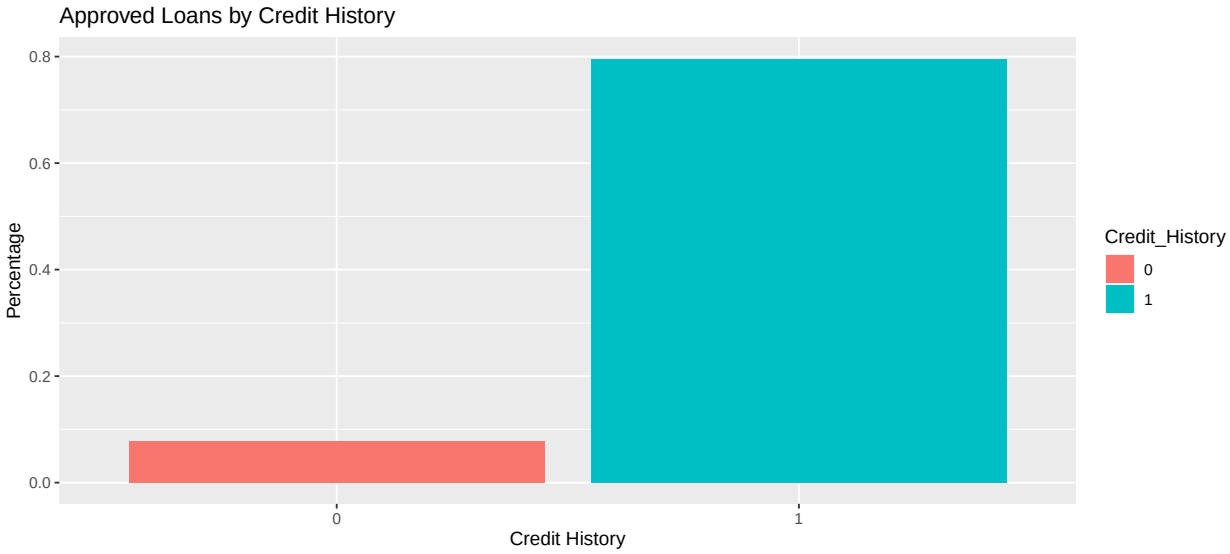
We also generate histograms with the count of each observation to assess our numeric variable distributions. This will let us know more about the skewness, average values, and where potential outliers may be found for our numeric variables. The graph below shows their distributions.



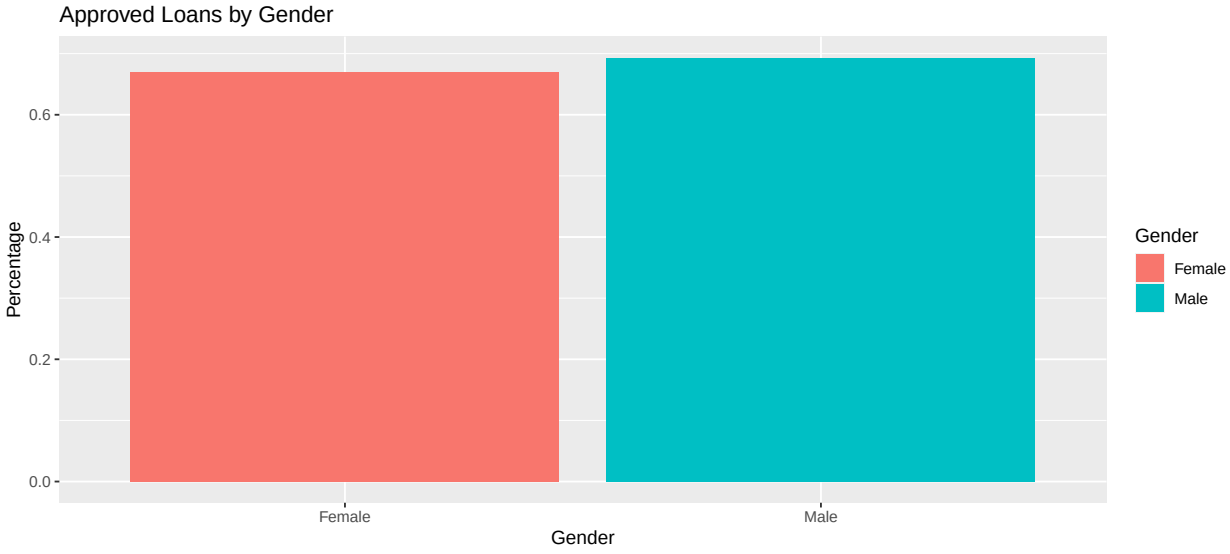

The applicant income and co-applicant income variables are highly right skewed with a smaller number of individual applicants stretching the distribution towards higher incomes. For analysis purposes, we must keep in mind that only a handful of applicants had higher incomes while the bulk of applicants were concentrated at the lower end of the income distribution. The loan amount term has one spike at 360. Meanwhile, the loan amount distribution is the closest to normal. These results are consistent with our summary table.

Next we will review the impact of the categorical variables' proportions on loan approval in more detail by isolating the factor levels individually. Here again, we visualize the proportions as a bar chart without missing values and expand the size of the chart to see the nuances of each. These are placed alongside each variable's frequency table by level to visualize their proportions. The results are as follows:

```
##   Credit_History Loan_Status       Freq
## 1              0           Y 0.07865169
## 2              1           Y 0.79578947
```

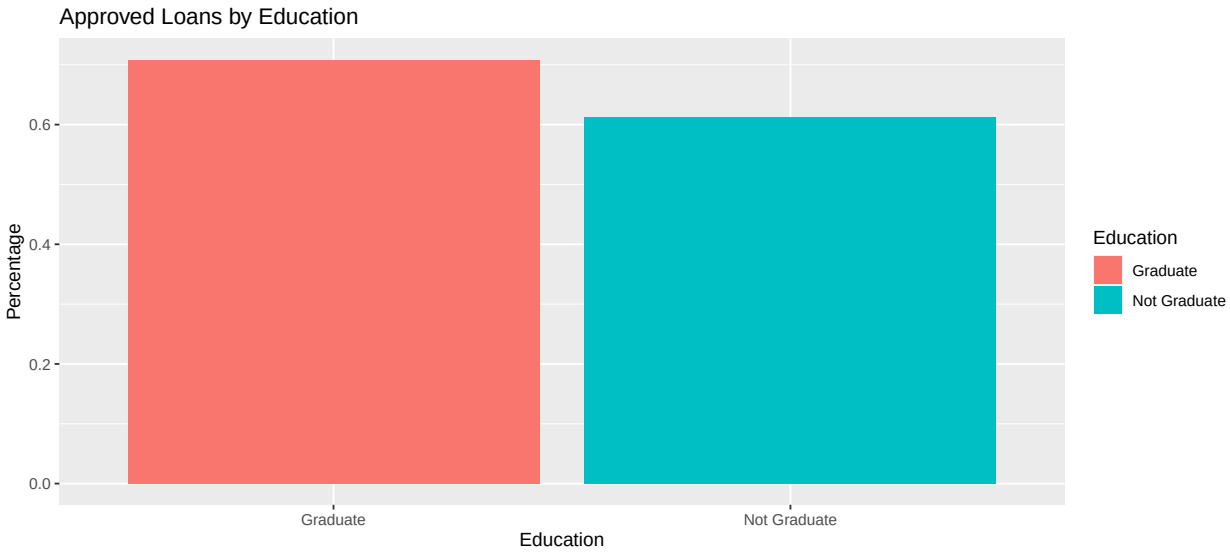



```
##   Gender Loan_Status      Freq
## 1 Female           Y 0.6696429
## 2   Male           Y 0.6932515
```

