Loan prediction report

Loan Prediction Problem

Problem Statement

Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan.

Problem

Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers' segments, those are eligible for loan amount so that they can specifically target these customers. Here they have provided a partial data set.

Preprocessing

Realistically, most of the data we will get, even from the government, can have errors, and it's important to identify these errors before spending time analyzing the data.

Let's start by reading the data using the function read.csv() and show the first part of the dataset:

```
setwd("C:/Learn/R/8. Machine Learning/Capstone/IDV/Loan-Prediction-with-R-master")
tr <- read.csv('train.csv', header = TRUE)
head(tr)
```

```
##    Loan_ID Gender Married Dependents    Education Self_Employed
## 1 LP001002   Male      No          0     Graduate            No
## 2 LP001003   Male     Yes          1     Graduate            No
## 3 LP001005   Male     Yes          0     Graduate           Yes
## 4 LP001006   Male     Yes          0 Not Graduate            No
## 5 LP001008   Male      No          0     Graduate            No
## 6 LP001011   Male     Yes          2     Graduate           Yes
##   ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term
## 1            5849                0         NA               360
## 2            4583             1508        128               360
## 3            3000                0         66               360
## 4            2583             2358        120               360
## 5            6000                0        141               360
## 6            5417             4196        267               360
##   Credit_History Property_Area Loan_Status
## 1              1         Urban           Y
## 2              1         Rural           N
## 3              1         Urban           Y
## 4              1         Urban           Y
## 5              1         Urban           Y
## 6              1         Urban           Y
```

The first row in the dataset defines the column header. Each of these headers is described in the above table. Now, we will run the summary function to have a quick look on the stats:

```
summary(tr)
```

```
##      Loan_ID       Gender    Married   Dependents        Education
##  LP001002:  1         : 13      :  3     : 15     Graduate    :480
##  LP001003:  1   Female:112   No :213   0 :345     Not Graduate:134
##  LP001005:  1   Male  :489   Yes:398   1 :102
##  LP001006:  1                           2 :101
##  LP001008:  1                           3+: 51
##  LP001011:  1
##  (Other) :608
##  Self_Employed ApplicantIncome CoapplicantIncome   LoanAmount
##     : 32       Min.   :  150   Min.   :    0     Min.   :  9.0
##  No :500       1st Qu.: 2878   1st Qu.:    0     1st Qu.:100.0
##  Yes: 82       Median : 3812   Median : 1188     Median :128.0
##                Mean   : 5403   Mean   : 1621     Mean   :146.4
##                3rd Qu.: 5795   3rd Qu.: 2297     3rd Qu.:168.0
##                Max.   :81000   Max.   :41667     Max.   :700.0
##                                                  NA's   :22
##  Loan_Amount_Term Credit_History     Property_Area Loan_Status
##  Min.   : 12      Min.   :0.0000   Rural    :179   N:192
##  1st Qu.:360      1st Qu.:1.0000   Semiurban:233   Y:422
##  Median :360      Median :1.0000   Urban    :202
##  Mean   :342      Mean   :0.8422
##  3rd Qu.:360      3rd Qu.:1.0000
##  Max.   :480      Max.   :1.0000
##  NA's   :14       NA's   :50
```

- There are (+) sign on 51 record
- The mean of Credit_history variable is 0.8422. That's weird knowing that this variable has value of 1 for customers who have credit history and 0 otherwise.
- There are blank fields in Gender, Married, Dependents and Self_Employed.
- There are NAs in LoanAmount, Loan_Amount_term and Credit_History.

Let's do now a quick fix on some of the variables:

```
setwd("C:/Learn/R/8. Machine Learning/Capstone/IDV/Loan-Prediction-with-R-master")
tr <- read.csv(file="train.csv", na.strings=c("", "NA"), header=TRUE)
library(plyr); library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
tr$Dependents <- revalue(tr$Dependents, c("3+"="3"))
```

Now, let's have a closer look at the missing data:

```
sapply(tr, function(x) sum(is.na(x)))
```

```
##           Loan_ID            Gender           Married        Dependents
##                 0                13                 3                15
##         Education     Self_Employed   ApplicantIncome CoapplicantIncome
##                 0                32                 0                 0
##        LoanAmount  Loan_Amount_Term    Credit_History     Property_Area
##                22                14                50                 0
##       Loan_Status
##                 0
```

```
#please install the following packages below if you do not have it in your library
library(mice)
```

```
## Loading required package: lattice

##
## Attaching package: 'mice'

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(VIM)
```

```
## Loading required package: colorspace

## Loading required package: grid

## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

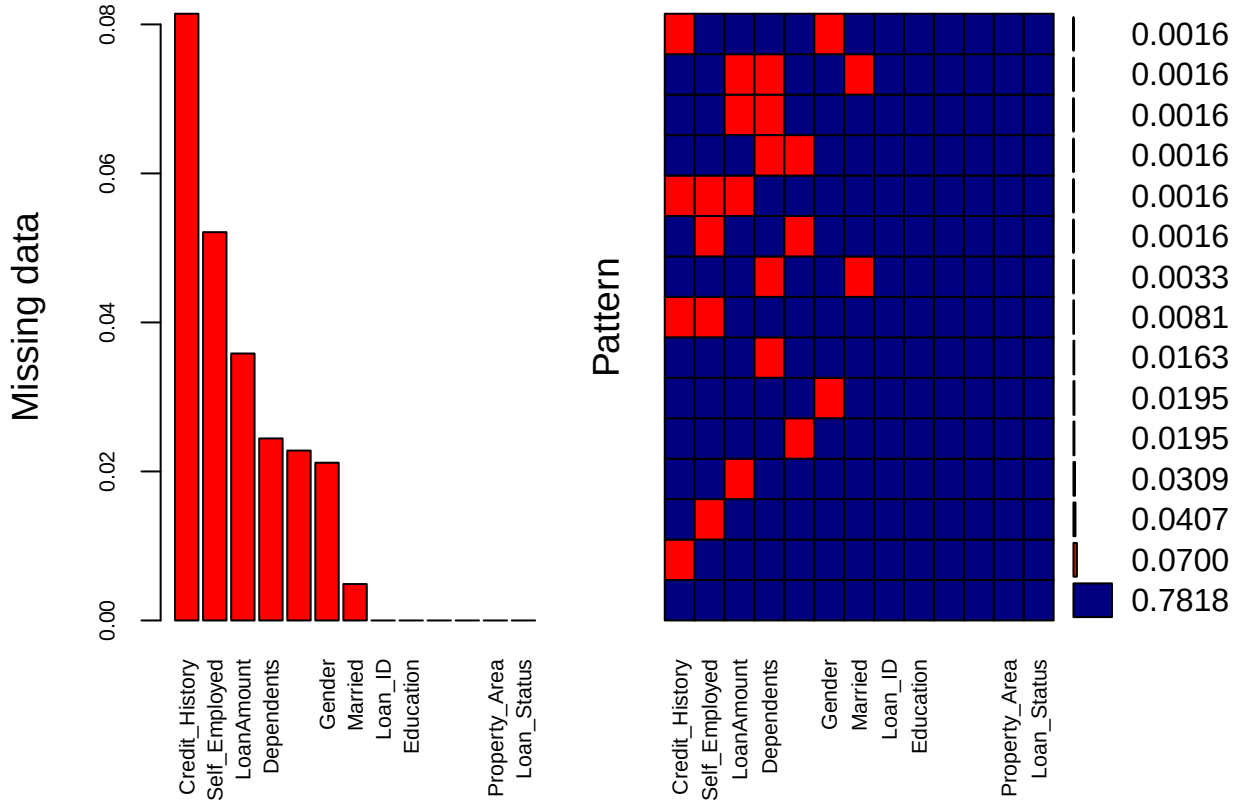
```
## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##           Please use the package to use the new (and old) GUI.

## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

```
mice_plot <- aggr(tr, col=c('navyblue','red'),
                  numbers=TRUE, sortVars=TRUE,
                  labels=names(tr), cex.axis=.7,
                  gap=3, ylab=c("Missing data","Pattern"))
```

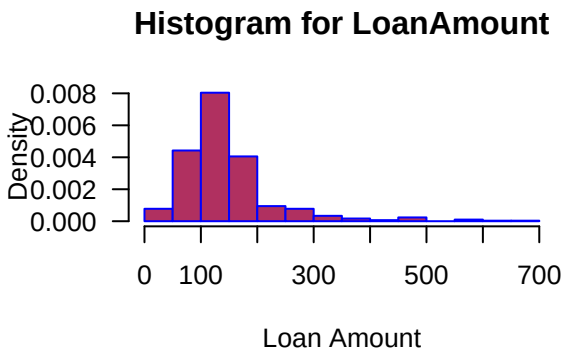



```
##
##  Variables sorted by number of missings:
##           Variable       Count
##     Credit_History 0.081433225
##      Self_Employed 0.052117264
```

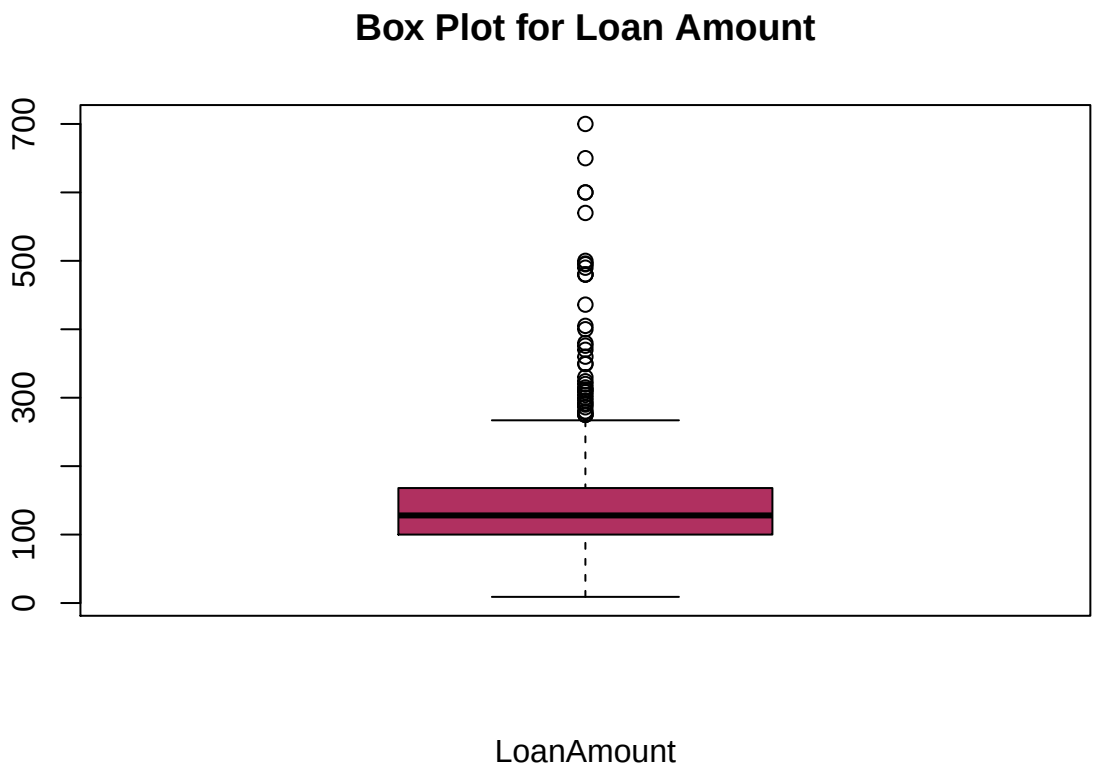
```
##          LoanAmount 0.035830619
##          Dependents 0.024429967
##   Loan_Amount_Term 0.022801303
##              Gender 0.021172638
##             Married 0.004885993
##             Loan_ID 0.000000000
##           Education 0.000000000
##    ApplicantIncome 0.000000000
##  CoapplicantIncome 0.000000000
##      Property_Area 0.000000000
##        Loan_Status 0.000000000
```

From the chart and the table, there are seven variables that have missing data. Now, it's the time to give a look at the distribution of the data. We will start with the numerical variables: Loan Amount and ApplicantIncome: Below are the histograms and the boxplots of the loan amount and the applicant income variables:

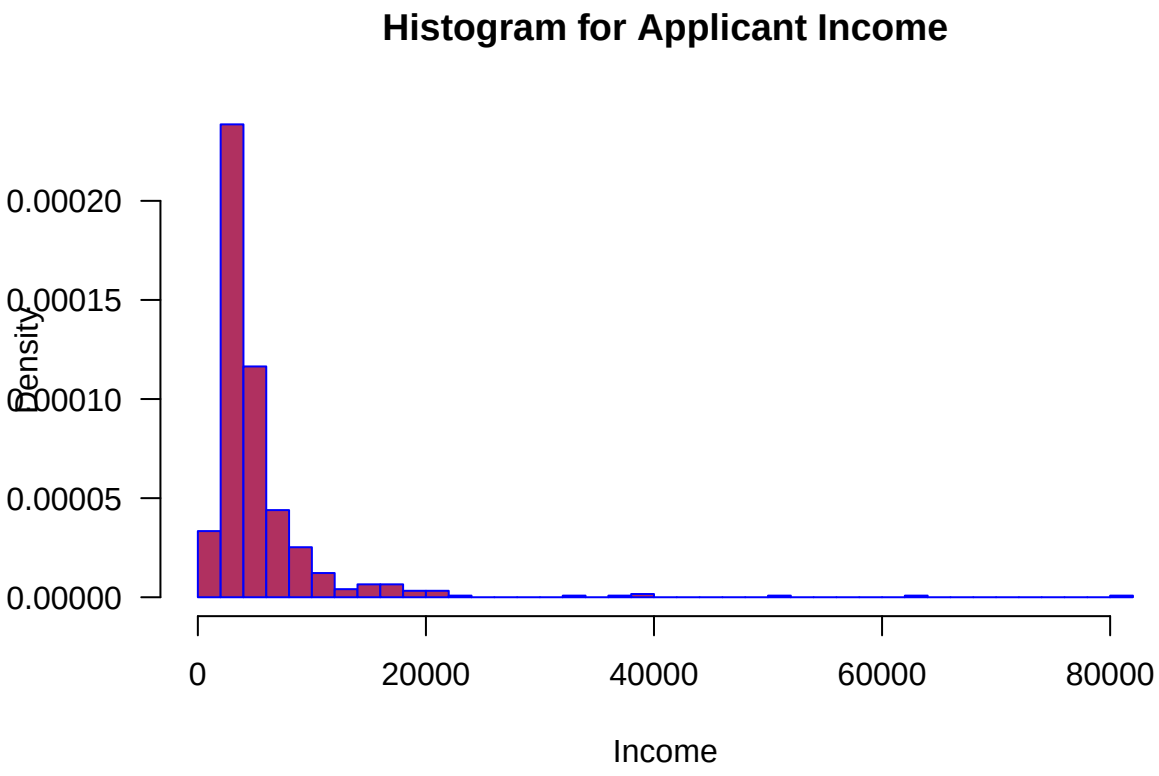
```
par(mfrow=c(2,2))
hist(tr$LoanAmount,
     main="Histogram for LoanAmount",
     xlab="Loan Amount",
     border="blue",
     col="maroon",
     las=1,
     breaks=20, prob = TRUE)
```



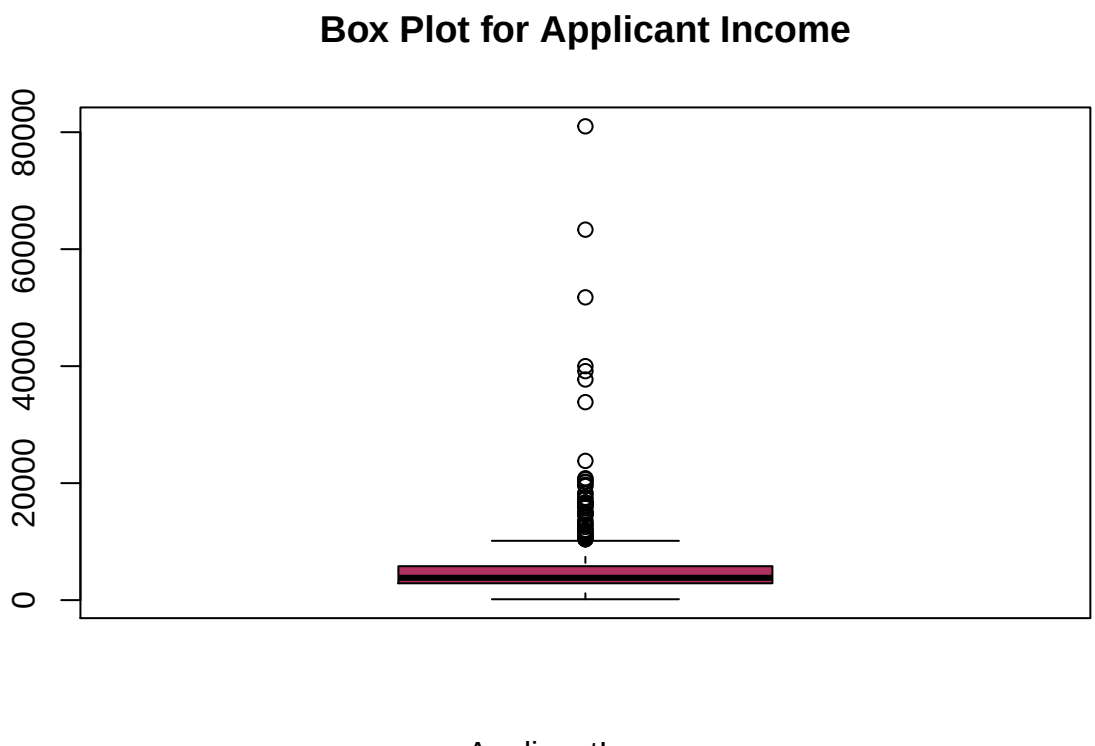
```r
boxplot(tr$LoanAmount, col='maroon',xlab = 'LoanAmount', main = 'Box Plot for Loan Amount')
```




```r
hist(tr$ApplicantIncome,
     main="Histogram for Applicant Income",
     xlab="Income",
     border="blue",
     col="maroon",
     las=1,
     breaks=50, prob = TRUE)
```




```
boxplot(tr$ApplicantIncome, col='maroon',xlab = 'ApplicantIncome', main = 'Box Plot for Applicant Income
```



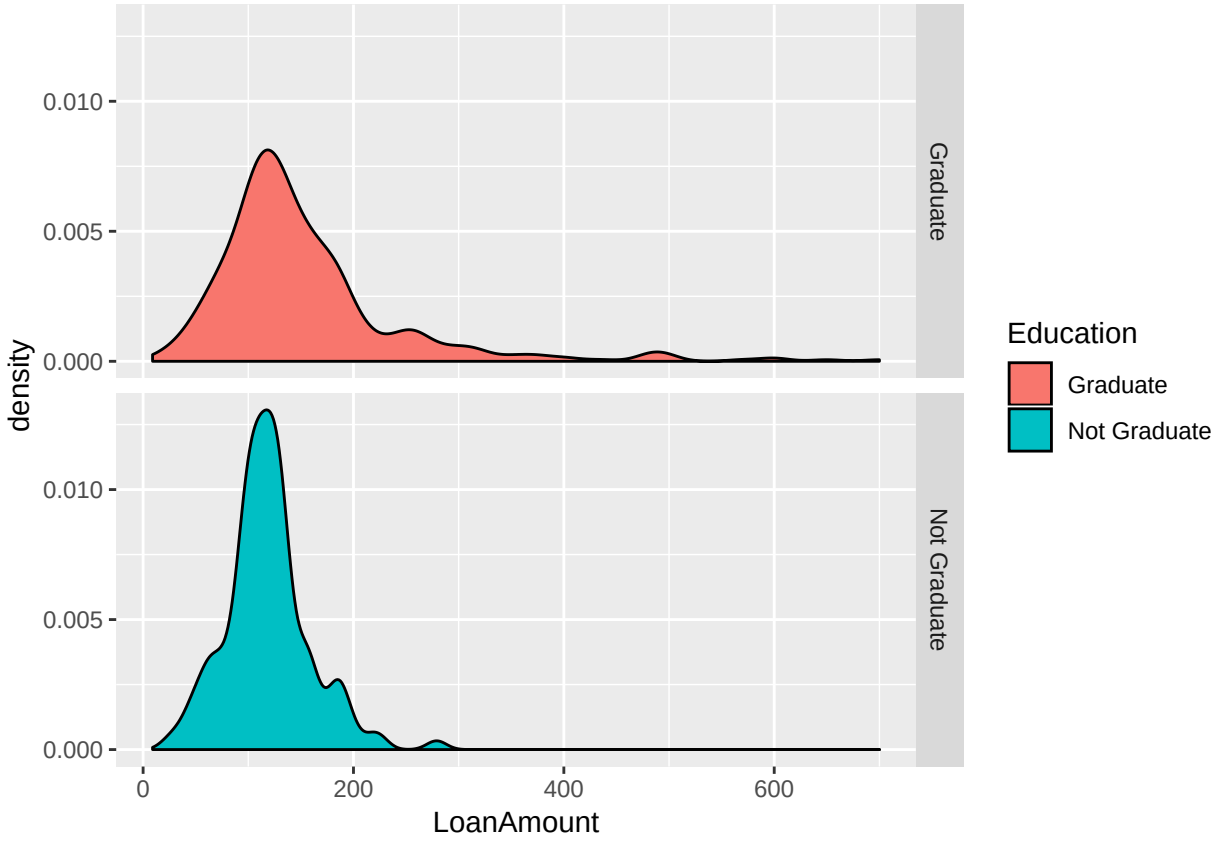
Here we notice that there are few extreme values in both variables. Let's also examine if the applicant' loan amounts distribution is affected by their educational level:

```
library(ggplot2)
data(tr, package="lattice")
```

```
## Warning in data(tr, package = "lattice"): data set 'tr' not found
```

