Home Loan Eligibility Prediction*

Application of the random forest model in Machine Learning

Linzi Guan(lg3183): lg3183@columbia.edu

13 March 2022

Abstract

With the purpose to reduce the time and human costs and improve the accuracy in loan eligibility validation process, an automatic loan egibility prediction model is developed in this paper with the home loan dataset from Kaggle website, which takes a customer's demongraphic features as inputs and makes a prediction on whether a loan would be issued for that customer as outputs. The paper finally arrives at a random forest model with accuracy of 81.25%, as an improvement from the simple classification tree model. This study appeals for home loan companies and banks to develop an automatic loan eligibility model and for customers to determine whether they would successfully get the loan.

Keywords: Loan Egibility Prediction, Random Forest, Machine Learning

1 Introduction

It has been a time consuming problem for the home loan companies and banks to manually validate the loan eligibility which would incur large amounts of human costs through heavy due diligence works to check the customer credibility. Moreover, once there are frauds in the identification process, such as agency problems during which the responsible agent is bribed to issue problematic loans that are never returned, there would be huge losses incurred to the institutions issuing those loans. Therefore, a machine learning model is developed to take consideration of automatically determining whether the incoming customer meets the requirements for loans or not for institutions and assisting customers to predict whether they can be loaned with their backgrounds. In this paper, a tree is initially used for classification and later random forest model is developed for improvement on predicting whether the customer is going to be offered loan or not.

The data set used is called *Home Loan Predictions*, which is retrieved from the Kaggle website. The data describes the customer profiles and their loan status from a home loan company called Housing Finance company. Only the data set with the labelled loan status outcome is used between the two provided data sets, as the other one without labels does not help with the training and testing process of the model.

Based on the data set, a classification tree is firstly built and a random forest model is then developed for predicting the loan status and the feature importance analysis is conducted. The model has an accuracy of 81.25% and the most important features found are credit history, applicant income, loan amount and coapplicant income.

The paper would then be structured by a full disclosure, preprocess and analysis of the data set for the model in the Data Section, the detailed process of building the model and the results obtained from the model in the Model and Results Section, some insights and discussions generated from the model in the Discussion and further discussions in the Limitation and Next Steps.

*Dataset retrieved: GOVARDHAN C. 2019. Home Loan Predictions, Version 1. Retrieved March 13, 2022 from https://www.kaggle.com/gavincanacam/home-loan-predictions

2 Data

Our data is of customer profiles and their loan status from Housing Finance company retrieved from the Kaggle website. There are 614 customer profiles with 13 variables(Refer to Appendix 1).

Data Variables

The raw data set contains 13 variables including Loan_ID, Gender, Married, Dependents, Education, Self_Employed, ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, Credit_History, Property_Area and Loan_Status (Refer to Appendix 2) with 8 categorical variables and 6 numerical variables. And detailed explanations of those variables are as follows:

Categorical Variables:

Loan_ID identifies the customer profile and represents customer. Each customer has one unique loan id; *Gender* represents the self-reported gender of the customer and takes two values: Male and Female; *Married* represents the self-reported marital status of the customer and takes two values: Yes(standing for married) and No(standing for non-married); *Dependents* represents the number of dependents the customer has and it takes values of “0”, “1”, “2”, “3+”; *Education* represents the self-reported education level and takes two values: Graduate and Not Graduate; *Self_Employed* represents whether the customer is self employed and takes two values: Yes and No; *Property_Area* represents the property area that the house is located; *Loan_Status* represents whether the loan is issued for that customer, and it takes two values: Y for yes and N for no.

Numerical Variables:

ApplicantIncome records the income of the customer; *CoapplicantIncome* records the income of the co applicant of the customer; *LoanAmount* recors the amount that the customer is applying for; *Loan_Amount_Term* represents the term of the loan; *Credit_History* represents whether the customer has been paid back the loan before or not, which takes two values: 1 for yes and 0 for no.

Among those variables, the *Loan_Status* variable is the dependent variable of prediction and others except Loan_ID are independent variables that taken into inputs.

Data Preprocessing

To avoid further interruptions in the model, missing values are firstly removed from the data set. As Loan_ID does not help for the model prediction, it is dropped from the data set. The tree and random forest model can take categorical variables as inputs and it would be more interpretable if credit_history takes values of “yes” and “no” instead of being a numerical variable, so it is changed to “yes” if it is 1 and “no” if 0. Dependents is changed to an ordered dummy of 0 if “0” dependent, 1 if “1”, 2 if “2” and 3 if “3+”. Similarly, the property area is changed to an ordered dummy of 0 if “rural”, 1 if “Semiurban” and 2 if “urban”. For variable Loan_Amount_Term, 360 is a clear boundary, so it is changed to a categorical variable of the term of loans fewer than 360 months as < 360 and else as $>= 360$. Meanwhile, there are empty strings recorded in some variables, which are removed as missing values. For easier interpretation, the status is transformed to “yes” from “Y” and “no” from “N”. Moreover, all characters are transformed into factors for further model usage as categorical variables. Finally, there are 480 customer profiles for 12 variables(Refer to Appendix 3).

3 Exploratory Data Analysis

After having a brief exploratory data analysis of both the tables(Refer to Appendix 4) and bar plot analysis(Refer to Appendix 5), there are some relationships observed between those variables and the loan status. And the relationship can be seen in Figure 1 as follows.