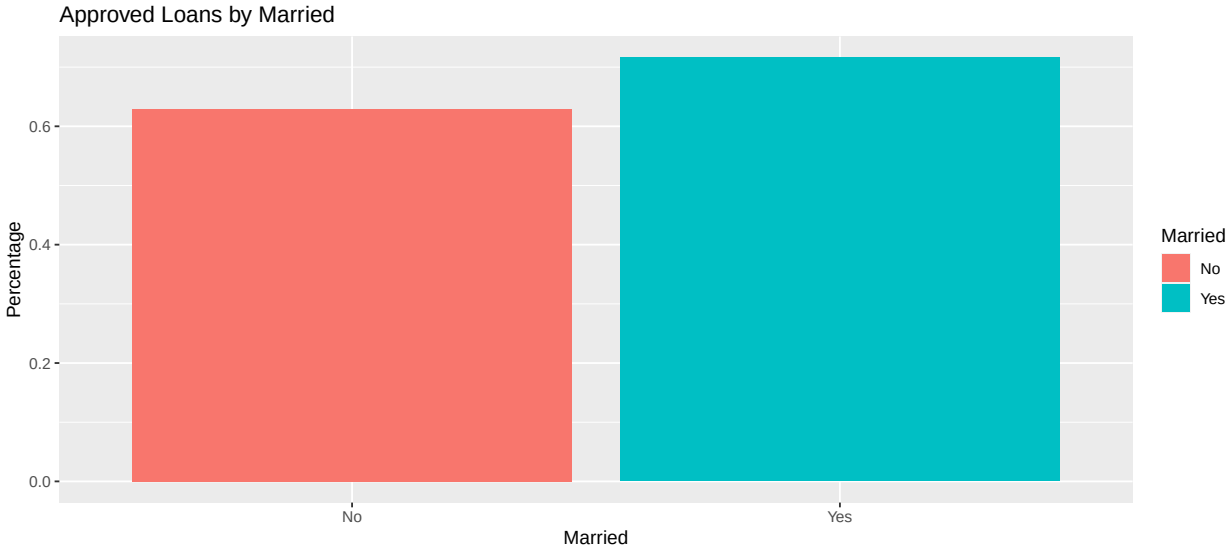



```
##      Education Loan_Status      Freq
```

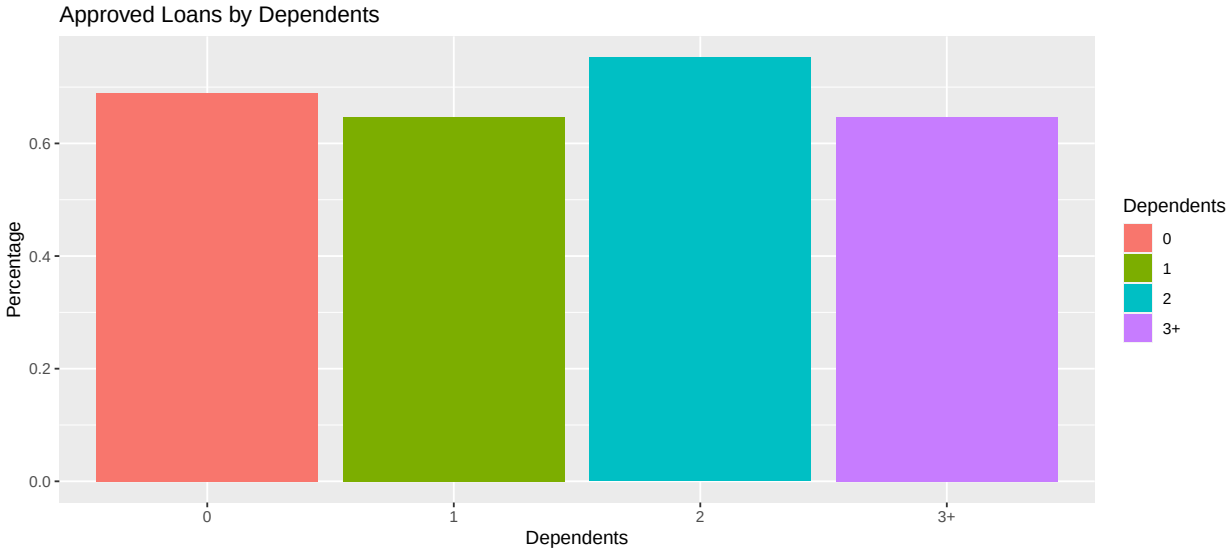
```
## 1     Graduate            Y 0.7083333
## 2 Not Graduate            Y 0.6119403
```




```
##   Married Loan_Status      Freq
## 1      No            Y 0.6291080
## 2     Yes            Y 0.7160804
```



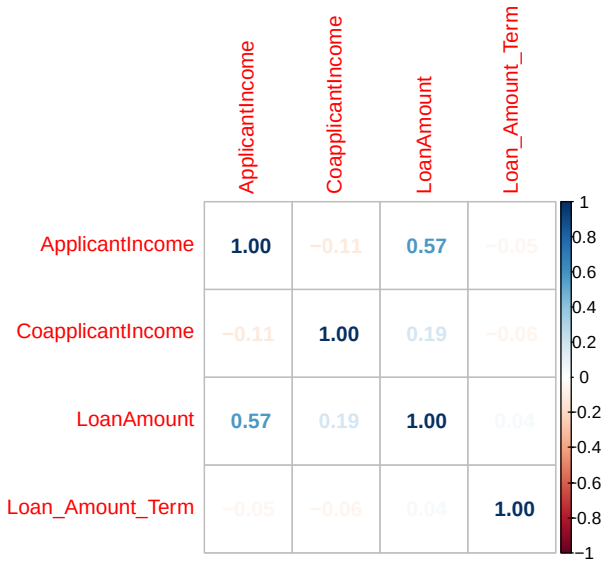

```
##   Dependents Loan_Status      Freq
## 1          0            Y 0.6898551
## 2          1            Y 0.6470588
## 3          2            Y 0.7524752
## 4         3+            Y 0.6470588
```




These bar charts confirm our thoughts about the dataset's disproportionalities. Missing values have little effect on the overall proportions and so they can be removed. It remains male dominated with applicants who are married, have no dependents, are highly educated, and have a long credit history.