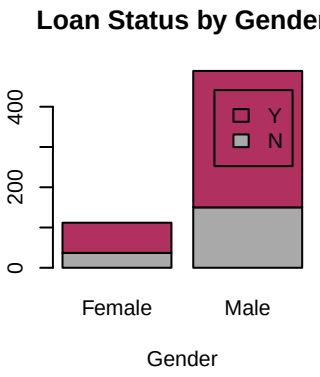
```
ggplot(data=tr, aes(x=LoanAmount, fill=Education)) +
  geom_density() +
  facet_grid(Education~.)
```

```
## Warning: Removed 22 rows containing non-finite values (stat_density).
```

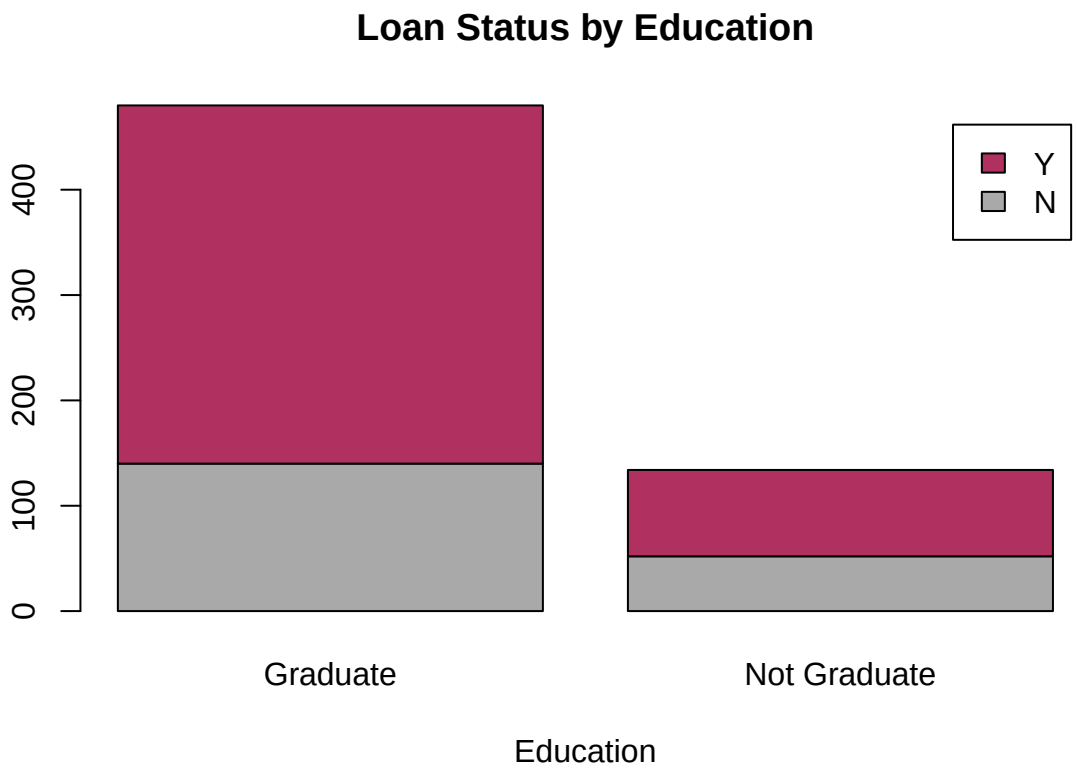


We note that graduates have more outliers and their loan amount distribution is wider. Now let's give a look at the categorical variables in the dataset:

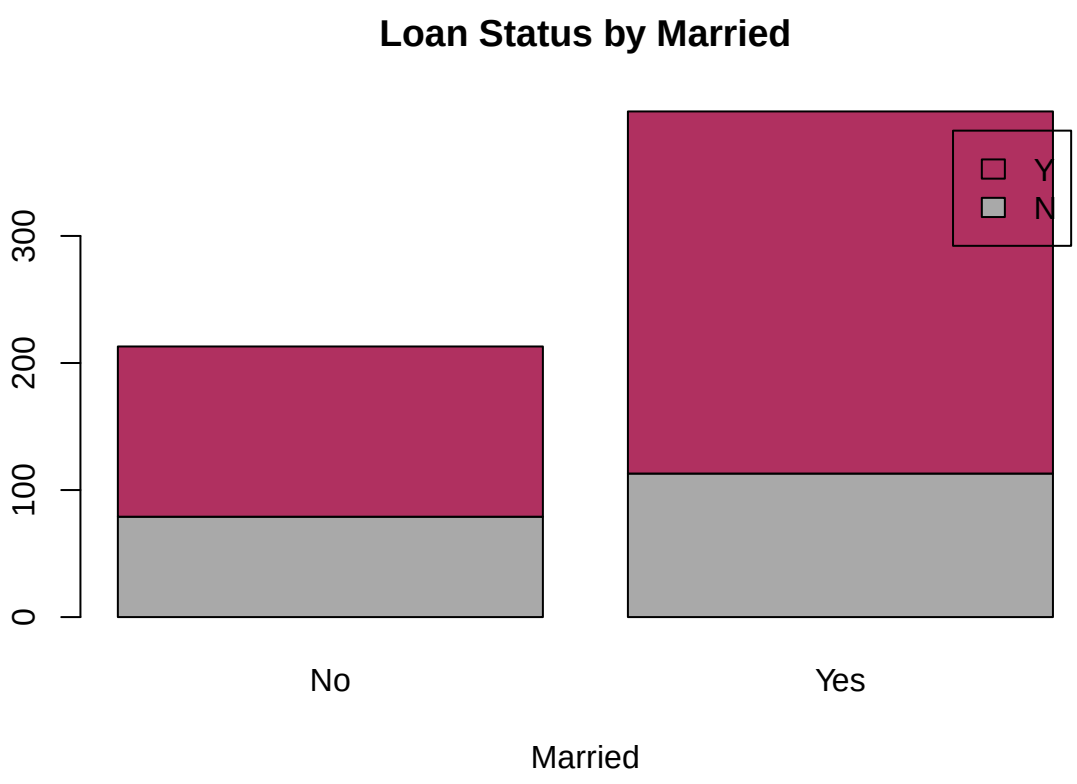
```
par(mfrow=c(2,3))
counts <- table(tr$Loan_Status, tr$Gender)
barplot(counts, main="Loan Status by Gender",
        xlab="Gender", col=c("darkgrey","maroon"),
        legend = rownames(counts))
```