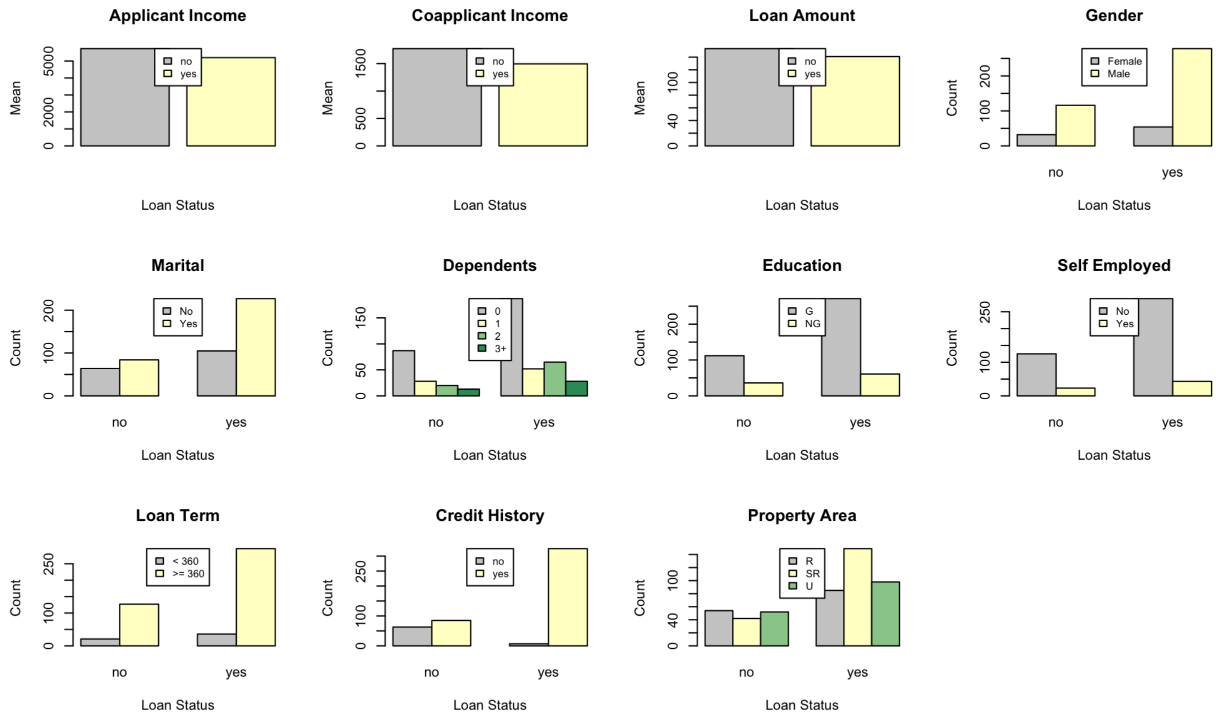


Figure 1: Relationship between variables.

4 Model and results

Making a prediction on whether the customer can be offered loan is a classification problem, so firstly, a simple classification tree model is built as the start. The input variables are Gender, Married, Dependents, Education, Self-Employed, ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, Credit_History, and Property Area and the output variable is Loan_Status.

The train test split is set to be 7:3 randomly and both gini and deviance methods are applied separately in two tree models with minimum number of cases specified at 25. Using the deviance method for the simple tree model, the prediction accuracy is 76.39% and the top three important features are: credit history, marital status and property area and especially customers with a credit history of being rejected loans are very likely to be rejected again in another application(Refer to Appendix 6.1). Using the gini method for the simple tree model, the prediction accuracy is 75.69% and the top three important features are: Dependents, Coapplicant income and credit history(Refer to Appendix 6.2).

As an improvement of simple tree model, a random forest model taking trees on an aggregate level and automatically conducting the feature selection to de-correlate features is then developed for prediction with the same input and output variables. Using the same train-test split and subsets as the simple tree model, a random forest model with 1000 trees is trained on the training set and then tested on the testing set. The confusion matrix is then computed as in Figure 2 (Refer to Appendix 7).

As can be seen from the figure, the accuracy rate is 81.25% in the random forest model. And the feature importance of the random forest model is shown in Figure 3. From both the mean decrease accuracy and the mean decrease gini graphs, it can be inferred that the most features are credit history, applicant income, loan amount and coapplicant income.

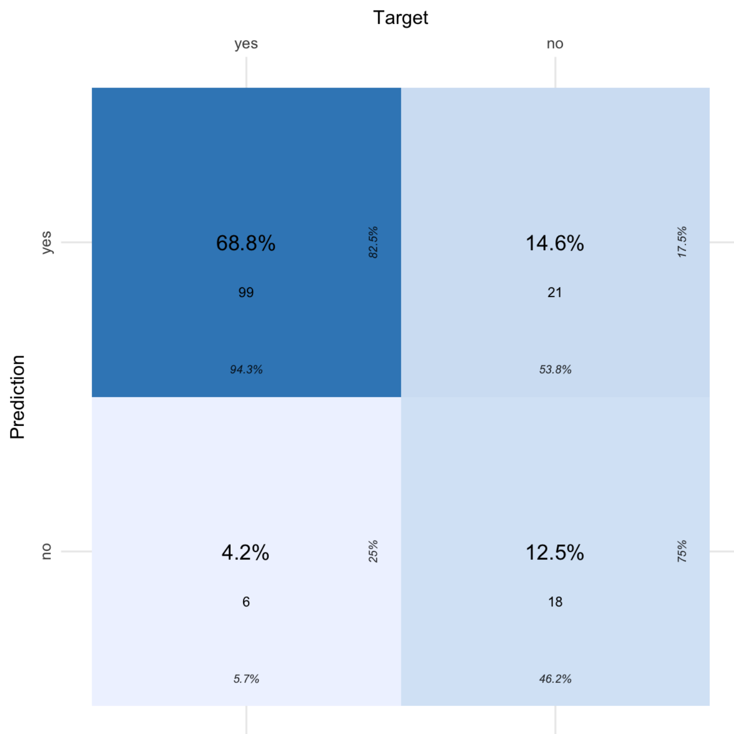


Figure 2: Confusion Matrix of the random forest model.