## Correlations

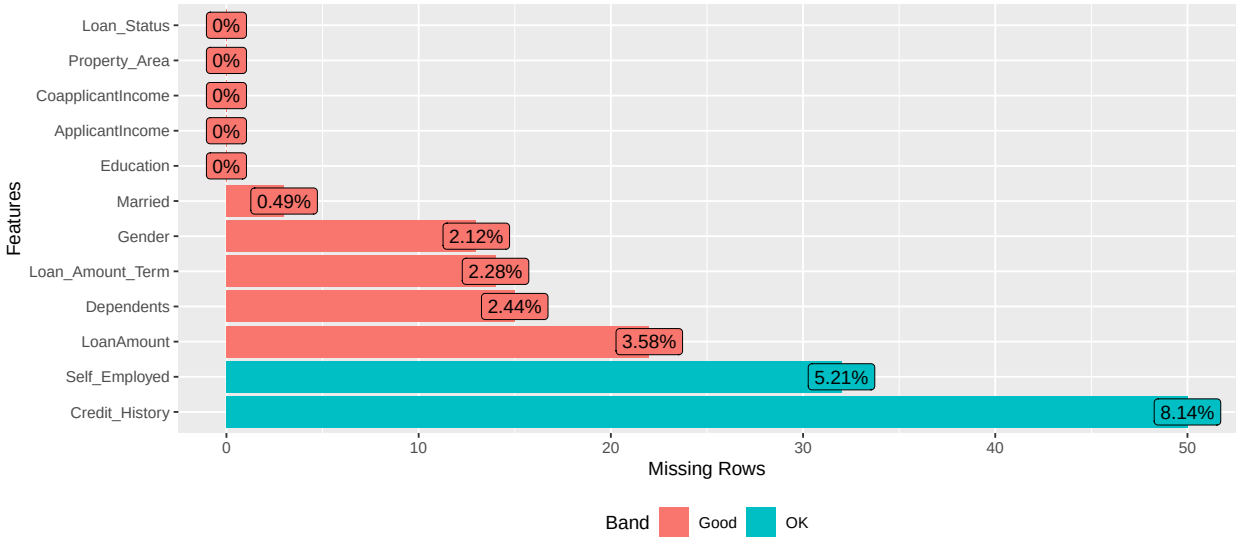
To determine how well each variable is correlated with our target variable and with one another, we construct a correlation plot. This plot contains the values of all correlation between variables represented by colors and numbers. The lighter the color, the lower the correlation. Meanwhile, darker blue indicates stronger positive correlations while darker red indicates stronger negative correlations.




Given that our numeric features have correlation values near 0, they do not seem to be strongly correlated with our target. They also do not seem to have any correlation with one another so this is a factor that does not have to be dealt with.

# Data Preparation

## Handling missing values




We can see from this chart that credit_history contributes to 8.14% of missing data, self_employed accounts for more than 5% of missing data and so on. All records having missing categorical predictors must be removed. We perform this removal by filtering the variables to their complete cases. We then impute numeric values using MICE (Multivariate Imputation by Chained Equations).

```
## [1] 511  12
```

Our final, missing value cleaned dataset contains 511 rows and 12 columns. This reduces the observation size by the amount of missing values present in each variable because those missing were imputed or filtered out.

## Preprocess using transformation

Due to the right skew in the numeric features, we adjust the distribution to center, scale, and transform the data to reduce bias. This preprocessing uses the caret package's 'preprocessing' function to return a box cox transformation on all numeric variables in our loan dataset. These numeric variables include the applicant and coapplicant income, the loan amount, and loan amount term. We process further with a center scaling of the values to place them all within the same grid axes. This will provide greater clarity to understand linearity and separability before modeling.

## Training and Test Partition

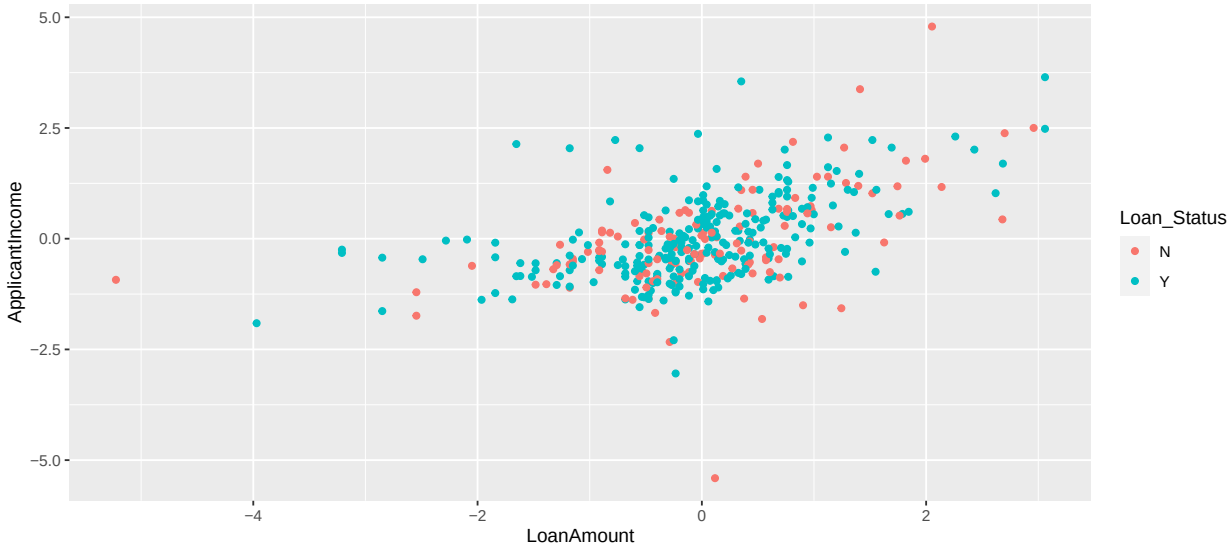
In this step for data preparation we will partition the training dataset in training and validation sets using `createDataPartition` method from `caret` package. We will reserve 75% for training and rest 25% for validation purpose.