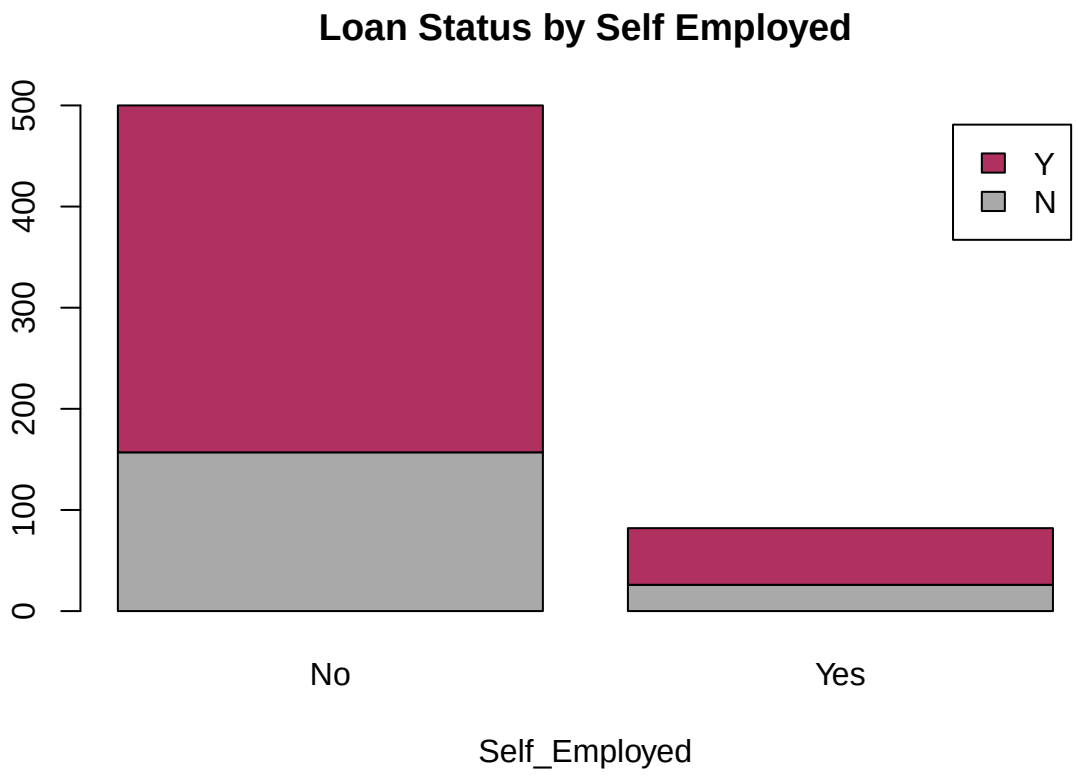

```
counts2 <- table(tr$Loan_Status, tr$Education)
barplot(counts2, main="Loan Status by Education",
        xlab="Education", col=c("darkgrey","maroon"),
        legend = rownames(counts2))
```



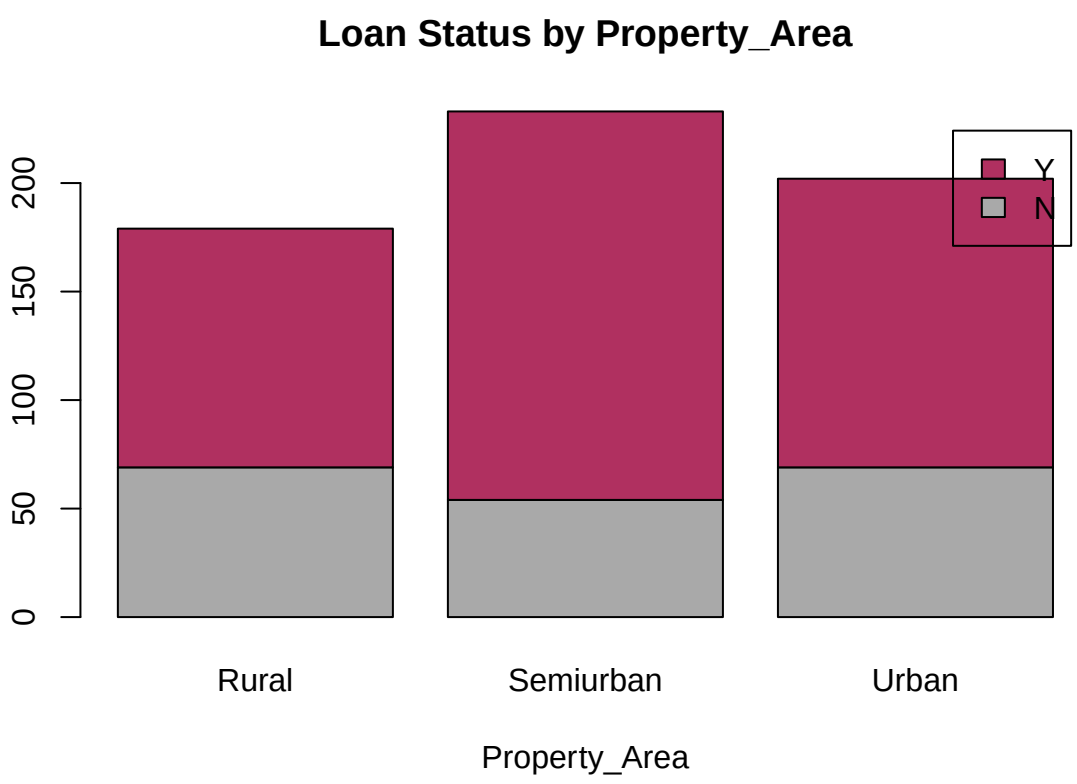

```
counts3 <- table(tr$Loan_Status, tr$Married)
barplot(counts3, main="Loan Status by Married",
        xlab="Married", col=c("darkgrey","maroon"),
        legend = rownames(counts3))
```



