HW3

Vinayak Patel

10/24/2021

Objective

In the project, I will explore the dataset for loan approval. I will create various models to predict the loan approvals. In the end, I will test the performance of each model based on the accuracy of the prediction

Data Exploration

Load the required libraries

Load Data

```
# Load Data
Loan_approval =  read.csv("C:/Users/patel/Downloads/Loan_approval.csv", header=T, na.strings=c("","NA"))
```

The loan approval status data dictionary is as below

VARIABLE	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)

Data Summary

```r
#dim
dim(Loan_approval)
```

```
## [1] 614  13
```

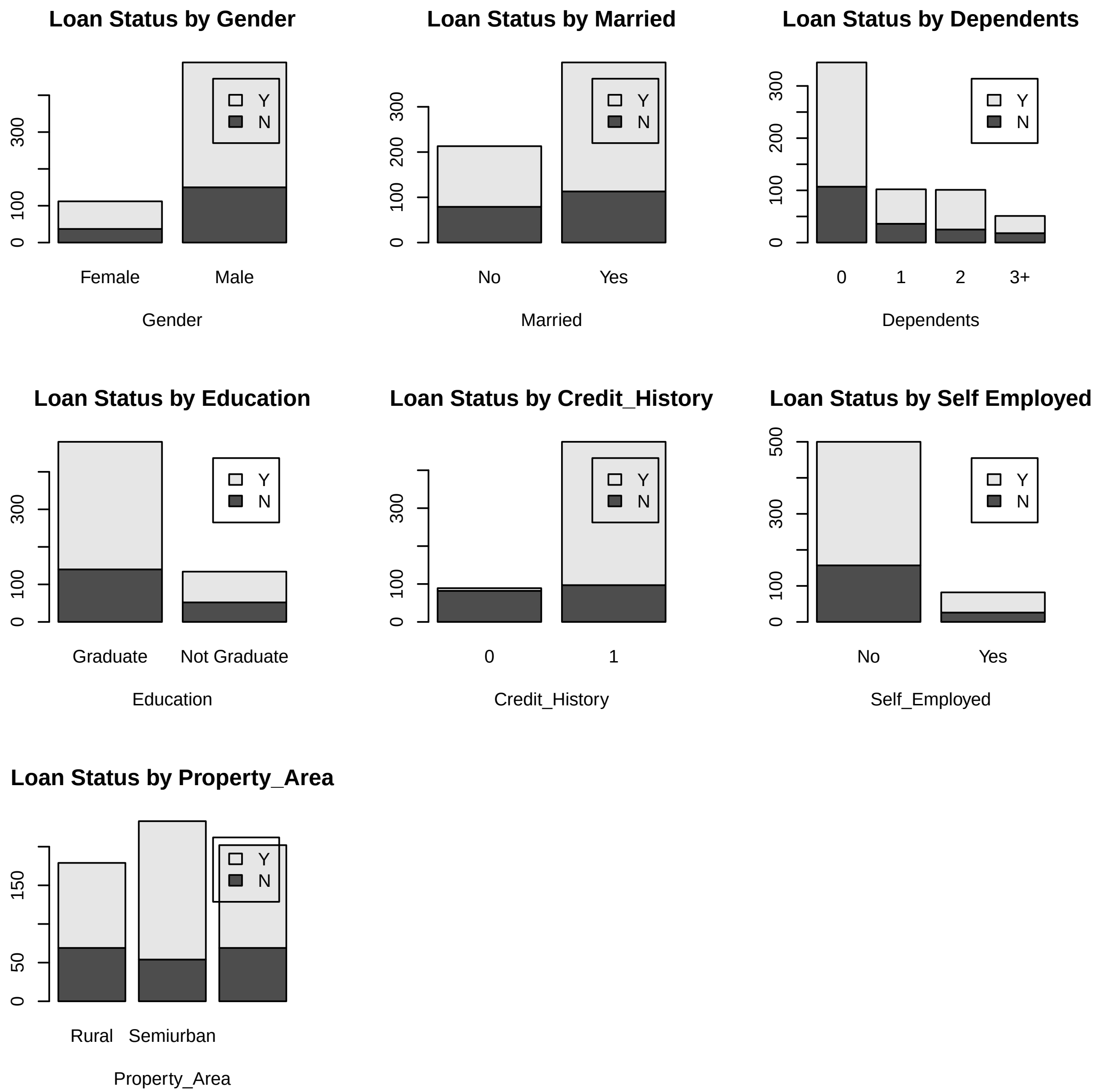
There are 614 observations of 13 variables.

Frequency Distributions

This function lets us compare the distribution of a target variable vs another variable. The variables can be categorical or continuous.

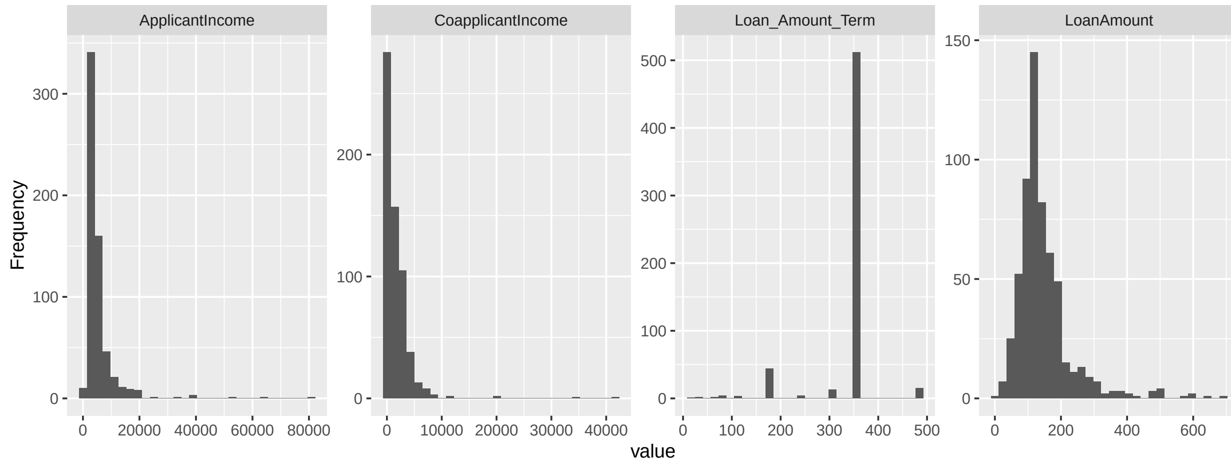
For categorical features

```r
##To visualize distributions for all categorical features:
par(mfrow=c(3,3))

barplot(table(Loan_approval$Loan_Status, Loan_approval$Gender), main="Loan Status by Gender",
        xlab="Gender", legend = TRUE)

barplot(table(Loan_approval$Loan_Status, Loan_approval$Married), main="Loan Status by Married",
        xlab="Married", legend = TRUE)

barplot(table(Loan_approval$Loan_Status, Loan_approval$Dependents), main="Loan Status by Dependents",
        xlab="Dependents", legend = TRUE)

barplot(table(Loan_approval$Loan_Status, Loan_approval$Education), main="Loan Status by Education",
        xlab="Education", legend = TRUE)

barplot(table(Loan_approval$Loan_Status, Loan_approval$Credit_History), main="Loan Status by Credit_His
        xlab="Credit_History", legend = TRUE)

barplot(table(Loan_approval$Loan_Status, Loan_approval$Self_Employed), main="Loan Status by Self Employe
        xlab="Self_Employed", legend = TRUE)

barplot(table(Loan_approval$Loan_Status, Loan_approval$Property_Area)
, main="Loan Status by Property_Area",
        xlab="Property_Area", legend = TRUE)
```



continuous features

```r
#To visualize distributions for all continuous features:
plot_histogram(Loan_approval)
```