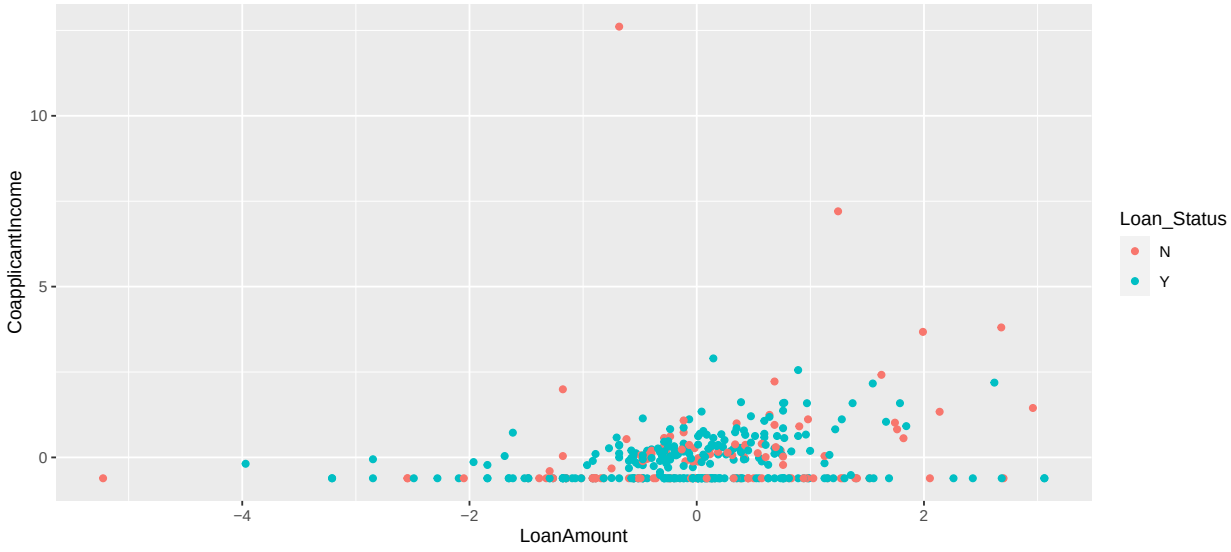
# Build Models

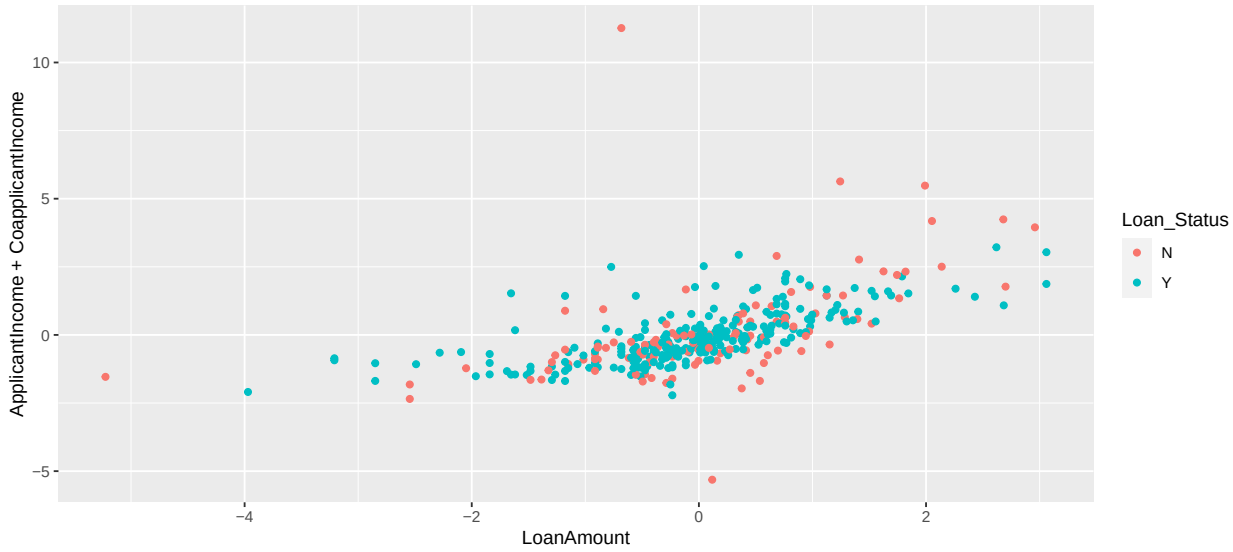
## Linear Discriminant Analysis (LDA)

Linear Discriminant analysis is when the dependent variable has two categories and the LDA function that passes through the means of the two categories can be used to discriminate between the two categories. LDA projects the data on a new axis to maximize the separation between two categories. Linear Discriminant Analysis is similar to PCA but its focus is more towards maximizing the separability among known categories. LDA is a special form of QDA, where each class are assumed to share the same covariance matrix

**Here LDA doesn't seem to be a good model as below plots shows that data points given are not linearly separable.**






```
## Call:
## lda(Loan_Status ~ LoanAmount + ApplicantIncome + CoapplicantIncome,
##     data = loan_data)
##
## Prior probabilities of groups:
##         N         Y
## 0.3209393 0.6790607
##
## Group means:
##    LoanAmount ApplicantIncome CoapplicantIncome
## N  0.07966414     0.003576320         0.0571435
## Y -0.03765106    -0.001690249        -0.0270073
##
## Coefficients of linear discriminants:
##                           LD1
## LoanAmount        -0.9679778
## ApplicantIncome    0.3878895
## CoapplicantIncome -0.3369942
```

## K-nearest neighbor (KNN)

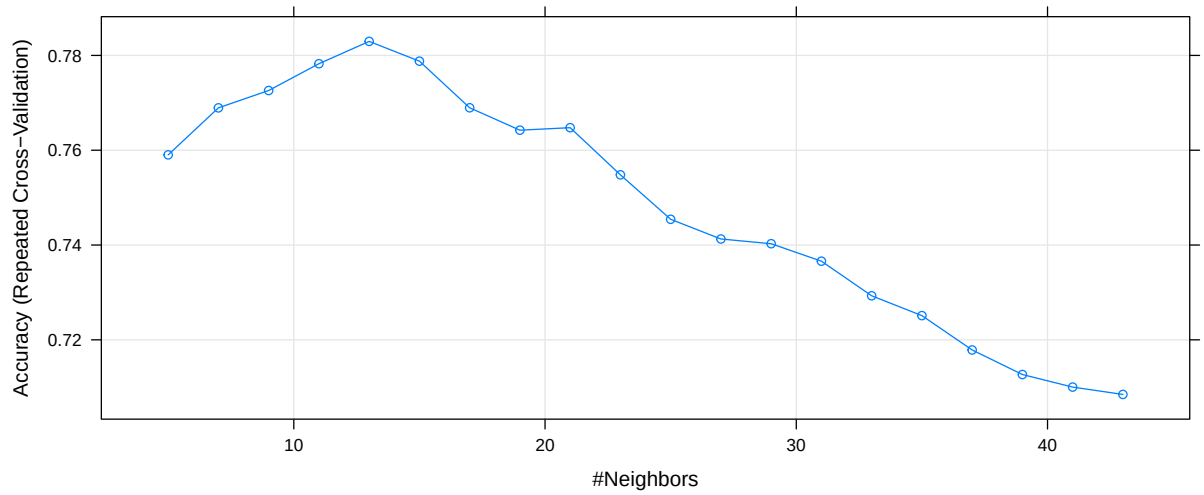
A K-nearest neighbor, or KNN algorithm, is a method of classifying data that relies on similar points being close together to identify groups. It uses the distance between data points to assign the nearest points to groups based on the value of K. This is why it is important to preprocess the data.