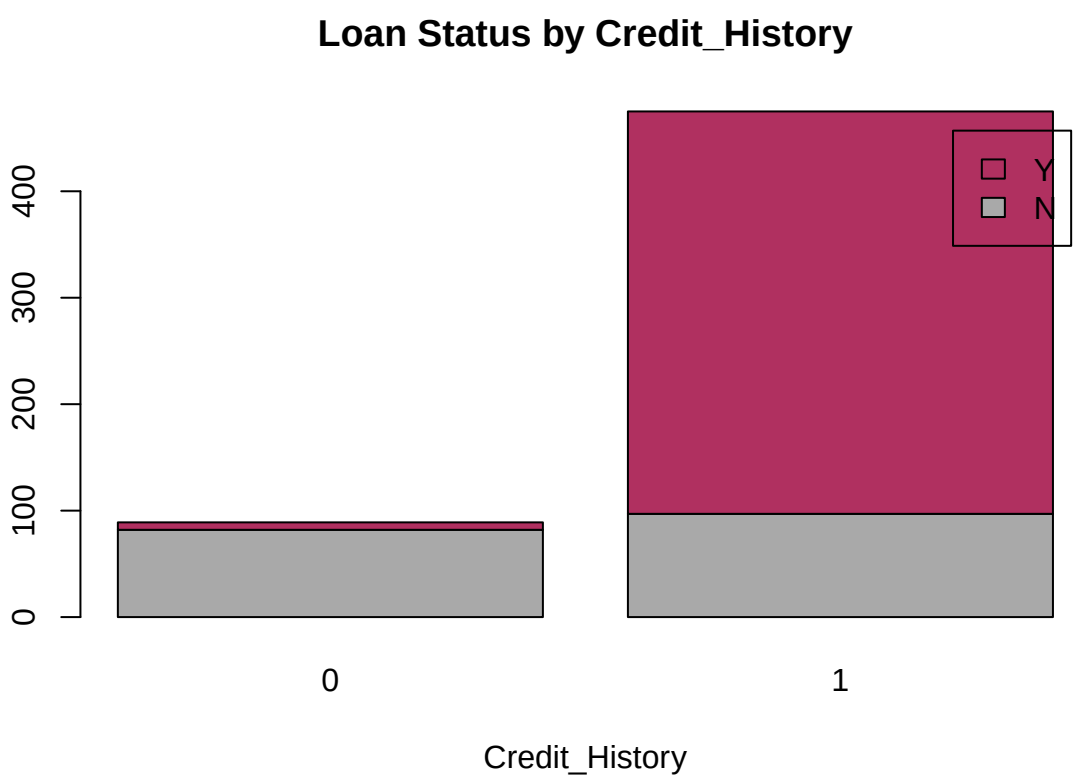

```
counts4 <- table(tr$Loan_Status, tr$Self_Employed)
barplot(counts4, main="Loan Status by Self Employed",
        xlab="Self_Employed", col=c("darkgrey","maroon"),
        legend = rownames(counts4))
```




```
counts5 <- table(tr$Loan_Status, tr$Property_Area)
barplot(counts5, main="Loan Status by Property_Area",
        xlab="Property_Area", col=c("darkgrey","maroon"),
        legend = rownames(counts5))
```




```
counts6 <- table(tr$Loan_Status, tr$Credit_History)
barplot(counts6, main="Loan Status by Credit_History",
        xlab="Credit_History", col=c("darkgrey","maroon"),
        legend = rownames(counts5))
```



If we look at the Gender graph, we note that males have more records and more than half of the applicants' applications have been approved. There are less female applicants but still more than half of their applications have been approved. We look at the other charts with the same eye to evaluate how each category performed in regards to the approval of the loan applications.

Tidying the data

Now that we've identified several errors in the data set, we need to fix them before we continue with our analysis. Let's review the issues:

There are missing values in some variables. Based on the importance of the variables, we will decide on the method to use.Looking at the distributions of the data, we noticed that ApplicantIncome and LoanAmount have outliers. Fixing outliers can be tricky. It's hard to tell if they were caused by measurement error, errors while recording, or if the outliers are real anomaly. If we decide to remove records, we have to document the reason behind this decision.

In this data set, we will assume that missing values are systematic because the missing data are coming in certain variables in a random manner. Also, we note that missing values are on both numerical and categorical data, therefore, we will be using the mice package in R. This package helps in imputing missing values with plausible data values. These values are inferred from a distribution that is designed for each missing data point. In the missing data plot above, we note that 0.78 of the data are not missing any information, 0.07 are missing the Credit_History value, and the remaining ones show other missing patterns.

The mice() function takes care of the imputing process:

```
imputed_Data <- mice(tr, m=2, maxit = 2, method = 'cart', seed = 500)
```

```
##
```

```
##  iter imp variable
##   1   1  Gender  Married  Dependents  Self_Employed  LoanAmount  Loan_Amount_Term  Credit_History
##   1   2  Gender  Married  Dependents  Self_Employed  LoanAmount  Loan_Amount_Term  Credit_History
##   2   1  Gender  Married  Dependents  Self_Employed  LoanAmount  Loan_Amount_Term  Credit_History
##   2   2  Gender  Married  Dependents  Self_Employed  LoanAmount  Loan_Amount_Term  Credit_History

## Warning: Number of logged events: 42
```

It's important to mention that mice stands for multiple imputation by chained equations. The 'm' argument in the function indicates how many rounds of imputation we want to do. For simplicity, I will choose 2. The 'method' argument indicates which of the many methods for imputations we want to use. I chose CART which stands for classification and regression trees. This method work with all variables types, and that's why I chose it. Now let's merge the imputed data into our original dataset. We can do this by using the complete() function:

```
tr <- complete(imputed_Data,2) #here I chose the second round of data imputation
```

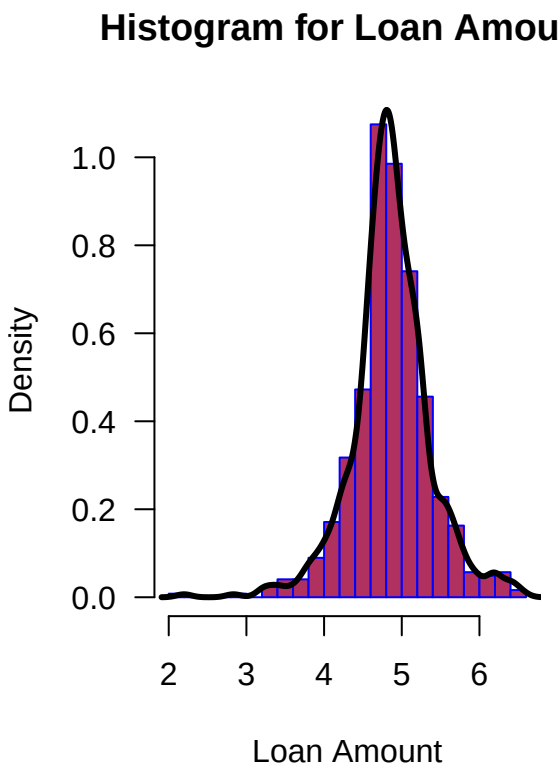
Check missing data again, we note that there is no missing data after the imputation:

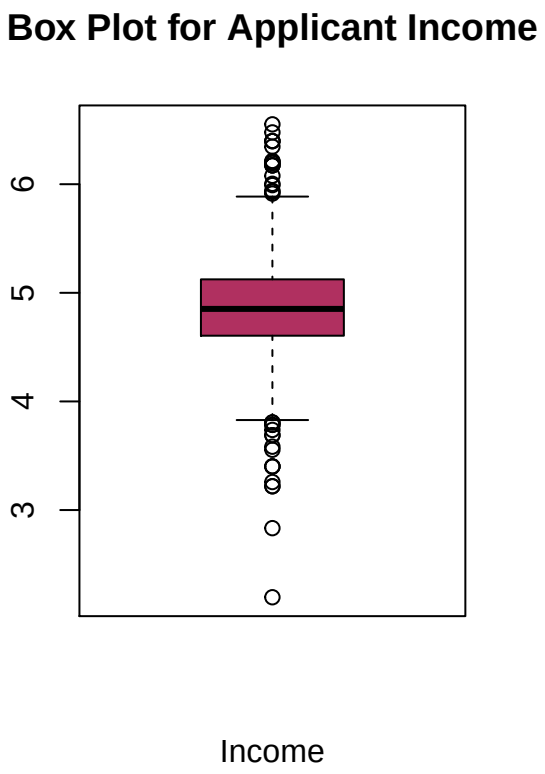
```
sapply(tr, function(x) sum(is.na(x)))
```

```
##           Loan_ID            Gender           Married        Dependents
##                 0                 0                 0                 0
##         Education     Self_Employed   ApplicantIncome CoapplicantIncome
##                 0                 0                 0                 0
##        LoanAmount  Loan_Amount_Term    Credit_History     Property_Area
##                 0                 0                 0                 0
##       Loan_Status
##                 0
```

It's time to treat the extreme values. Looking at the LoanAmount variable, we guess that extreme values are possible as some customers, for some reason, may want to apply for higher loan amounts. We will perform the log transformation to normalize the data:

```
tr$LogLoanAmount <- log(tr$LoanAmount)
par(mfrow=c(1,2))
hist(tr$LogLoanAmount,
     main="Histogram for Loan Amount",
     xlab="Loan Amount",
     border="blue",
     col="maroon",
     las=1,
     breaks=20, prob = TRUE)
lines(density(tr$LogLoanAmount), col='black', lwd=3)
boxplot(tr$LogLoanAmount, col='maroon',xlab = 'Income', main = 'Box Plot for Applicant Income')
```



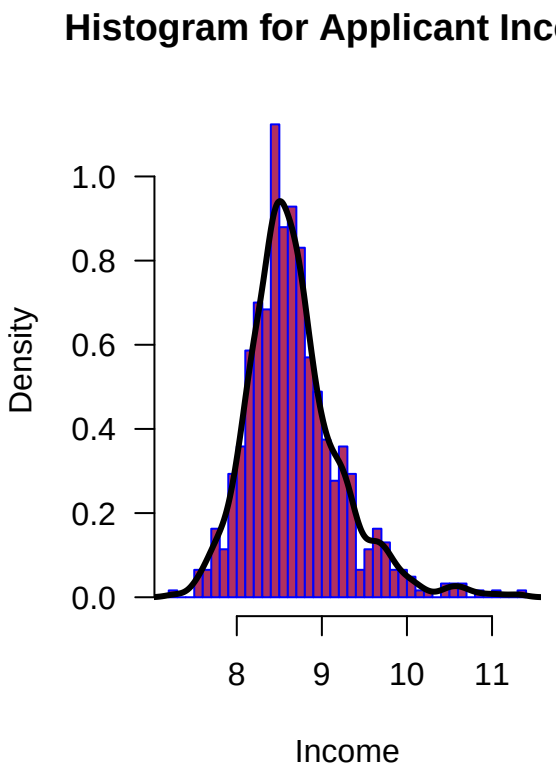


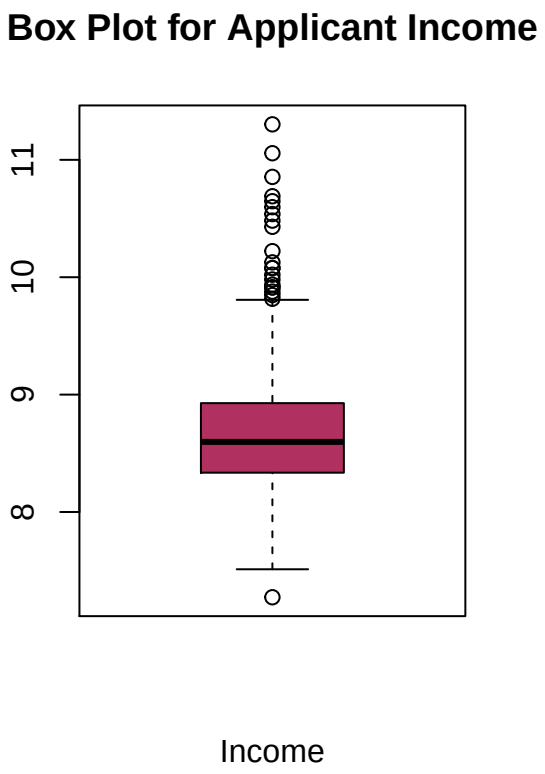
Now the distribution looks closer to normal and effect of extreme values has significantly subsided.

Coming to ApplicantIncome, it will be a good idea to combine both ApplicantIncome and Co-applicants as total income and then perform log transformation of the combined variable.

we will use the CART imputation method. If we know that the values for a measurement fall in a certain range, we can fill in empty values with the average of that measurement.

```
tr$Income <- tr$ApplicantIncome + tr$CoapplicantIncome
tr$ApplicantIncome <- NULL
tr$CoapplicantIncome <- NULL
tr$LogIncome <- log(tr$Income)
par(mfrow=c(1,2))
hist(tr$LogIncome,
     main="Histogram for Applicant Income",
     xlab="Income",
     border="blue",
     col="maroon",
     las=1,
     breaks=50, prob = TRUE)
lines(density(tr$LogIncome), col='black', lwd=3)
boxplot(tr$LogIncome, col='maroon',xlab = 'Income', main = 'Box Plot for Applicant Income')
```





We see that the distribution is better and closer to a normal distribution.

Building Predictive Models (Analysis)

Now it's the time to make the next big step in our analysis which is splitting the data into training and test sets.

A training set is the subset of the data that we use to train our models but the test set is a random subset of the data which are derived from the training set. We will use the test set to validate our models as un-foreseen data.