Data Cleaning

```
#remove loan_id
Loan_approval <- subset(Loan_approval, select = -Loan_ID )

##mutate as factors for categorical data

Loan_approval <- Loan_approval %>%
           mutate(Gender = factor(Gender),
                  Married = factor(Married),
                  Dependents = factor(Dependents),
                  Education = factor(Education),
                  Self_Employed = factor(Self_Employed),
                  Property_Area = factor(Property_Area),
                  Loan_Status = factor(Loan_Status),
                  Credit_History= factor(Credit_History))

summary(Loan_approval)
```

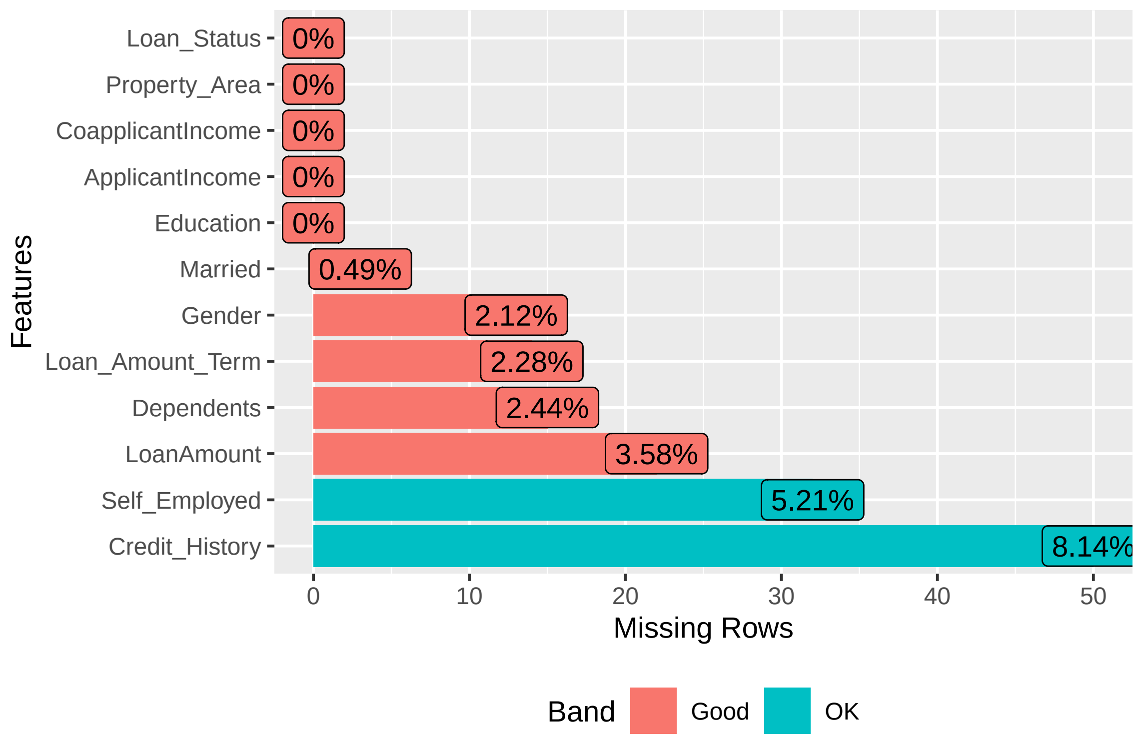
```
##     Gender     Married     Dependents         Education     Self_Employed
##  Female:112   No  :213   0    :345   Graduate     :480   No  :500
##  Male  :489   Yes :398   1    :102   Not Graduate:134   Yes : 82
##  NA's  : 13   NA's:  3   2    :101                      NA's: 32
##                          3+   : 51
##                          NA's: 15
##
##
##  ApplicantIncome CoapplicantIncome   LoanAmount     Loan_Amount_Term
##  Min.   :  150   Min.   :    0     Min.   :  9.0   Min.   : 12
##  1st Qu.: 2878   1st Qu.:    0     1st Qu.:100.0   1st Qu.:360
##  Median : 3812   Median : 1188     Median :128.0   Median :360
##  Mean   : 5403   Mean   : 1621     Mean   :146.4   Mean   :342
##  3rd Qu.: 5795   3rd Qu.: 2297     3rd Qu.:168.0   3rd Qu.:360
##  Max.   :81000   Max.   :41667     Max.   :700.0   Max.   :480
##                                    NA's   :22      NA's   :14
```

```
##   Credit_History   Property_Area Loan_Status
##   0    : 89       Rural     :179   N:192
##   1    :475       Semiurban:233   Y:422
##   NA's: 50        Urban     :202
##
##
##
##
```

I subset the load_id from the dataset and convert categorical data as factor.

Missing values table

```
#Checking the Missing data proportion
plot_missing(Loan_approval)
```