In this case we scaled and centered the data to reduce bias and improve our model's predictive capability (especially in our model's accuracy) as we have done below. Sometimes these groups are called batches. In general, smaller values for K indicate fewer batches and larger distances between similar points while larger values for K, have smaller distances between similar points. This K-value can be optimized.

To determine our optimal K-value we used a function from the caret package that iterates through a number of K-neighbors consecutively until it experimentally finds the one that produces the most accurate predictions. This function stops once it recognizes overfitting. It then selects the best value for K and creates a model with it. The graph below shows how this process worked and our optimal value K is the maximum point on the line.

```
## Created from 384 samples and 12 variables
##
## Pre-processing:
##   - centered (4)
##   - ignored (8)
##   - scaled (4)

## k-Nearest Neighbors
##
## 384 samples
##  11 predictor
##   2 classes: 'N', 'Y'
##
## Pre-processing: centered (14), scaled (14)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 346, 346, 346, 345, 346, 345, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.7590209  0.3575145
##    7  0.7689406  0.3751117
##    9  0.7725985  0.3746036
##   11  0.7782524  0.3884713
##   13  0.7829615  0.3994494
##   15  0.7787908  0.3849571
##   17  0.7689528  0.3503238
##   19  0.7642294  0.3309567
##   21  0.7647551  0.3299468
##   23  0.7548097  0.2947001
##   25  0.7454298  0.2607665
##   27  0.7412733  0.2448000
##   29  0.7402746  0.2423939
##   31  0.7365911  0.2319391
##   33  0.7292901  0.2050164
##   35  0.7251066  0.1885755
##   37  0.7178596  0.1615955
##   39  0.7126768  0.1383143
##   41  0.7100317  0.1282468
##   43  0.7084798  0.1212857
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 13.
```




```
## [1] 0.7952756
```

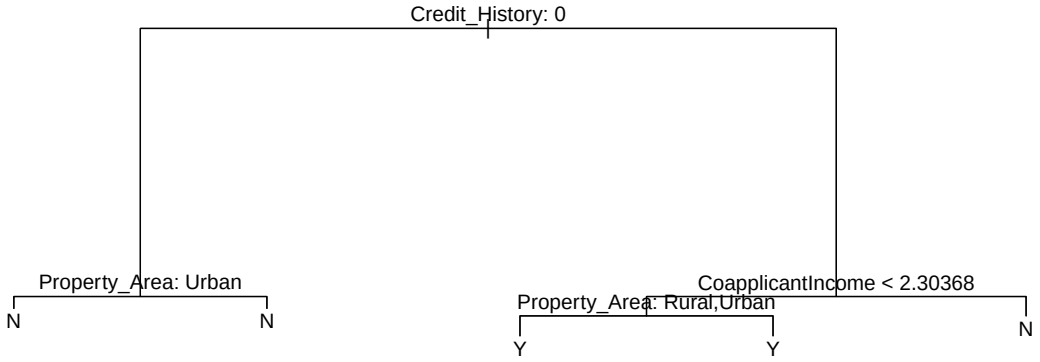
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 17  2
##          Y 24 84
##
##                Accuracy : 0.7953
##                  95% CI : (0.7146, 0.8617)
##     No Information Rate : 0.6772
##     P-Value [Acc > NIR] : 0.002202
##
##                   Kappa : 0.4553
##
##  Mcnemar's Test P-Value : 3.814e-05
##
##             Sensitivity : 0.4146
##             Specificity : 0.9767
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.7778
##              Prevalence : 0.3228
##          Detection Rate : 0.1339
##    Detection Prevalence : 0.1496
##       Balanced Accuracy : 0.6957
##
##        'Positive' Class : N
##
```

KNN model accuracy comes out as ~80%

## Decision Trees

In a decision tree model the data is split into distinct options of 'yes' or 'no' based on parameters that make the options possible. These splits are called nodes and the decisions made at them can be mapped. For example, we provide a small decision tree that shows how decisions can be made based on credit history, coapplicant income, and property area.

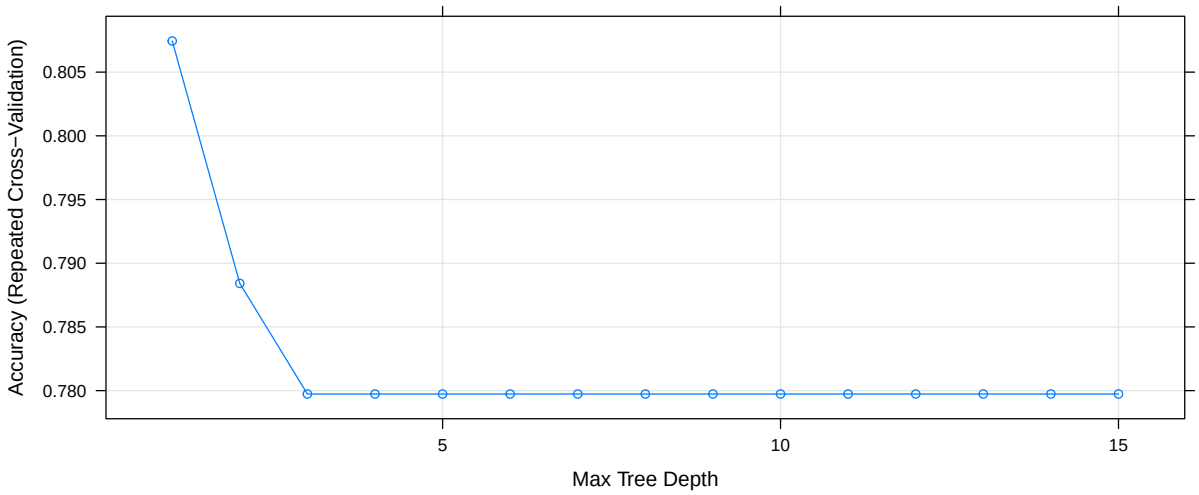
```
##
## Classification tree:
## tree(formula = Loan_Status ~ ., data = training)
## Variables actually used in tree construction:
## [1] "Credit_History"    "Property_Area"     "CoapplicantIncome"
## Number of terminal nodes:  5
## Residual mean deviance:  0.921 = 349 / 379
## Misclassification error rate: 0.1849 = 71 / 384
```

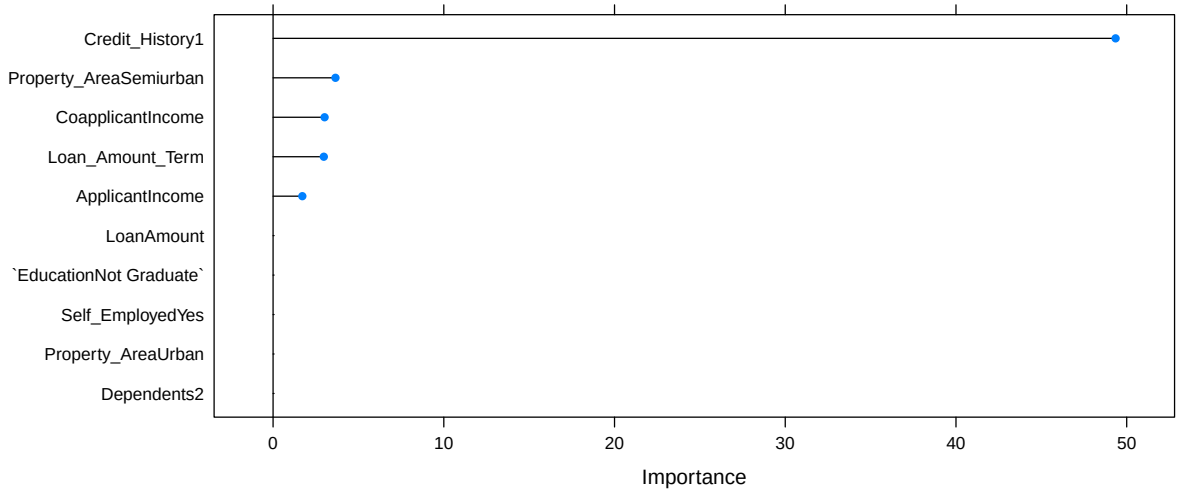



We also review which variables are most important for making decisions in our model.These are shown in the plot as a straight line extending from the axis to the length of its importance to the model. Accuracy was also used to select the optimal model using the largest value where our final tree depth used for this model is 1.