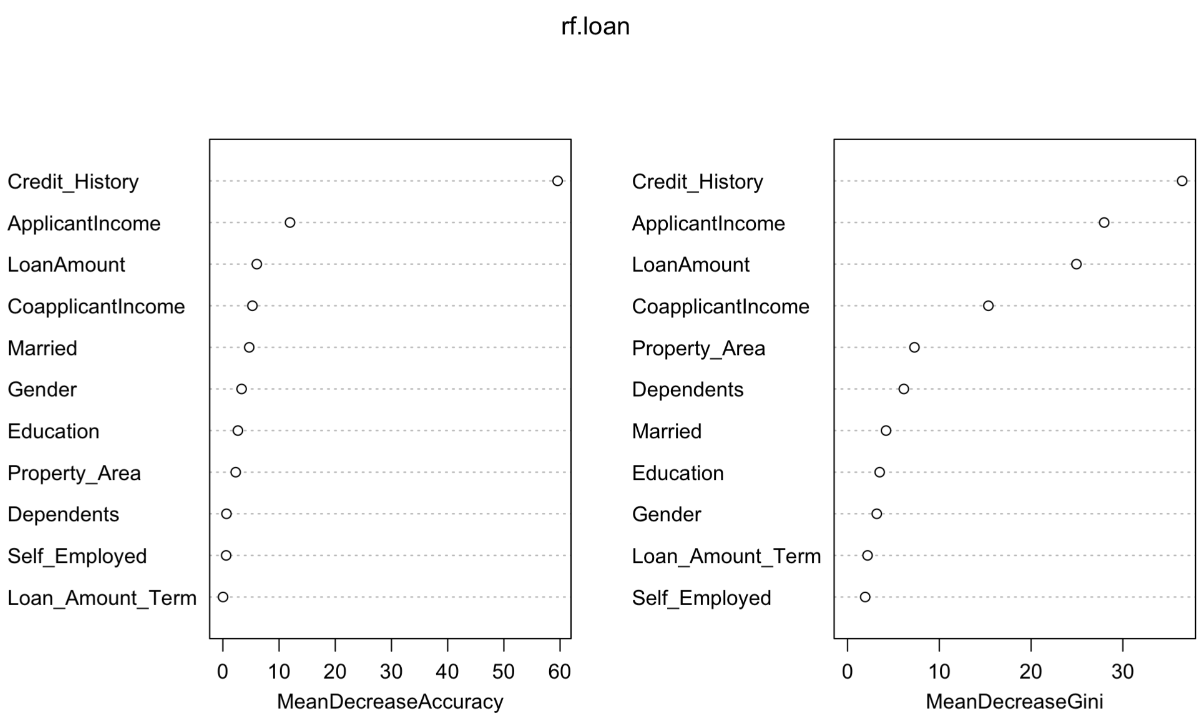


Figure 3: Feature Importance of the random forest model.

5 Discussions, weaknesses and next steps

5.1 Improvement of random forest model from simple tree model

As can be seen from the previous model results that the random forest model has a significant increase in model prediction accuracy score, from 76.39% of the tree model with "deviance" method and 75.69% of the tree model with "gini" method, to the 81.25% using the 1000-tree random forest model. Taking consideration of the algorithm behind random forest, this complies to the intuition that bagging the trees(bootstrapping with replacements) would provide a more aggregate and accurate prediction with de-correlating the variables as random forest randomly selects a subset of variables for each tree (Refer to Appendix 7), which keeps a low bias in the model but also realizes a large reduction in variance.

5.2 Weaknesses

Through the exploratory data analysis, we found that the relationships between some variables and the loan status are counter-intuitive, including applicantincome and coapplicantincome. Intuitively, the larger amounts of salary the customer earns and the coapplicant earns means the higher ability of them to pay back the loans, so customers that earn more should be more easily to get loans and the average applicant and coapplicant incomes of customers successfully offered loans should be higher than those failed to be get loans. However, the table shows opposite way(Refer to Appendix 4). There are several possible reasons for this and one is that there are outliers in the dataset that drags the average of applicants and coapplicants up(Refer to Appendix 3). And also, the dataset is moderately small of only 480 observations, which might be representative of the whole customer cases. Moreover, the decision process is a multi-stage taking many decisive variables into consideration, the lack of income can be supported by other alternative supporting documents such as real estate and others, so only the income cannot reflect the ability of payment ability.

5.3 Next steps

As discussed above, possible outliers in the applicant incomes and coapplicant incomes should be removed from the data set to avoid misrepresentations. Then, this is a rather small dataset, possible larger dataset could be obtained and augmented in the future. Moreover, cross validation could be applied further to decrease the overfitting problems.

Appendix

Appendix 1: Load the data set into data frame

```r
loan <- read.csv("Train_Loan_Home.csv", header = TRUE, sep = ",")
```

Appendix 2: Summary the raw data frame

```r
summary(loan)
```

```
##    Loan_ID             Gender             Married            Dependents       
##  Length:614         Length:614         Length:614         Length:614        
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                             
##                                                                             
##                                                                             
##                                                                             
##   Education         Self_Employed      ApplicantIncome CoapplicantIncome
##  Length:614         Length:614         Min.   :  150   Min.   :    0    
##  Class :character   Class :character   1st Qu.: 2878   1st Qu.:    0    
##  Mode  :character   Mode  :character   Median : 3812   Median : 1188    
##                                        Mean   : 5403   Mean   : 1621    
##                                        3rd Qu.: 5795   3rd Qu.: 2297    
##                                        Max.   :81000   Max.   :41667    
##                                                                         
##    LoanAmount    Loan_Amount_Term Credit_History   Property_Area     
##  Min.   :  9.0   Min.   : 12      Min.   :0.0000   Length:614        
##  1st Qu.:100.0   1st Qu.:360      1st Qu.:1.0000   Class :character  
##  Median :128.0   Median :360      Median :1.0000   Mode  :character  
##  Mean   :146.4   Mean   :342      Mean   :0.8422                     
##  3rd Qu.:168.0   3rd Qu.:360      3rd Qu.:1.0000                     
##  Max.   :700.0   Max.   :480      Max.   :1.0000                     
##  NA's   :22      NA's   :14       NA's   :50                         
##  Loan_Status       
##  Length:614        
##  Class :character  
##  Mode  :character  
##                    
##                    
##                    
##                    
```