Handling Missing Values

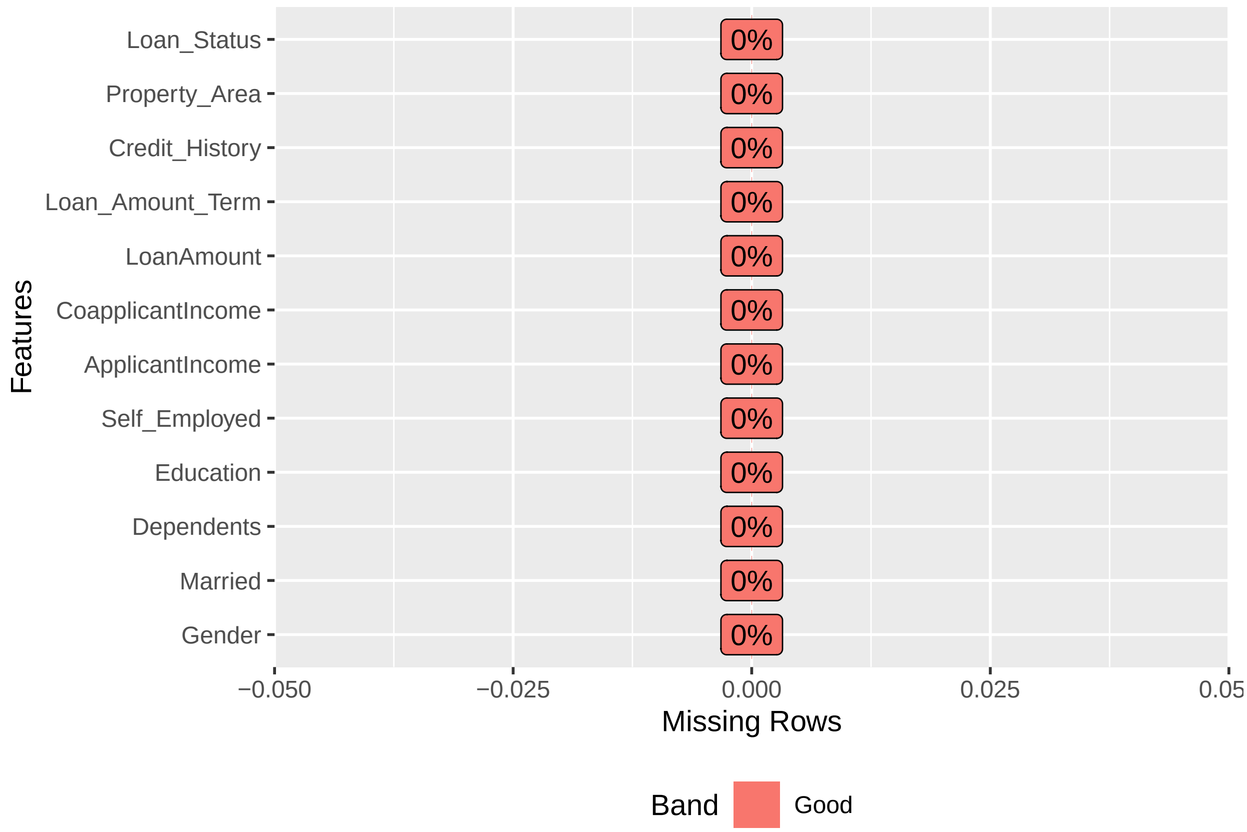
From the missing value chart, I concluded that there isn't any variance with missing values being more than 10 percent of the data. The dataset is almost complete just a few observations with missing values that can be omitted or impute. I will consider imputing the missing value with the missForest library.

```
LA_df<- missForest(Loan_approval)
```

```
##   missForest iteration 1 in progress...done!
```

```
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
##   missForest iteration 4 in progress...done!
```

```
Loan_approval_clean <- LA_df$ximp

plot_missing(Loan_approval_clean)
```



Splitting the data 70-30

```
set.seed(17)
# splitting the data into 70-30

df1_split=split_train_test(Loan_approval_clean,outcome=Loan_Status,0.7)

#display train
kable(head(df1_split$train,5))
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmou
1	Male	No	0	Graduate	No	5849	0	148.87
2	Male	Yes	1	Graduate	No	4583	1508	128.00
3	Male	Yes	0	Graduate	Yes	3000	0	66.00
5	Male	No	0	Graduate	No	6000	0	141.00
6	Male	Yes	2	Graduate	Yes	5417	4196	267.00

Linear Discriminant Analysis

Selection of the variable

I drop the categorical variables like Gender, Married, Dependents, Education, Self_Employed, Credit__History, Property__Area since Linear Discriminant Analysis (LDA) needs continuous variables to feed into the model.

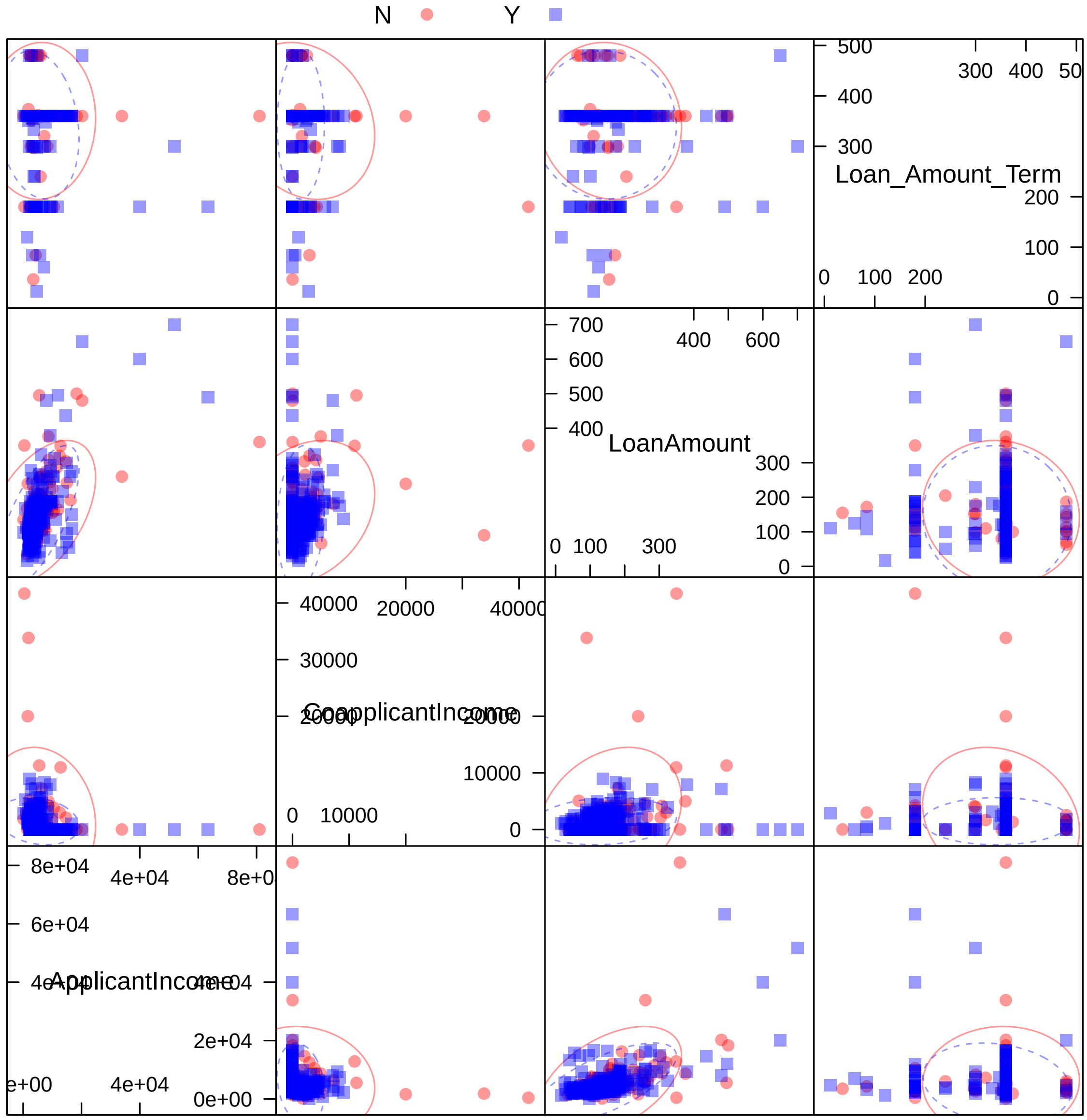
```
# remove categorical values
La_categ<- subset(df1_split$train, select = -c(Gender,Married,Dependents,Education,Self_Employed,Credit_
```

linearly separable or nor?

I will feature plot to see is there any linearly separable or nor?