```
## CART
##
## 384 samples
##  11 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 346, 346, 346, 345, 346, 345, ...
## Resampling results across tuning parameters:
##
##   maxdepth  Accuracy   Kappa
```

```
##     1        0.8074449  0.4816227
##     2        0.7884166  0.4426131
##     3        0.7797346  0.4291257
##     4        0.7797346  0.4291257
##     5        0.7797346  0.4291257
##     6        0.7797346  0.4291257
##     7        0.7797346  0.4291257
##     8        0.7797346  0.4291257
##     9        0.7797346  0.4291257
##    10        0.7797346  0.4291257
##    11        0.7797346  0.4291257
##    12        0.7797346  0.4291257
##    13        0.7797346  0.4291257
##    14        0.7797346  0.4291257
##    15        0.7797346  0.4291257
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was maxdepth = 1.
```






```
## [1] 0.8110236
```

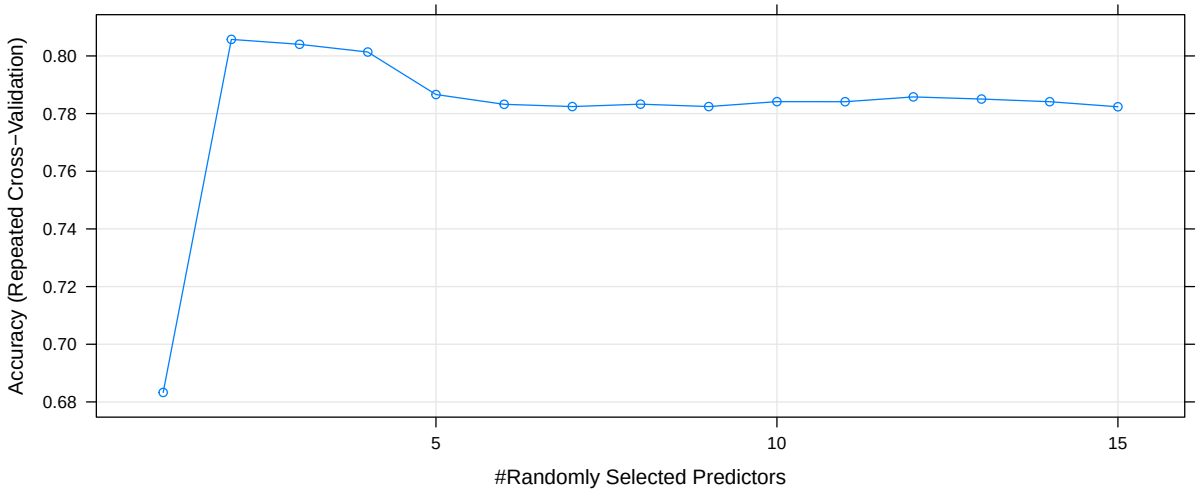
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 19  2
##          Y 22 84
##
##                Accuracy : 0.811
##                  95% CI : (0.732, 0.875)
##     No Information Rate : 0.6772
##     P-Value [Acc > NIR] : 0.0005479
##
##                   Kappa : 0.5046
##
##  Mcnemar's Test P-Value : 0.0001052
##
##             Sensitivity : 0.4634
##             Specificity : 0.9767
##          Pos Pred Value : 0.9048
##          Neg Pred Value : 0.7925
##              Prevalence : 0.3228
##          Detection Rate : 0.1496
##    Detection Prevalence : 0.1654
##       Balanced Accuracy : 0.7201
##
##        'Positive' Class : N
##
```