Appendix 3: Data preprocessing progress

```r
# remove all missing values from the dataset
loan <- na.omit(loan)
print(dim(loan))
```

```
## [1] 529  13
```

```r
# drop loan id from the dataset
loan <- subset(loan, select = -c(1))
print(dim(loan))
```

```
## [1] 529  12
```

```r
#Overview the gender variable
aggregate(loan$Gender, by=list(loan$Gender), FUN = length)
```

```
##   Group.1   x
## 1          12
## 2  Female  95
## 3    Male 422
```

```r
#Remove empty strings from the variable
loan$Gender <- replace(loan$Gender, loan$Gender == "", NA)
loan <- na.omit(loan)
aggregate(loan$Gender, by=list(loan$Gender), FUN = length)
```

```
##   Group.1   x
## 1  Female  95
## 2    Male 422
```

```r
#Overview the married variable
aggregate(loan$Married, by=list(loan$Married), FUN = length)
```

```
##   Group.1   x
## 1           2
## 2      No 185
## 3     Yes 330
```

```r
#Remove empty strings from the variable
loan$Married <- replace(loan$Married, loan$Married == "", NA)
loan <- na.omit(loan)
aggregate(loan$Married, by=list(loan$Married), FUN = length)
```

```
##   Group.1   x
## 1      No 185
## 2     Yes 330
```

```r
#Overview the dependents variable
aggregate(loan$Dependents, by=list(loan$Dependents), FUN = length)
```

```
##   Group.1   x
## 1          10
## 2       0 289
## 3       1  84
## 4       2  90
## 5      3+  42
```

```r
# change dependents to ordered dummy and remove empty strings
loan$Dependents <- replace(loan$Dependents, loan$Dependents == "0", 0)
loan$Dependents <- replace(loan$Dependents, loan$Dependents == "1", 1)
loan$Dependents <- replace(loan$Dependents, loan$Dependents == "2", 2)
loan$Dependents <- replace(loan$Dependents, loan$Dependents == "3+", 3)
loan$Dependents <- replace(loan$Dependents, loan$Dependents == "", NA)
loan <- na.omit(loan)
loan$Dependents <- as.numeric(loan$Dependents)
aggregate(loan$Dependents, by=list(loan$Dependents), FUN = length)
```

```
##   Group.1   x
## 1       0 289
## 2       1  84
## 3       2  90
## 4       3  42
```

```r
#Overview the education variable
aggregate(loan$Education, by=list(loan$Education), FUN = length)
```

```
##        Group.1   x
## 1     Graduate 402
## 2 Not Graduate 103
```

```r
#Overview the self_emlpoyed variable
aggregate(loan$Self_Employed, by=list(loan$Self_Employed), FUN = length)
```

```
##   Group.1   x
## 1          25
## 2      No 414
## 3     Yes  66
```

```r
#Remove empty strings from the variable
loan$Self_Employed <- replace(loan$Self_Employed, loan$Self_Employed == "", NA)
loan <- na.omit(loan)
aggregate(loan$Self_Employed, by=list(loan$Self_Employed), FUN = length)
```

```
##   Group.1   x
## 1      No 414
## 2     Yes  66
```

```
#Overview the self_employed variable
aggregate(loan$Loan_Amount_Term, by=list(loan$Loan_Amount_Term), FUN = length)
```

```
##   Group.1   x
## 1      36   2
## 2      60   2
## 3      84   3
## 4     120   3
## 5     180  36
## 6     240   2
## 7     300   9
## 8     360 411
## 9     480  12
```

```
#Group the terms shorter than 360 and longer than 360
loan$Loan_Amount_Term <- replace(loan$Loan_Amount_Term, loan$Loan_Amount_Term < 360, "< 360")
loan$Loan_Amount_Term <- replace(loan$Loan_Amount_Term, loan$Loan_Amount_Term >= 360, ">= 360")
aggregate(loan$Loan_Amount_Term, by=list(loan$Loan_Amount_Term), FUN = length)
```

```
##   Group.1   x
## 1   < 360  57
## 2  >= 360 423
```

```
#Overview the credit_history variable
aggregate(loan$Credit_History, by=list(loan$Credit_History), FUN = length)
```

```
##   Group.1   x
## 1       0  70
## 2       1 410
```

```
# change credit_history to "yes" if it is 1 and "no" if 0
loan$Credit_History <- ifelse(loan$Credit_History == 1, "yes", "no")
aggregate(loan$Credit_History, by=list(loan$Credit_History), FUN = length)
```

```
##   Group.1   x
## 1      no  70
## 2     yes 410
```

```
#Overview the property_area variable
aggregate(loan$Property_Area, by=list(loan$Property_Area), FUN = length)
```

```
##     Group.1   x
## 1     Rural 139
## 2 Semiurban 191
## 3     Urban 150
```

```
#change property_area to ordered dummy
loan$Property_Area <- replace(loan$Property_Area, loan$Property_Area == "Rural", 0)
loan$Property_Area <- replace(loan$Property_Area, loan$Property_Area == "Semiurban", 1)
loan$Property_Area <- replace(loan$Property_Area, loan$Property_Area == "Urban", 2)
loan$Property_Area <- as.numeric(loan$Property_Area)
aggregate(loan$Property_Area, by=list(loan$Property_Area), FUN = length)
```

```
##   Group.1   x
## 1       0 139
## 2       1 191
## 3       2 150
```

```r
# Overview the variable loan_status
aggregate(loan$Loan_Status, by=list(loan$Loan_Status), FUN = length)
```

```
##   Group.1   x
## 1       N 148
## 2       Y 332
```

```r
# change loan_status to ordered dummy
loan$Loan_Status <- replace(loan$Loan_Status, loan$Loan_Status == "Y", "yes")
loan$Loan_Status <- replace(loan$Loan_Status, loan$Loan_Status == "N", "no")
aggregate(loan$Loan_Status, by=list(loan$Loan_Status), FUN = length)
```

```
##   Group.1   x
## 1      no 148
## 2     yes 332
```

```r
#Change all categorical variables into dummies
loan$Gender <- as.factor(loan$Gender)
loan$Married <- as.factor(loan$Married)
loan$Education <- as.factor(loan$Education)
loan$Self_Employed <- as.factor(loan$Self_Employed)
loan$Loan_Amount_Term<- as.factor(loan$Loan_Amount_Term)
loan$Credit_History <- as.factor(loan$Credit_History)
loan$Loan_Status <- as.factor(loan$Loan_Status)
```

```r
#print a summary of the processed dataset
summary(loan)
```

```
##     Gender     Married     Dependents            Education   Self_Employed
##  Female: 86   No :169   Min.   :0.0000   Graduate    :383   No :414
##  Male  :394   Yes:311   1st Qu.:0.0000   Not Graduate: 97   Yes: 66
##                         Median :0.0000
##                         Mean   :0.7771
##                         3rd Qu.:2.0000
##                         Max.   :3.0000
##  ApplicantIncome CoapplicantIncome   LoanAmount     Loan_Amount_Term
##  Min.   :  150   Min.   :    0     Min.   :  9.0   < 360 : 57
##  1st Qu.: 2899   1st Qu.:    0     1st Qu.:100.0   >= 360:423
##  Median : 3859   Median : 1084     Median :128.0
##  Mean   : 5364   Mean   : 1581     Mean   :144.7
##  3rd Qu.: 5852   3rd Qu.: 2253     3rd Qu.:170.0
##  Max.   :81000   Max.   :33837     Max.   :600.0
##  Credit_History Property_Area   Loan_Status
##  no : 70        Min.   :0.000   no :148
##  yes:410        1st Qu.:0.000   yes:332
##                 Median :1.000
##                 Mean   :1.023
##                 3rd Qu.:2.000
##                 Max.   :2.000
```