```
library(AppliedPredictiveModeling)
transparentTheme(trans = .4)

featurePlot(x = La_categ[,1:4],
            y = La_categ$Loan_Status,
            plot = "ellipse",
            ## Add a key at the top
            auto.key = list(columns = 3))
```

```
## Warning in draw.key(simpleKey(...), draw = FALSE): not enough rows for columns
```



Scatter Plot Matrix

The plot suggests that it is not linearly separable. The different colors of eclipses in the scatter plot represent the loan approval status. Overlapping of eclipse suggests that it is not linearly separable. So Linear Discriminant Analysis Model would not be ideal for this dataset. However, I can still create an LDA model to verify how it performs with other models.

Build LDA Model

```
lda_rt_s<-Sys.time()
model_lda<- lda(Loan_Status ~. , data = La_categ)
lda_rt_e<-Sys.time()
lda_rt<- lda_rt_e-lda_rt_s
```

```r
model_lda
```

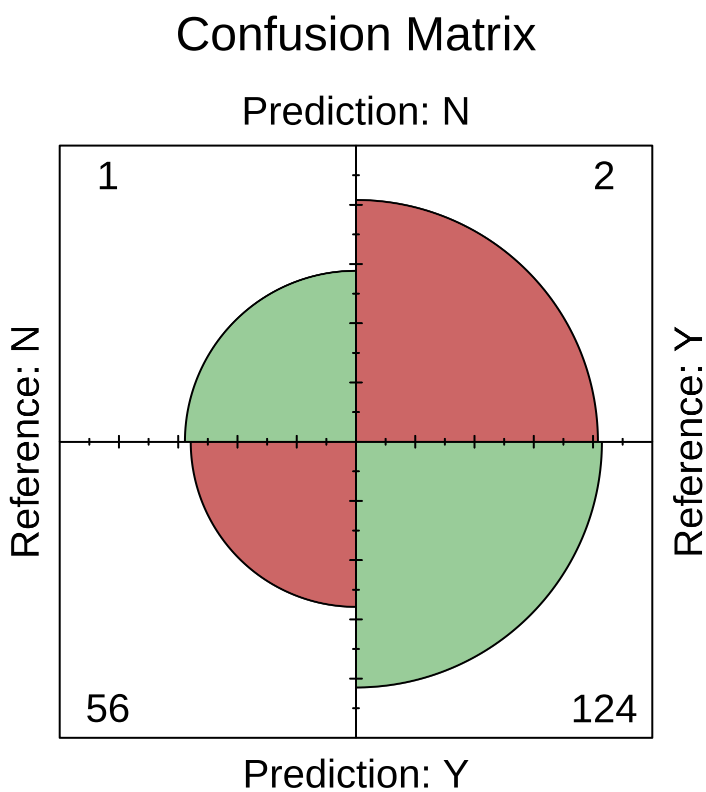
```
## Call:
## lda(Loan_Status ~ ., data = La_categ)
##
## Prior probabilities of groups:
##         N         Y
## 0.3132251 0.6867749
##
## Group means:
##   ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term
## N        5806.459          2079.622   156.3186         350.4028
## Y        5175.659          1467.662   142.7721         342.3622
##
## Coefficients of linear discriminants:
##                             LD1
## ApplicantIncome   -5.466203e-05
## CoapplicantIncome -2.298479e-04
## LoanAmount        -2.784649e-03
## Loan_Amount_Term  -9.309354e-03
```

Prior probabilities of groups: the proportion of training observations in each group. For example, there are 69% of the training observations is loan Approved

Group means: group center of gravity. Shows the mean of each variable in each group.

Coefficients of linear discriminant: Shows the linear combination of predictor variables that are used to form the LDA decision rule

```r
La_categ_test<- subset(df1_split$test, select = -c(Gender,Married,Dependents,Education,Self_Employed,Cr


predict_lda_test <- predict(model_lda, La_categ_test)

cm_lda <- confusionMatrix( predict_lda_test$class, La_categ_test$Loan_Status)


#confusionMatrix
fourfoldplot(cm_lda$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1, main = "Confusion Matrix")
```



K-nearest neighbor (KNN) algorithm

Preparation

Preprocessing is all about correcting the problems in data before building a machine learning model using that data. Problems can be of many types like missing values, attributes with a different range, etc.

```
prepro <- preProcess(x = df1_split$train, method = c("center", "scale"))
prepro
```

```
## Created from 431 samples and 12 variables
##
## Pre-processing:
##   - centered (4)
##   - ignored (8)
##   - scaled (4)
```

TrainControl() method. It controls the computational nuances of the train() method. I will use method `"repeatedcv"` for cross-validation