In a sparse data like ours, it's easy to overfit the data. Overfit in simple terms means that the model will learn the training set that it won't be able to handle most of the cases it has never seen before. Therefore, we are going to score the data using our test set. Once we split the data, we will treat the testing set like it no longer exists. Let's split the data:

```
set.seed(42)
y <- tr$Loan_Status
test_index <- createDataPartition(y, times = 1, p = 0.7, list = FALSE)
```

```
## Error in createDataPartition(y, times = 1, p = 0.7, list = FALSE): could not find function "createDa
```

```
sample <- sample.int(n = nrow(tr), size = floor(.70*nrow(tr)), replace = F)
trainnew <- tr[test_index, ]
```

```
## Error in `[.data.frame`(tr, test_index, ): object 'test_index' not found
```

```
testnew  <- tr[-test_index, ]
```

```
## Error in `[.data.frame`(tr, -test_index, ): object 'test_index' not found
```

Logistic Regression

Firstly, let's train our data with logistic regression. We will first train with credit history as clients who has borrowed money before are usually cleared from background check and the credit lines are more likely to be approved.

```
fit_glm <- glm(Loan_Status ~ Credit_History, data=trainnew, family = "binomial")
```

```
## Error in is.data.frame(data): object 'trainnew' not found
```

```
p_hat_glm <- predict(fit_glm, testnew)
```

```
## Error in predict(fit_glm, testnew): object 'fit_glm' not found
```

```
y_hat_glm <- factor(ifelse(p_hat_glm > 0.5, "Y", "N"))
```

```
## Error in ifelse(p_hat_glm > 0.5, "Y", "N"): object 'p_hat_glm' not found
```

```
confusionMatrix(data = y_hat_glm, reference = testnew$Loan_Status)
```

```
## Error in confusionMatrix(data = y_hat_glm, reference = testnew$Loan_Status): could not find function
```

to further improve the prediction, we can include other variables such as credit_history, education, employment, property owned, income and the amount of loan requested.

```
fit_glm <- glm(Loan_Status ~ Credit_History+Education+Self_Employed+Property_Area+LogLoanAmount+
                    LogIncome, data=trainnew, family = "binomial")
```

```
## Error in is.data.frame(data): object 'trainnew' not found
```

```
p_hat_glm <- predict(fit_glm, testnew)
```

```
## Error in predict(fit_glm, testnew): object 'fit_glm' not found
```

```
y_hat_glm <- factor(ifelse(p_hat_glm > 0.5, "Y", "N"))
```

```
## Error in ifelse(p_hat_glm > 0.5, "Y", "N"): object 'p_hat_glm' not found
```

```
confusionMatrix(data = y_hat_glm, reference = testnew$Loan_Status)
```

```
## Error in confusionMatrix(data = y_hat_glm, reference = testnew$Loan_Status): could not find function
```

We can see that the accuracy improved from 82% to 82.5%.

A larger p-value indicates that changes in the predictor are not associated with changes in the dependent variable and that it's insignificant. The p-value for all the variables are so small and therefore, it's significant.

Random Forest

Another popular model that is usually used for prediction is the random forest model. A random forest is an ensemble learning approach to supervised learning. This approach develops multiple predictive models, and the results are aggregated to improve classification.

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
fit_rf <- randomForest(Loan_Status ~ Credit_History+Education+Self_Employed+Property_Area+LogLoanAmount+
                    LogIncome, data = trainnew)
```

```
## Error in eval(m$data, parent.frame()): object 'trainnew' not found
```

```
y_hat_rf <- predict(fit_rf, testnew)
```

```
## Error in predict(fit_rf, testnew): object 'fit_rf' not found
```

```
confusionMatrix(data = y_hat_rf, reference = testnew$Loan_Status)
```

```
## Error in confusionMatrix(data = y_hat_rf, reference = testnew$Loan_Status): could not find function "
```

As shown above, the accuracy of the random forest is only 79.2%, which is worst than the logistic regression model. In this case, the random forest seems to be not an suitable model for predicting the loan. However, the random forest function grew 500 traditional decision trees by sampling 429 observations with replacement from the training sample. Random forests provides natural measure of variable importance. The relative importance measure specified by type=2 option is the total decrease in node impurities from splitting on that variable, averaged over all trees. In our trees,the most important variable is Credit_History and the least is Self_Employed. We have finally measured the accuracy for the training sample and applied the prediction to the test sample. We note that the accuracy for both are less than the decision tree's accuracy.

Also, note that the p-value is larger than the previous model but still at a significant level.

We will run the same model but this time we will select the highest three in importance:

```r
set.seed(42)
fit.forest2 <- randomForest(Loan_Status ~ Credit_History+LogLoanAmount+
                              LogIncome, data=trainnew,importance=TRUE)
```

```
## Error in eval(m$data, parent.frame()): object 'trainnew' not found
```

```r
fit.forest2
```

```
## Error in eval(expr, envir, enclos): object 'fit.forest2' not found
```

```r
y_hat_rf <- predict(fit.forest2, testnew)
```

```
## Error in predict(fit.forest2, testnew): object 'fit.forest2' not found
```

```r
confusionMatrix(data = y_hat_rf, reference = testnew$Loan_Status)
```

```
## Error in confusionMatrix(data = y_hat_rf, reference = testnew$Loan_Status): could not find function "
```

By selecting the important variable, we could futher improved our accuracy to 82%. Random forest is a very useful model that take into account the importance of certain variable. It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier and it runs efficiently on large databases.

####Training Various Models Here I will provide the a quick way to measure the suitability of the various model based on accuracy alone.

```r
###NOTE: this codes will take a long time to finish running. I don't recommend you to run it as I have
#models <- c("glm", "lda",  "naive_bayes",  "svmLinear", "qda",
#                "knn", "kknn",
#                "rf", "ranger",  "wsrf", "Rborist",
#                "avNNet", "mlp", "monmlp",
#                "adaboost", "gbm",
#                "svmRadial", "svmRadialCost", "svmRadialSigma")
#library(caret)
#library(dslabs)
#set.seed(1)

#fits <- lapply(models, function(model){
#   print(model)
#   train(Loan_Status ~ Credit_History+Education+Self_Employed+Property_Area+LogLoanAmount+
#                     LogIncome , method = model, data = trainnew)
#})

#names(fits) <- models

#pred <- sapply(fits, function(object)
#   predict(object, newdata = testnew))
```

```
## Error in is.data.frame(x): object 'pred' not found
```

```
## Error in eval(expr, envir, enclos): object 'acc' not found
```

####Conclusion From various model that we trained above, the naive_bayes model seems to be the most suitable model for our dataset as it produced the highest accuracy of 82.5%. However, please note that the results above only give you a brief results on the most suitable model to look at as it we merely evaluate it based on the accuracy. Other factors might need to be consider when doing the data prediction such as its sensitivity and precision. Some model like random forest could also generate better result by ranking the importance of its variables.

Overall, based on the accuracy alone, naive_model