Appendix 4: Exploratory Data Analysis: Tables

```
library(dplyr)

mean_numerical <- loan %>%
  group_by(Loan_Status) %>%
  summarise(mean_income=mean(ApplicantIncome), mean_coincome=mean(CoapplicantIncome), mean_loan= mean(Lo
mean_numerical
```

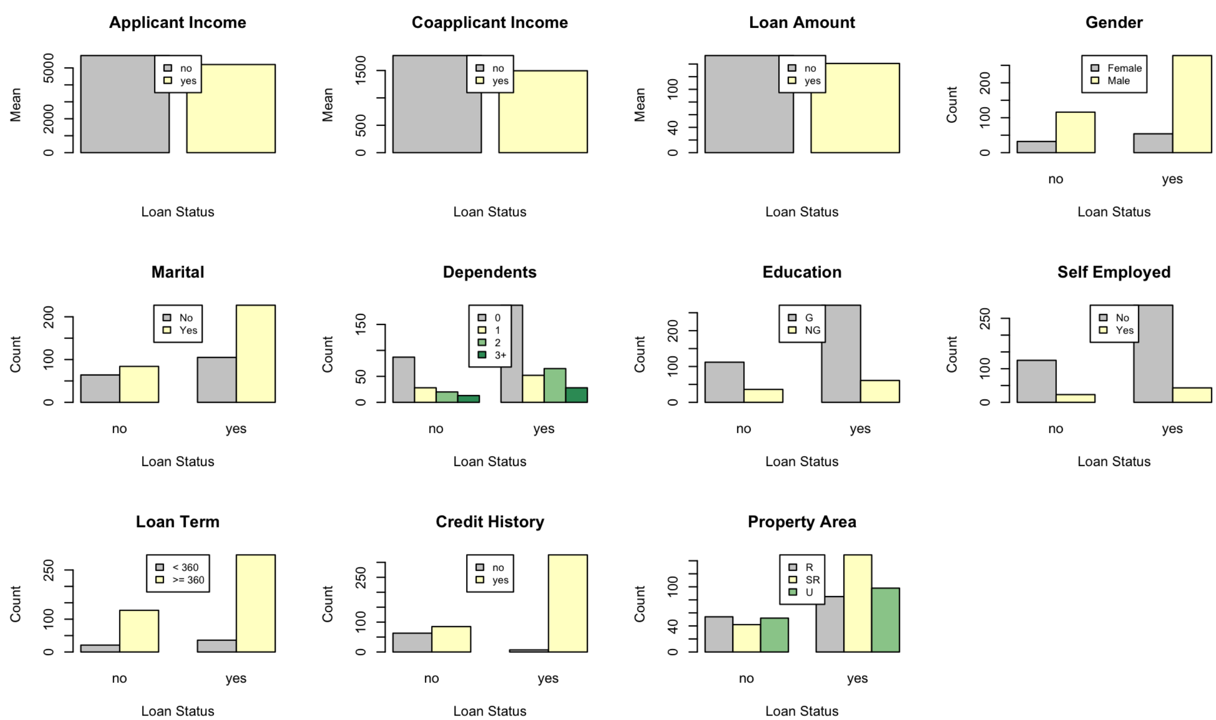
```
## # A tibble: 2 x 4
##   Loan_Status mean_income mean_coincome mean_loan
##   <fct>             <dbl>         <dbl>     <dbl>
## 1 no                5730.         1773.      153.
## 2 yes               5201.         1496.      141.
```