```
trControl <- trainControl(method="repeatedcv",number = 10, repeats = 5)
start_time<-Sys.time()
model_knn <- train(Loan_Status ~ ., data = df1_split$train,
```

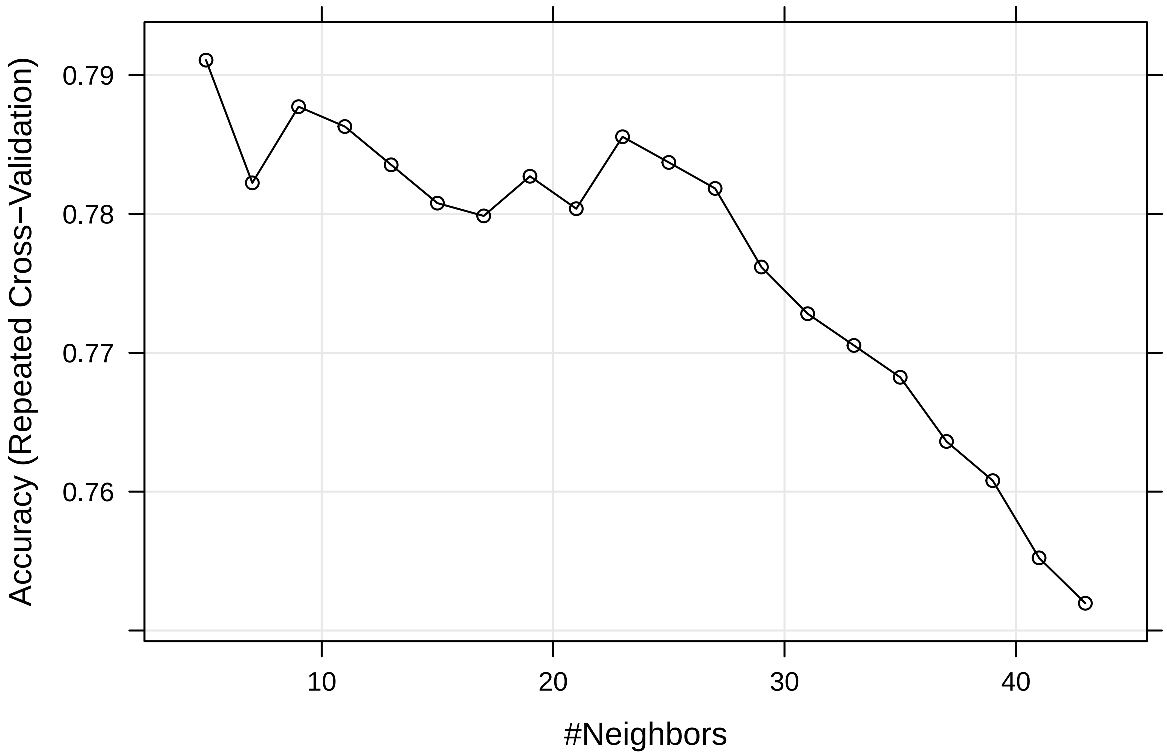
```
                 method = "knn",
                 trControl = trControl,
                 preProcess = c("center","scale"),
                 tuneLength = 20)
model_knn
```

```
## k-Nearest Neighbors
##
## 431 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: 388, 387, 388, 388, 388, 388, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.7910737  0.4350433
##    7  0.7822345  0.3983626
##    9  0.7877202  0.4081612
##   11  0.7862917  0.3998529
##   13  0.7835337  0.3886200
##   15  0.7807747  0.3769046
##   17  0.7798540  0.3751641
##   19  0.7827107  0.3826019
##   21  0.7803730  0.3751288
##   23  0.7855547  0.3924389
##   25  0.7837058  0.3856369
##   27  0.7818348  0.3795482
##   29  0.7761754  0.3604183
##   31  0.7728108  0.3484925
##   33  0.7705275  0.3415704
##   35  0.7682342  0.3321826
##   37  0.7636142  0.3146655
##   39  0.7607903  0.3029999
##   41  0.7552295  0.2811076
##   43  0.7519621  0.2672359
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```
end_time<-Sys.time()
knn_rt<- end_time-start_time
```

Accuracy was used to select the optimal model using the largest value. The final value used for the model was k = 5.

```
plot(model_knn)
```



Predict from knn model

```
predict_knn_test <- predict(model_knn,newdata = df1_split$test)
mean(predict_knn_test == df1_split$test$Loan_Status) # accuracy
```

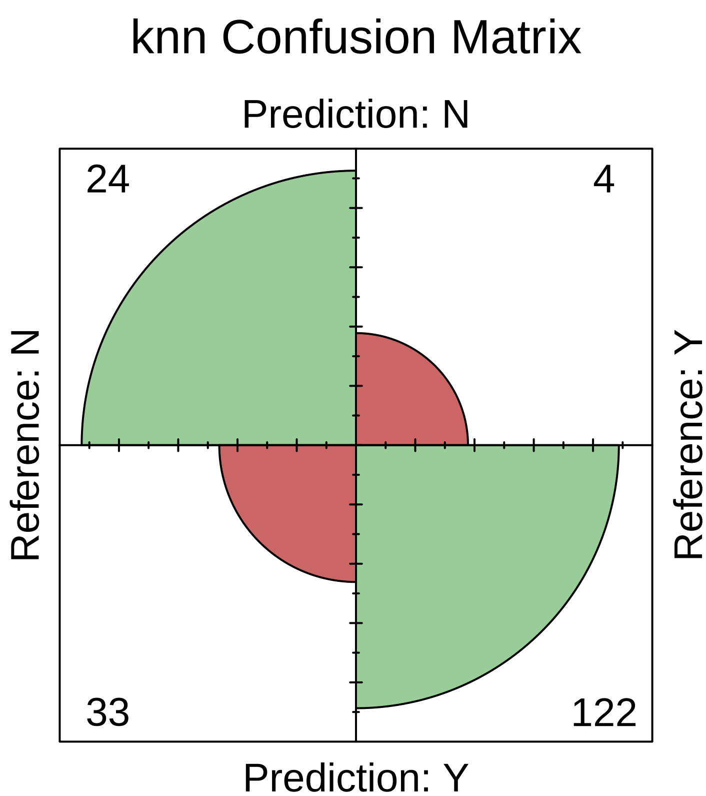
```
## [1] 0.7978142
```

```
cm_knn <- confusionMatrix(predict_knn_test, df1_split$test$Loan_Status)
cm_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  24   4
##          Y  33 122
##
##                Accuracy : 0.7978
##                  95% CI : (0.7323, 0.8535)
##     No Information Rate : 0.6885
##     P-Value [Acc > NIR] : 0.0006328
##
##                   Kappa : 0.4523
```