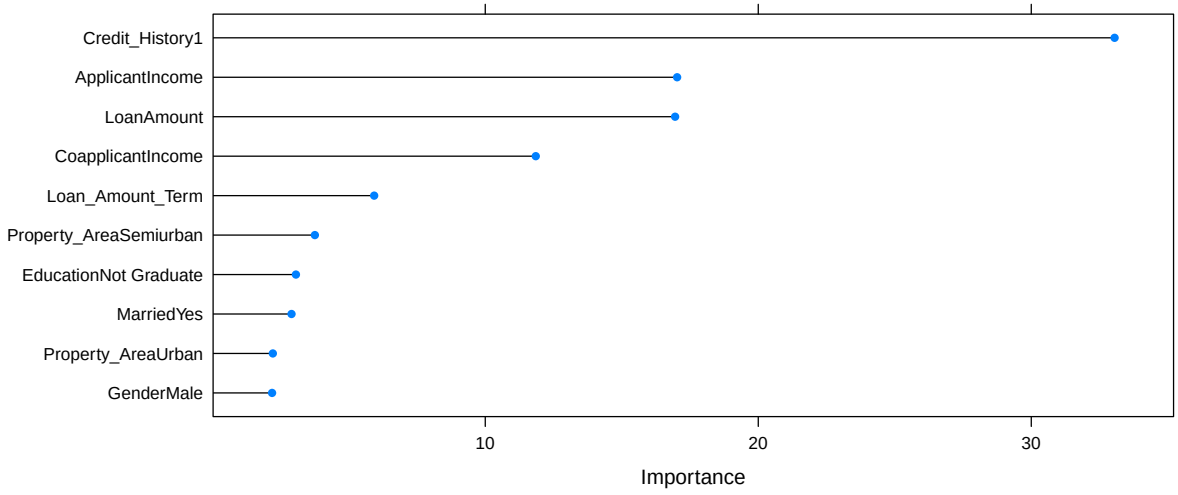
Our decision Tree model accuracy comes out as ~81%. As shown in the confusion matrix, there is room for improvement in this model's sensitivity among other variables. We try to improve this with the random forest model.

## Random Forests

A random forest model works by building a number of decision trees and selecting the most accurate decisions from the trees. These decisions are randomized and in our case, tries 3 variables at each node or split in the tree. We set our number of trees to 500 and train the model to predict loan status. We review the variables of most importance in the model and in this case, give the model a boost to improve accuracy.

```
## Random Forest
##
## 384 samples
##  11 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 346, 346, 346, 345, 346, 345, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.6832883  0.01878108
##    2    0.8057355  0.48027628
##    3    0.8040261  0.48616264
##    4    0.8013495  0.48531970
##    5    0.7866172  0.45449370
##    6    0.7832209  0.44798478
##    7    0.7824336  0.44825547
##    8    0.7832659  0.45090023
##    9    0.7824336  0.44867593
##   10    0.7841430  0.45361152
##   11    0.7841206  0.45327710
##   12    0.7857850  0.45879852
##   13    0.7850427  0.45712485
##   14    0.7841206  0.45227555
##   15    0.7823662  0.44938391
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```




```
## [1] 0.8188976

## Confusion Matrix and Statistics
##
##          Reference
## Prediction  N  Y
##          N 20  2
##          Y 21 84
##
##                Accuracy : 0.8189
##                  95% CI : (0.7408, 0.8816)
##     No Information Rate : 0.6772
##     P-Value [Acc > NIR] : 0.0002545
##
##                   Kappa : 0.5286
```

```
##
##  Mcnemar's Test P-Value : 0.0001746
##
##             Sensitivity : 0.4878
##             Specificity : 0.9767
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 0.8000
##              Prevalence : 0.3228
##          Detection Rate : 0.1575
##    Detection Prevalence : 0.1732
##       Balanced Accuracy : 0.7323
##
##        'Positive' Class : N
##
```

Our random Forest model accuracy comes out as ~81.1%. This is an improvement upon our decision model and the sensitivity did increase as we desired.

# Model Performance

All 4 of the models we built above have an accuracy rate of around 80% with the LDA model getting the light edge in accuracy (81.1% to 79.5% for the other 3).

Next we will look at some more detailed accuracy metrics produced from the predict function.

In this particular case, since the accuracy results are so similar it is wise to examine which models are most often leading to type 1 or type 2 errors. Assuming a null hypothesis is not giving a loan a type 1 error is giving someone a loan when they should not have gotten one and a type 2 error is not giving someone a loan when they should have gotten one. This correlates to the sensitivity and specificity respectively. The balanced accuracy metric takes the mean of sensitivity and specificity in order to diagnose if a model appears to be accurate but is really only predicting the positive or negative case correctly. In this case the random forest model has a slight edge over the LDA in balanced accuracy (0.7212 to 0.7201).

Although the LDA has the smallest p-value it is still <0.0025 for all models so they can all be used so this does not help us decide which model to choose. The same can be said for the Mcnemar's p-value.

The LDA also has the best Kappa score of 0.5046. The Kappa score can be a much more accurate indicator of accuracy then the standard accuracy rate. The kappa score takes into account the expected accuracy of a model given the instances of each class. This helps a lot with unbalanced class numbers in the dataset. It is encouraging to see the LDA Kappa is also the highest which corresponds with it having the highest accuracy. Although the accuracy score is high a Kappa score around 0.5046 only indicates a moderately good model.

Given the LDa has the highest accuracy, kappa, and nearly the highest balanced accuracy, the LDA is the model that we would use going forward.

# Conclusion

After reviewing the results of 4 different models (LDA, KNN, Decision Tree, and Random Forest), we found that LDA is the most accurate by numerous metrics and should be considered the best model for this data. However, the metrics for these 4 models was very close and it did not leave us with no choice but to pick the LDA. If presenting this to stakeholders we would recommend the LDA model with the caveat they further testing and verification may be needed to ensure the LDA is the most accurate model of other iterations of the split of training and testing data.

## References

https://www.r-bloggers.com/2018/07/prop-table/

https://www.datacamp.com/community/tutorials/decision-trees-R

https://stats.stackexchange.com/questions/82162/cohens-kappa-in-plain-english

## Code Appendix

```
knitr::opts_chunk$set(echo=FALSE, error=FALSE, warning=FALSE, message=FALSE, fig.align="center", fig.wi
# Libraries

library(summarytools)
library(tidyverse)
library(DataExplorer)
library(reshape2)
library(mice)
library(caret)
library(MASS)
library(e1071)
library(caret)
library(tree)
library(randomForest)
library(corrplot)

set.seed(622)
# read data, change blank to NA and and remove loan_id
loan_data <- read.csv('https://raw.githubusercontent.com/amit-kapoor/Data622Group2/main/Loan_approval.cs
  na_if("") %>%
  dplyr::select(-1)

# categorical columns as factors
loan_data <- loan_data %>%
  mutate(Gender=as.factor(Gender),
         Married=as.factor(Married),
         Dependents=as.factor(Dependents),
         Education=as.factor(Education),
         Self_Employed=as.factor(Self_Employed),
         Property_Area=as.factor(Property_Area),
         Credit_History=as.factor(Credit_History),
         Loan_Status=as.factor(Loan_Status))

dfSummary(loan_data, style = 'grid', graph.col = FALSE)

# select categorical columns
cat_cols = c()
j <- 1
for (i in 1:ncol(loan_data)) {
  if (class((loan_data[,i])) == 'factor') {
      cat_cols[j]=names(loan_data[i])
      j <- j+1
```