Appendix 5: Exploratory Data Analysis: Barplots

```
par(mfrow=c(3,4))
income.status <- mean_numerical[c("Loan_Status","mean_income")]
row.names(income.status) <- c("no", "yes")
barplot(income.status$mean_income,
        main = "Applicant Income",
        xlab = "Loan Status", ylab = "Mean",
        col = c("#CCCCCC", "#FFFFCC"),
        legend.text = rownames(income.status),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars

coincome.status <- mean_numerical[c("Loan_Status","mean_coincome")]
row.names(coincome.status) <- c("no", "yes")
barplot(coincome.status$mean_coincome,
        main = "Coapplicant Income",
        xlab = "Loan Status", ylab = "Mean",
        col = c("#CCCCCC", "#FFFFCC"),
        legend.text = rownames(coincome.status),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars

amount.status <- mean_numerical[c("Loan_Status","mean_loan")]
row.names(amount.status) <- c("no", "yes")
barplot(amount.status$mean_loan,
        main = "Loan Amount",
        xlab = "Loan Status", ylab = "Mean",
        col = c("#CCCCCC", "#FFFFCC"),
        legend.text = rownames(amount.status),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars

gender.status <- table(loan$Gender,loan$Loan_Status)
barplot(gender.status,
        main = "Gender",
        xlab = "Loan Status", ylab = "Count",
        col = c("#CCCCCC", "#FFFFCC"),
        legend.text = rownames(gender.status),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars

married.status <- table(loan$Married,loan$Loan_Status)
barplot(married.status,
        main = "Marital",
        xlab = "Loan Status", ylab = "Count",
```

```r
        col = c("#CCCCCC", "#FFFFCC"),
        legend.text = rownames(married.status),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars

dependents.status <- table(loan$Dependents,loan$Loan_Status)
barplot(dependents.status,
        main = "Dependents",
        xlab = "Loan Status", ylab = "Count",
        col = c("#CCCCCC","#FFFFCC", "#99CC99", "#339966"),
        legend.text = c("0", "1", "2", "3+"),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars

Education.status <- table(loan$Education,loan$Loan_Status)
barplot(Education.status,
        main = "Education",
        xlab = "Loan Status", ylab = "Count",
        col = c("#CCCCCC", "#FFFFCC"),
        legend.text = c("G", "NG"),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars

self_employed.status <- table(loan$Self_Employed,loan$Loan_Status)
barplot(self_employed.status,
        main = "Self Employed",
        xlab = "Loan Status", ylab = "Count",
        col = c("#CCCCCC", "#FFFFCC"),
        legend.text = rownames(self_employed.status),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars
term.status <- table(loan$Loan_Amount_Term,loan$Loan_Status)
barplot(term.status,
        main = "Loan Term",
        xlab = "Loan Status", ylab = "Count",
        col = c("#CCCCCC", "#FFFFCC"),
        legend.text = rownames(term.status),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars

credit.status <- table(loan$Credit_History,loan$Loan_Status)
barplot(credit.status,
        main = "Credit History",
```

```
        xlab = "Loan Status", ylab = "Count",
        col = c("#CCCCCC", "#FFFFCC"),
        legend.text = rownames(credit.status),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars

property.status <- table(loan$Property_Area,loan$Loan_Status)
barplot(property.status,
        main = "Property Area",
        xlab = "Loan Status", ylab = "Count",
        col = c("#CCCCCC", "#FFFFCC", "#99CC99"),
        legend.text = c("R", "SR", "U"),
        args.legend = list(x = "top",
                           inset = c(- 0.15, 0),
                           cex = 0.75),
        beside = TRUE) # Grouped bars
```



Appendix 6: Classification tree

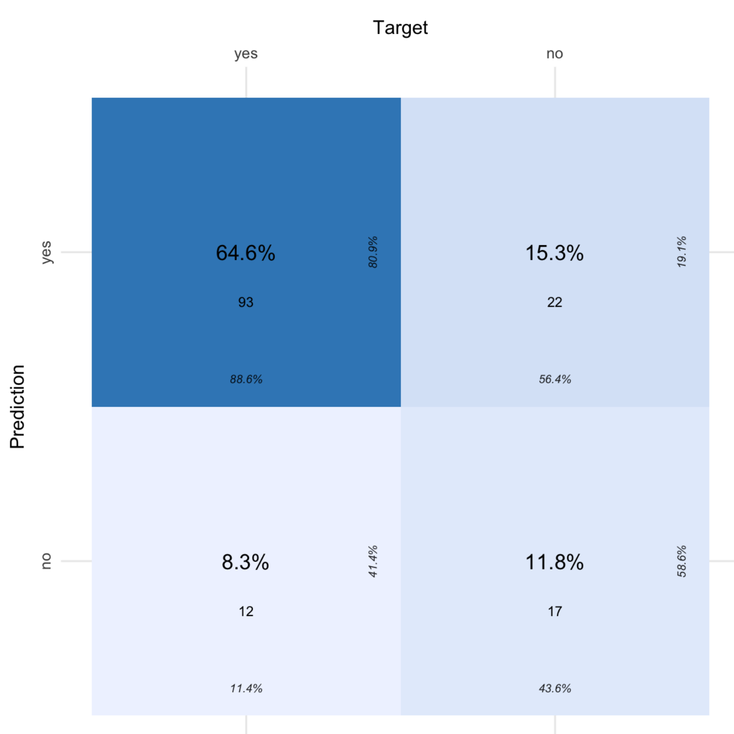
```
#Fit a classification tree to the training data and use both "gini" and "deviance" as the splitting cri
library(tree)
set.seed(1)
train <- sample(1:nrow(loan), 0.7*nrow(loan))
loan.train <- loan[train,]
loan.test <- loan[-train,]
y <- loan["Loan_Status"]
y.test <- y[-train,  ]
```

6.1 Using "deviance"

```
set.seed(1)
tree.loan.d <- tree(Loan_Status ~., loan, subset=train, split="deviance", minsize = 25)
tree.pred.d <- predict(tree.loan.d, loan.test, type="class")
table(tree.pred.d, y.test)
```

```
##            y.test
## tree.pred.d no yes
##         no  17  12
##         yes 22  93
```

```
library(cvms)
library(tibble)   # tibble()
cfm.t.d <- as_tibble(table(tree.pred.d, y.test))
plot_confusion_matrix(cfm.t.d ,
                      target_col = "y.test",
                      prediction_col = "tree.pred.d",
                      counts_col = "n")
```