```
##
## Mcnemar's Test P-Value : 4.161e-06
##
##             Sensitivity : 0.4211
##             Specificity : 0.9683
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.7871
##              Prevalence : 0.3115
##          Detection Rate : 0.1311
##    Detection Prevalence : 0.1530
##       Balanced Accuracy : 0.6947
##
##        'Positive' Class : N
##
```

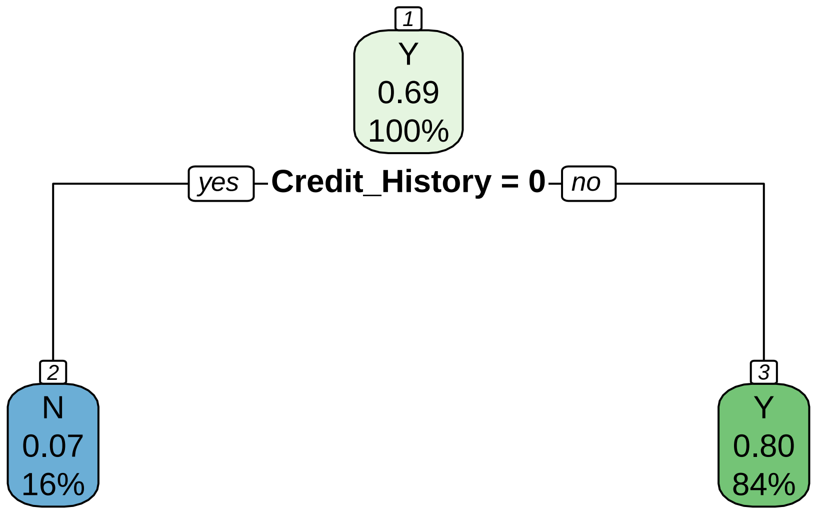
```
fourfoldplot(cm_knn$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1, main = "knn Confusion Matrix")
```



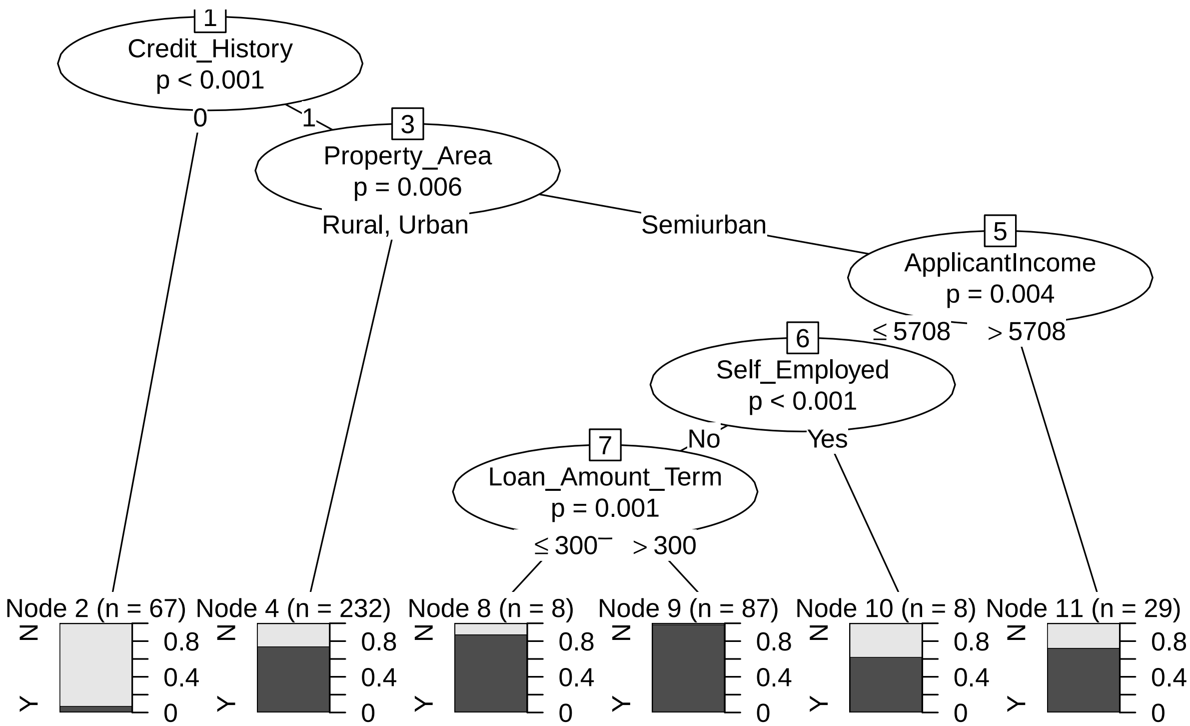
Decision Tree model

```
start_time<-Sys.time()
model_dt <- rpart(Loan_Status~ ., data=df1_split$train)
```

```
end_time<-Sys.time()
dt_rt<- end_time-start_time
rpart.plot(model_dt, nn=TRUE)
```




```
ctree_ <- ctree(Loan_Status~ ., data=df1_split$train)
plot(ctree_)
```

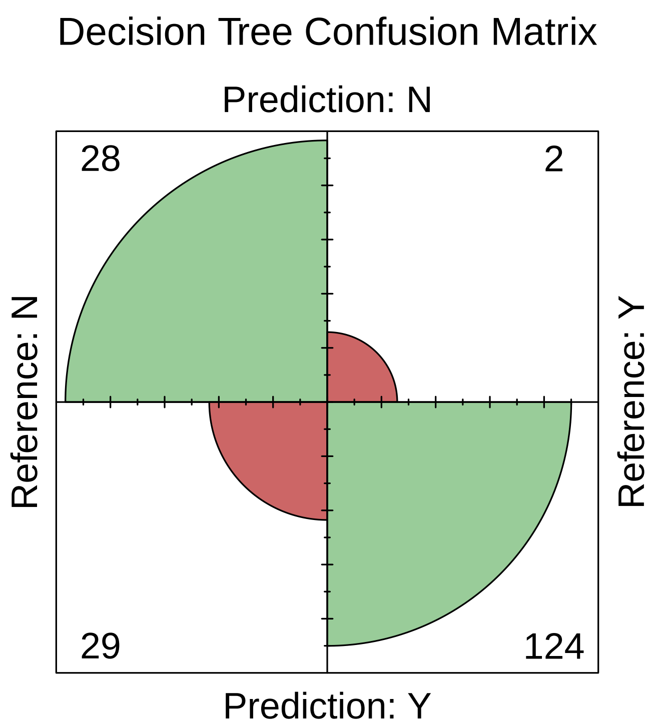