```
  }
}

loan_fact <-  loan_data[cat_cols]
# long format
loan_factm <- melt(loan_fact, measure.vars = cat_cols, variable.name = 'metric', value.name = 'value')

# plot categorical columns
ggplot(loan_factm, aes(x = value)) +
  geom_bar(aes(fill = metric)) +
  facet_wrap( ~ metric, nrow = 5L, scales = 'free') + coord_flip() +
  theme(legend.position = "none")
plot_histogram(loan_data, geom_histogram_args = list("fill" = "tomato4"))
loan_ch <- with(loan_data, table(Credit_History, Loan_Status)) %>%
  prop.table(margin = 1) %>% as.data.frame() %>% filter(Loan_Status == 'Y')

loan_ch
ggplot(loan_ch, aes(x=Credit_History, y=Freq, fill=Credit_History)) + geom_bar(stat='identity') + labs(
loan_gen <- with(loan_data, table(Gender, Loan_Status)) %>%
  prop.table(margin = 1) %>% as.data.frame() %>% filter(Loan_Status == 'Y')

loan_gen
ggplot(loan_gen, aes(x=Gender, y=Freq, fill=Gender)) + geom_bar(stat='identity') + labs(title = 'Approv
loan_ed <- with(loan_data, table(Education, Loan_Status)) %>%
  prop.table(margin = 1) %>% as.data.frame() %>% filter(Loan_Status == 'Y')

loan_ed
ggplot(loan_ed, aes(x=Education, y=Freq, fill=Education)) + geom_bar(stat='identity') + labs(title = 'Ap
loan_mar <- with(loan_data, table(Married, Loan_Status)) %>%
  prop.table(margin = 1) %>% as.data.frame() %>% filter(Loan_Status == 'Y')

loan_mar
ggplot(loan_mar, aes(x=Married, y=Freq, fill=Married)) + geom_bar(stat='identity') + labs(title = 'Appr
loan_dep <- with(loan_data, table(Dependents, Loan_Status)) %>%
  prop.table(margin = 1) %>% as.data.frame() %>% filter(Loan_Status == 'Y')

loan_dep
ggplot(loan_dep, aes(x=Dependents, y=Freq, fill=Dependents)) + geom_bar(stat='identity') + labs(title =
cors <- loan_data %>%
  select_if(is.numeric) %>%
  na.omit() %>%
  cor()
corrplot::corrplot(cors, method="number")
# G = cor(loan_data[6:(length(loan_data)-3)])
# corrplot(G, method = 'number') # colorful number
# plot missing values
plot_missing(loan_data)
# Filter out the data which has missing categorical predictors
loan_data <- loan_data %>% filter(!is.na(Credit_History) &
                                  !is.na(Self_Employed) &
                                  !is.na(Dependents) &
                                  !is.na(Gender) &
                                  !is.na(Married))
```

```r
# impute numeric predictors using mice
loan_data <- complete(mice(data=loan_data, method="pmm", print=FALSE))
dim(loan_data)
# library(e1071) - where this was used
set.seed(622)
loan_data <- loan_data %>%
  dplyr::select(c("ApplicantIncome", "CoapplicantIncome", "LoanAmount", "Loan_Amount_Term")) %>%
  preProcess(method = c("BoxCox","center","scale")) %>%
  predict(loan_data)
set.seed(622)
partition <- createDataPartition(loan_data$Loan_Status, p=0.75, list = FALSE)

training <- loan_data[partition,]
testing <- loan_data[-partition,]

# training/validation partition for independent variables
#X.train <- ld.clean[partition, ] %>% dplyr::select(-Loan_Status)
#X.test <- ld.clean[-partition, ] %>% dplyr::select(-Loan_Status)

# training/validation partition for dependent variable Loan_Status
#y.train <- ld.clean$Loan_Status[partition]
#y.test <- ld.clean$Loan_Status[-partition]
training %>%
  ggplot(aes(x = LoanAmount, y= ApplicantIncome, color = Loan_Status)) + geom_point()
training %>%
  ggplot(aes(x = LoanAmount, y= CoapplicantIncome, color = Loan_Status)) + geom_point()
training %>%
  ggplot(aes(x = LoanAmount, y= ApplicantIncome+CoapplicantIncome, color = Loan_Status)) + geom_point()
# LDA model
lda_model <- lda(Loan_Status ~ LoanAmount+ApplicantIncome+CoapplicantIncome, data = loan_data)
lda_model
# prediction from lda model
lda_predict <- lda_model %>%
  predict(testing)
# KNN model
set.seed(622)
train.knn <- training[, names(training) != "Direction"]
prep <- preProcess(x = train.knn, method = c("center", "scale"))
prep
cl <- trainControl(method="repeatedcv", repeats = 5)
knn_model <- train(Loan_Status ~ ., data = training,
                   method = "knn",
                   trControl = cl,
                   preProcess = c("center","scale"),
                   tuneLength = 20)
knn_model
# prediction from knn model
plot(knn_model)
knn_predict <- predict(knn_model,newdata = testing)
mean(knn_predict == testing$Loan_Status) # accuracy
confusionMatrix(knn_predict, testing$Loan_Status)
# Decision Trees model
set.seed(622)
```

```
tree.loans = tree(Loan_Status~., data=training)
summary(tree.loans)
plot(tree.loans)
text(tree.loans, pretty = 0)
# Decision Trees model
set.seed(622)
control <- trainControl(method="repeatedcv", number=10, repeats=3, search='grid')
metric <- "Accuracy"
tunegrid <- expand.grid(.maxdepth=c(1:15))
tree.loans <- train(Loan_Status~., data = training, method="rpart2", tuneGrid=tunegrid, trControl=contro
print(tree.loans)
plot(tree.loans)

treeImp <- varImp(tree.loans, scale = FALSE)
plot(treeImp, top = 10)
# prediction from decision tree model
tree.predict <- predict(tree.loans, testing,type='raw')
mean(tree.predict == testing$Loan_Status) # accuracy
confusionMatrix(tree.predict, testing$Loan_Status)
set.seed(622)
# Random Forest model
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
metric <- "Accuracy"
tunegrid <- expand.grid(.mtry=c(1:15))
rf.loans <- train(Loan_Status~., data = training, method="rf",tuneGrid=tunegrid, trControl=control)
print(rf.loans)
plot(rf.loans)

rfImp <- varImp(rf.loans, scale = FALSE)
plot(rfImp, top = 10)
# prediction from random forest model
rf.predict <- predict(rf.loans, testing,type='raw')
mean(rf.predict == testing$Loan_Status) # accuracy
confusionMatrix(rf.predict, testing$Loan_Status)
```