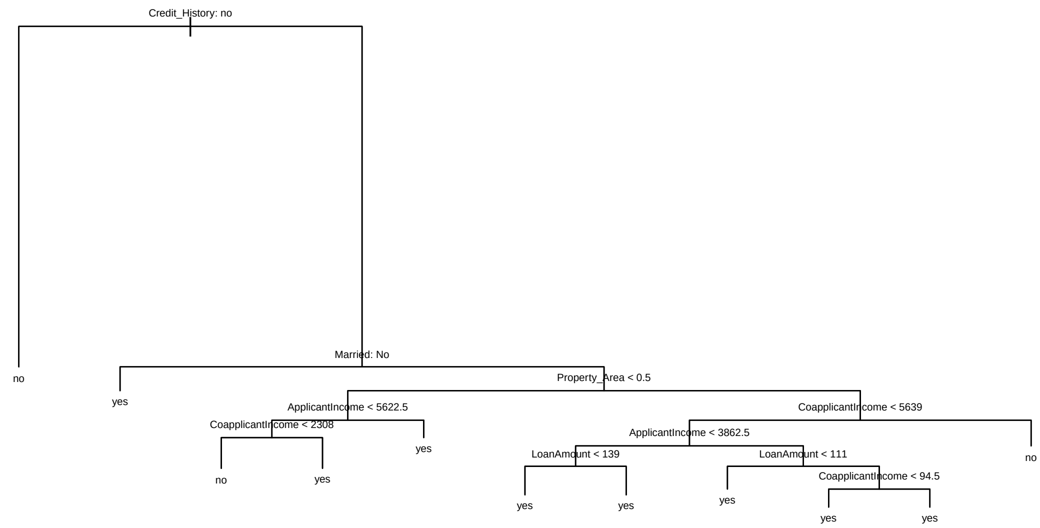
Compute the prediction accuracy of the test

```
round(mean(tree.pred.d == y.test)*100,2) # Prediction accuracy on test
```

```
## [1] 76.39
```

Print the tree

```
plot(tree.loan.d)
text(tree.loan.d, pretty = 0, cex = .5)
```

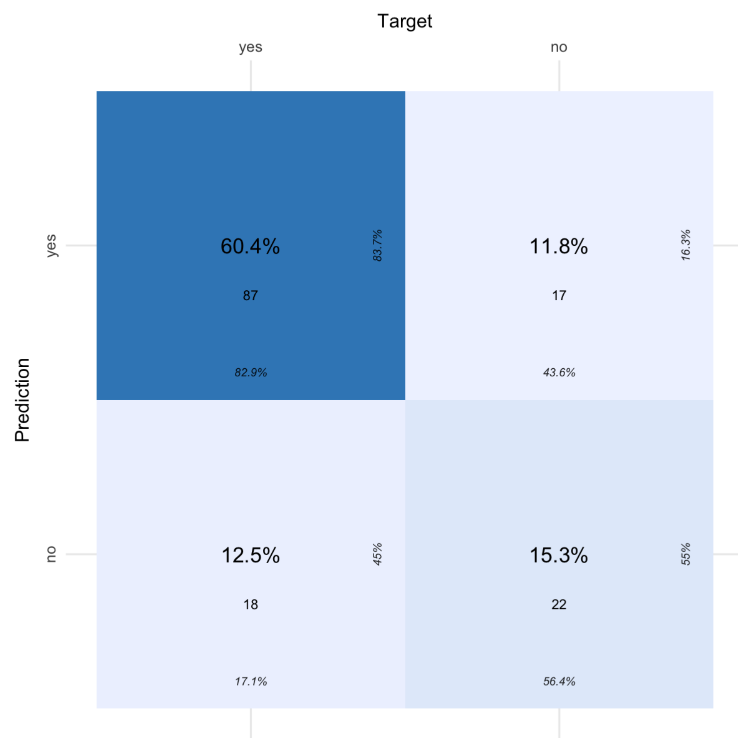


6.2 Using "gini"

```
set.seed(1)
tree.loan.g <- tree(Loan_Status ~., loan, subset=train, split="gini",minsize = 25)
tree.pred.g <- predict(tree.loan.g, loan.test, type="class")
table(tree.pred.g, y.test)
```

```
##            y.test
## tree.pred.g no  yes
##         no   22  18
##         yes 17  87
```

```
cfm.t.g <- as_tibble(table(tree.pred.g, y.test))
plot_confusion_matrix(cfm.t.g ,
                      target_col = "y.test",
                      prediction_col = "tree.pred.g",
                      counts_col = "n")
```

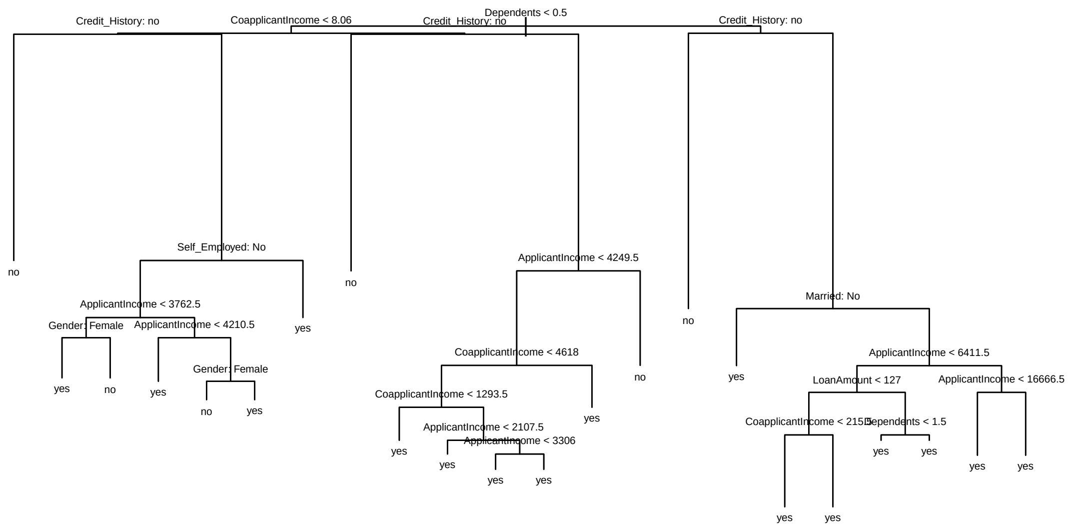


Compute the prediction accuracy of the test

```
round(mean(tree.pred.g == y.test)*100,2) # Prediction accuracy on test
```

```
## [1] 75.69
```

```
plot(tree.loan.g)
text(tree.loan.g, pretty = 0, cex = .5)
```