```
summary(model_dt)
```

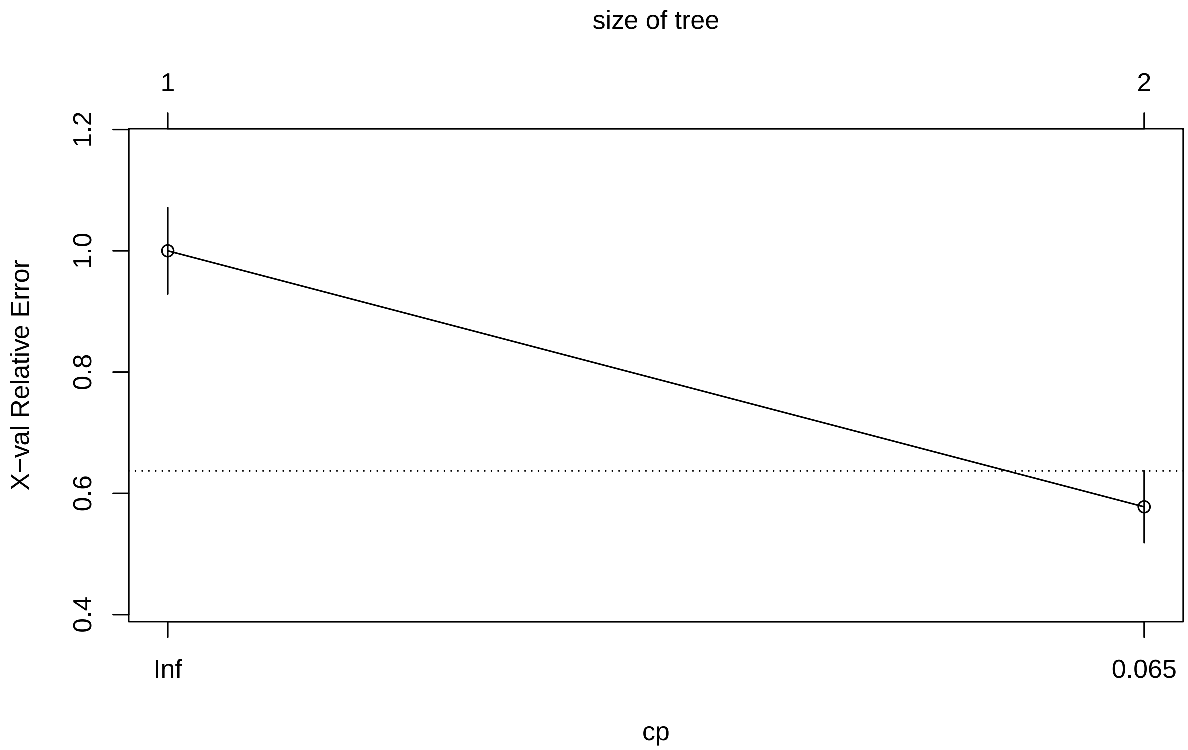
```
## Call:
## rpart(formula = Loan_Status ~ ., data = df1_split$train)
##   n= 431
##
##          CP nsplit rel error    xerror       xstd
## 1 0.4222222      0 1.0000000 1.0000000 0.07132476
## 2 0.0100000      1 0.5777778 0.5777778 0.05920553
##
## Variable importance
## Credit_History
##            100
##
## Node number 1: 431 observations,    complexity param=0.4222222
##   predicted class=Y  expected loss=0.3132251  P(node) =1
##     class counts:   135   296
##    probabilities: 0.313 0.687
##   left son=2 (67 obs) right son=3 (364 obs)
##   Primary splits:
##       Credit_History    splits as  LR,           improve=59.455720, (0 missing)
##       Property_Area     splits as  LRL,          improve= 6.303598, (0 missing)
##       Loan_Amount_Term  < 360.8041 to the right, improve= 4.654997, (0 missing)
##       LoanAmount        < 200.5    to the right, improve= 2.683292, (0 missing)
##       CoapplicantIncome < 8219.5   to the right, improve= 2.289073, (0 missing)
##
## Node number 2: 67 observations
##   predicted class=N  expected loss=0.07462687  P(node) =0.1554524
```

```
##     class counts:    62     5
##    probabilities: 0.925 0.075
##
## Node number 3: 364 observations
##   predicted class=Y  expected loss=0.2005495  P(node) =0.8445476
##     class counts:    73   291
##    probabilities: 0.201 0.799
```

```
dtControl= rpart.control(minsplit = 20, xval = 81, cp=0.01)
predict_dt_test <- predict(model_dt, df1_split$test,
                  type = "class",
                  control=dtControl)

cm_dt<- confusionMatrix(predict_dt_test, df1_split$test$Loan_Status)

fourfoldplot(cm_dt$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1, main = "Decision Tree Confusion Matrix")
```




```
plotcp(model_dt)
```



Random Forest Model

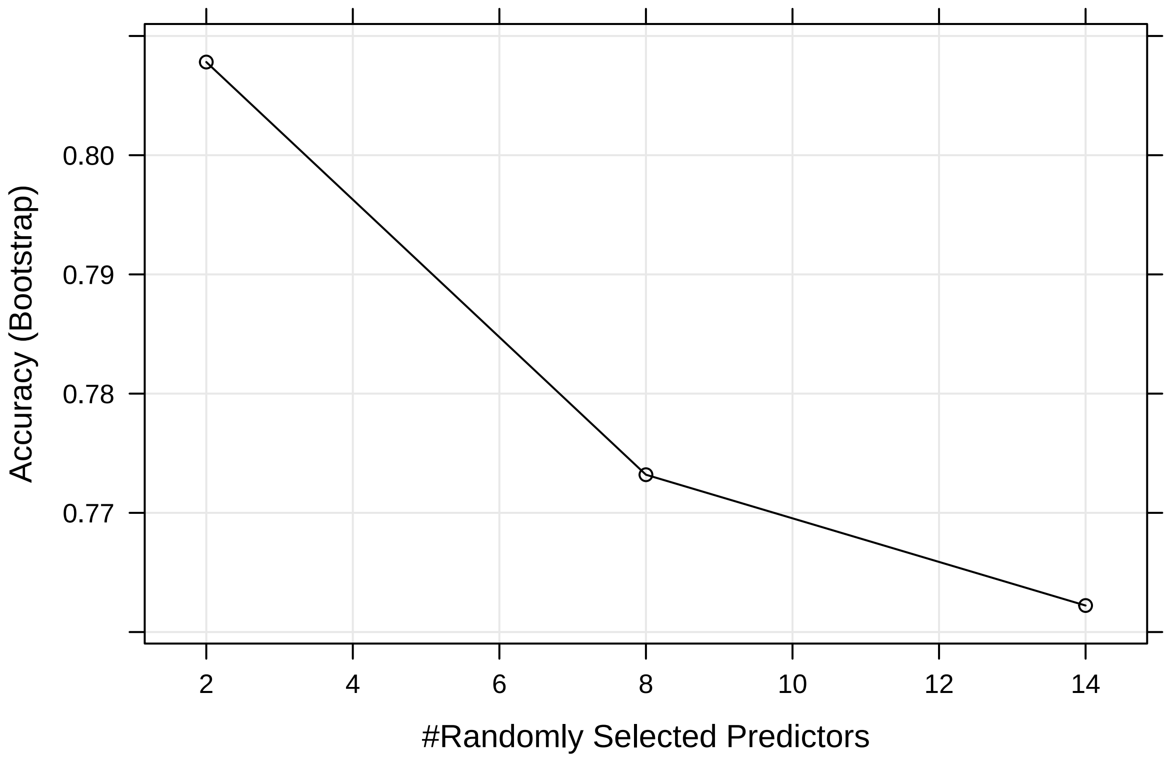
Build Model

```
Rfcontrol <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
start_time<-Sys.time()
model_rf <- train(Loan_Status~., data = df1_split$train, method="rf")
end_time<-Sys.time()