Dependents < 0.5
Credit_History: no
CoapplicantIncome < 8.06
Credit_History: no
Credit_History: no
no
Self_Employed: No
no
ApplicantIncome < 4249.5
no
Married: No
ApplicantIncome < 3762.5
Gender: Female
ApplicantIncome < 4210.5
yes
CoapplicantIncome < 4618
no
yes
ApplicantIncome < 6411.5
LoanAmount < 127
ApplicantIncome < 16666.5
yes
no
yes
Gender: Female
no
yes
CoapplicantIncome < 1293.5
yes
ApplicantIncome < 2107.5
ApplicantIncome < 3306
yes
yes
yes
yes
CoapplicantIncome < 2155
Dependents < 1.5
yes
yes
yes
yes
yes
yes

Appendix 7: Random Forest Model

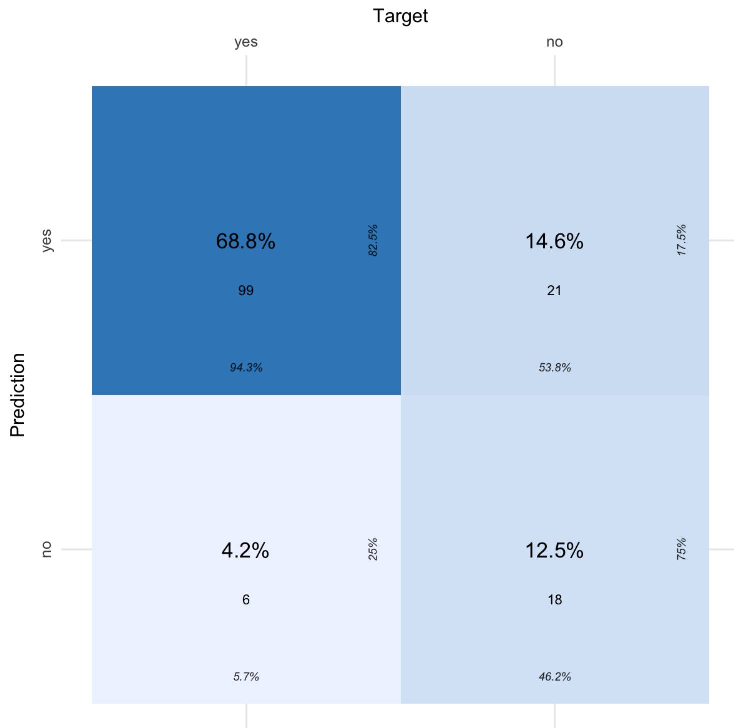
```
library(randomForest)
set.seed(1)
rf.loan <- randomForest(Loan_Status~., data = loan, subset = train, importance = TRUE)
importance(rf.loan)
```

```
##                            no       yes MeanDecreaseAccuracy MeanDecreaseGini
## Gender             -3.5723335  6.154066            3.3305819         3.184605
## Married            -0.7186122  5.861364            4.6894489         4.195608
## Dependents         -4.3609759  3.518970            0.6407502         6.134959
## Education           0.6527760  2.752323            2.6749970         3.500709
## Self_Employed      -2.3331215  2.128590            0.5835789         1.923666
## ApplicantIncome     2.3237100 11.758452           11.9474324        27.965931
## CoapplicantIncome   1.0943497  5.429672            5.2572201        15.346591
## LoanAmount         -0.9460537  7.603894            6.0374585        24.943772
## Loan_Amount_Term   -3.2876631  2.272634            0.0310081         2.181102
## Credit_History     51.2569021 51.760339           59.5643399        36.459870
## Property_Area       2.0777785  1.350466            2.2828471         7.291546
```

```
set.seed(1)
rf.pred <- predict(rf.loan, loan.test,type="class")
table(rf.pred, y.test)
```

```
##        y.test
## rf.pred no yes
##     no  18   6
##     yes 21  99
```

```
cfm.rf <- as_tibble(table(rf.pred, y.test))
plot_confusion_matrix(cfm.rf ,
                      target_col = "y.test",
                      prediction_col = "rf.pred",
                      counts_col = "n")
```



Compute the prediction accuracy of the test

```r
round(mean(rf.pred == y.test)*100,2) # Prediction accuracy on test
```

```
## [1] 81.25
```

print the variable importance graph

```r
varImpPlot(rf.loan)
```

rf.loan


Credit_History
ApplicantIncome
LoanAmount
CoapplicantIncome
Married
Gender
Education
Property_Area
Dependents
Self_Employed
Loan_Amount_Term
0
10
20
30
40
50
60
MeanDecreaseAccuracy
Credit_History
ApplicantIncome
LoanAmount
CoapplicantIncome
Property_Area
Dependents
Married
Education
Gender
Loan_Amount_Term
Self_Employed
0
10
20
30
MeanDecreaseGini

Reference

An Exploratory Data Analysis for Loan Prediction Based on Nature of the Clients. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7 Issue-4S, November 2018. X.Francis Jency, V.P.Sumathi, Janani Shiva Sri.

GOVARDHAN C. 2019. Home Loan Predictions, Version 1. Retrieved March 13, 2022 from https://www.kaggle.com/gavincanacam/home-loan-predictions.

R Core Team. 2020. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/ (https://www.R-project.org/).