rf_rt<- end_time-start_time

print(model_rf)
```

```
## Random Forest
##
## 431 samples
##  11 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 431, 431, 431, 431, 431, 431, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
```

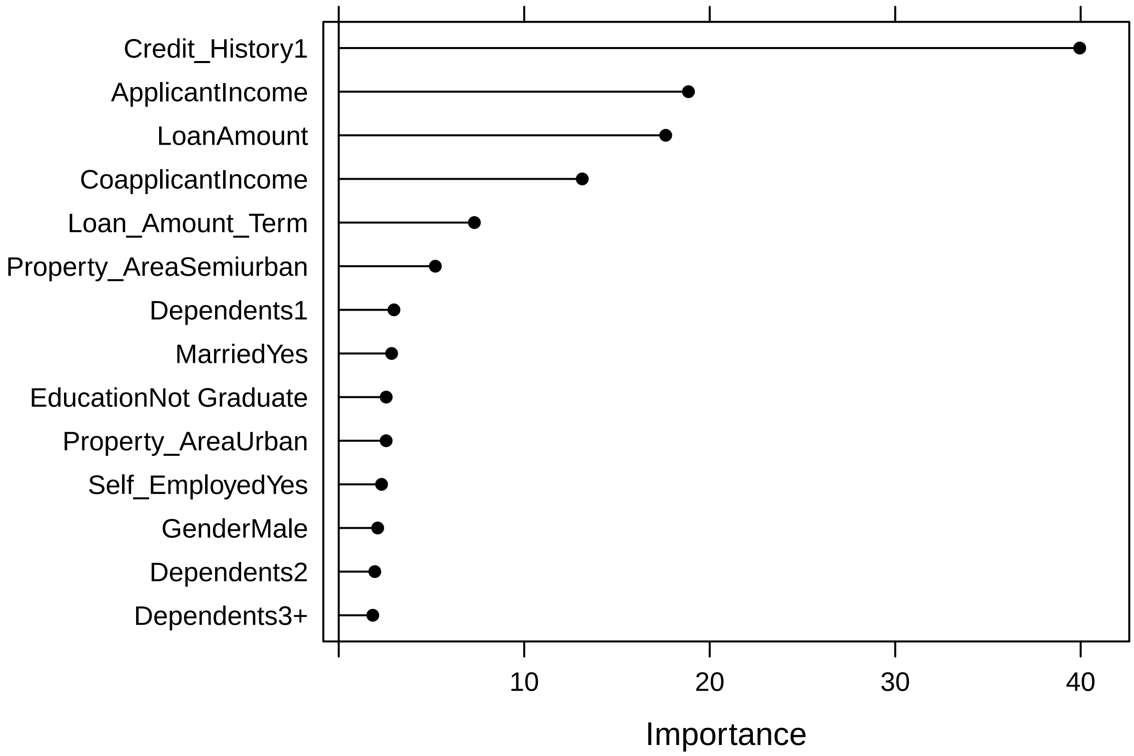
```
##    2    0.8078148  0.4893961
##    8    0.7731983  0.4328786
##   14    0.7622259  0.4084476
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
plot(model_rf)
```



Importance variable

```r
rfImp <- varImp(model_rf, scale = FALSE)
plot(rfImp)
```



Top 5 Importance variable are `Credit_History1`, `ApplicantIncome` `LoanAmount`, `CoapplicantIncome` and `Loan_Amount_Term`.

```
# prediction from random forest model
predict_rf_test <- predict(model_rf, df1_split$test,type='raw')
mean(predict_rf_test == df1_split$test$Loan_Status) # accuracy
```

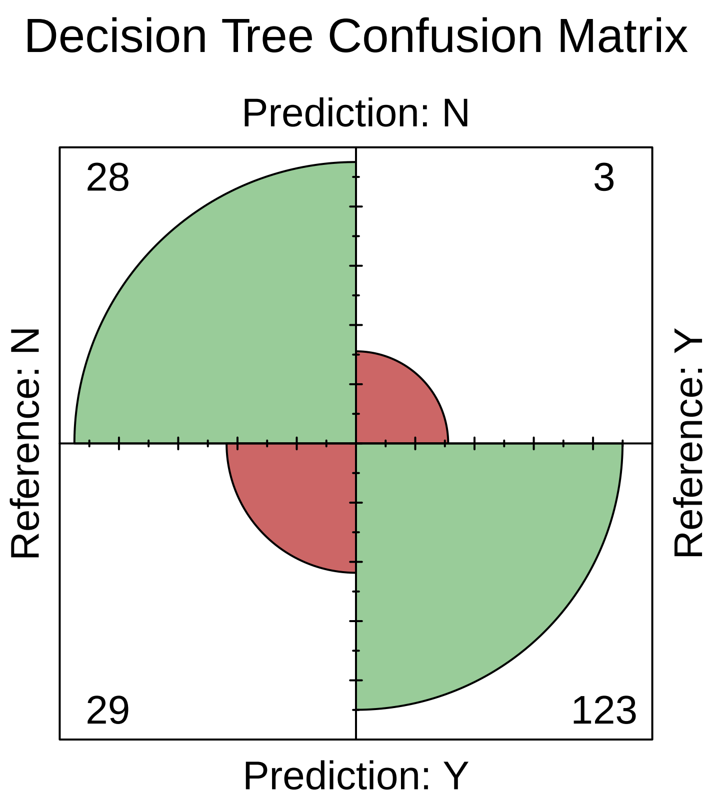
```
## [1] 0.8251366
```

```
cm_rf <- confusionMatrix(predict_rf_test, df1_split$test$Loan_Status)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  28   3
##          Y  29 123
##
##                Accuracy : 0.8251
##                  95% CI : (0.7622, 0.8772)
##     No Information Rate : 0.6885
##     P-Value [Acc > NIR] : 2.012e-05
##
##                   Kappa : 0.5341
##
```

```
##   Mcnemar's Test P-Value : 9.897e-06
##
##              Sensitivity : 0.4912
##              Specificity : 0.9762
##           Pos Pred Value : 0.9032
##           Neg Pred Value : 0.8092
##               Prevalence : 0.3115
##           Detection Rate : 0.1530
##     Detection Prevalence : 0.1694
##        Balanced Accuracy : 0.7337
##
##         'Positive' Class : N
##
```

```
fourfoldplot(cm_rf$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1, main = "Decision Tree Confusion Matrix")
```



Model Performance

```
results<-as.data.frame(round(cm_lda$overall,4))
names(results)[1] <-"lda"
results$knn <- round(cm_knn$overall, 4)
results$decisiontree <- round(cm_dt$overall, 4)
```

```
results$randomforest <- round(cm_rf$overall, 4)


runtime<-rbind(c(lda_rt, knn_rt, dt_rt, rf_rt))
results<-data.frame(rbind(as.matrix(results), as.matrix(runtime)))
row.names(results)[8] <- "Runtime"

kable(results)
```

	lda	knn	decisiontree	randomforest
Accuracy	0.6831000	0.79780	0.830600	0.82510
Kappa	0.0023000	0.45230	0.546200	0.53410
AccuracyLower	0.6103000	0.73230	0.768300	0.76220
AccuracyUpper	0.7497000	0.85350	0.881900	0.87720
AccuracyNull	0.6885000	0.68850	0.688500	0.68850
AccuracyPValue	0.5982000	0.00060	0.000000	0.00000
McnemarPValue	0.0000000	0.00000	0.000000	0.00000
Runtime	0.0218899	22.56568	0.051867	56.98691

As `results` suggest that decision tree and random forest perform better than LDA and knn. both model Accuracy as 0.8306 and 0.8251 respectively The best performance or the model I pick is the decision tree algorithm because `Accuracy` of the model is better but also it is significantly faster than random forest algorithm. The runtime of a decision tree is 0.051867 and on the other hand 56.9869082