Untitled

Housing Loan Prediction

Dream Housing Finance company deals in all home loans. Customer first apply for home loan after that company validates the customer eligibility for loan.

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. The prediction analysis identify the customers segments, those are eligible for loan amount. This will help the company to target these customers for home loans.

Library Used

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
## randomForest 4.6-12
```

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

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

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

```
##
## Attaching package: 'kernlab'
```

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

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
## The following object is masked from 'package:e1071':
##
##     impute

## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units

## Loading required package: statmod

##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai
##
## ----------------------------------------------------------------------

##
## Attaching package: 'h2o'

## The following objects are masked from 'package:stats':
##
##     cor, sd, var

## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

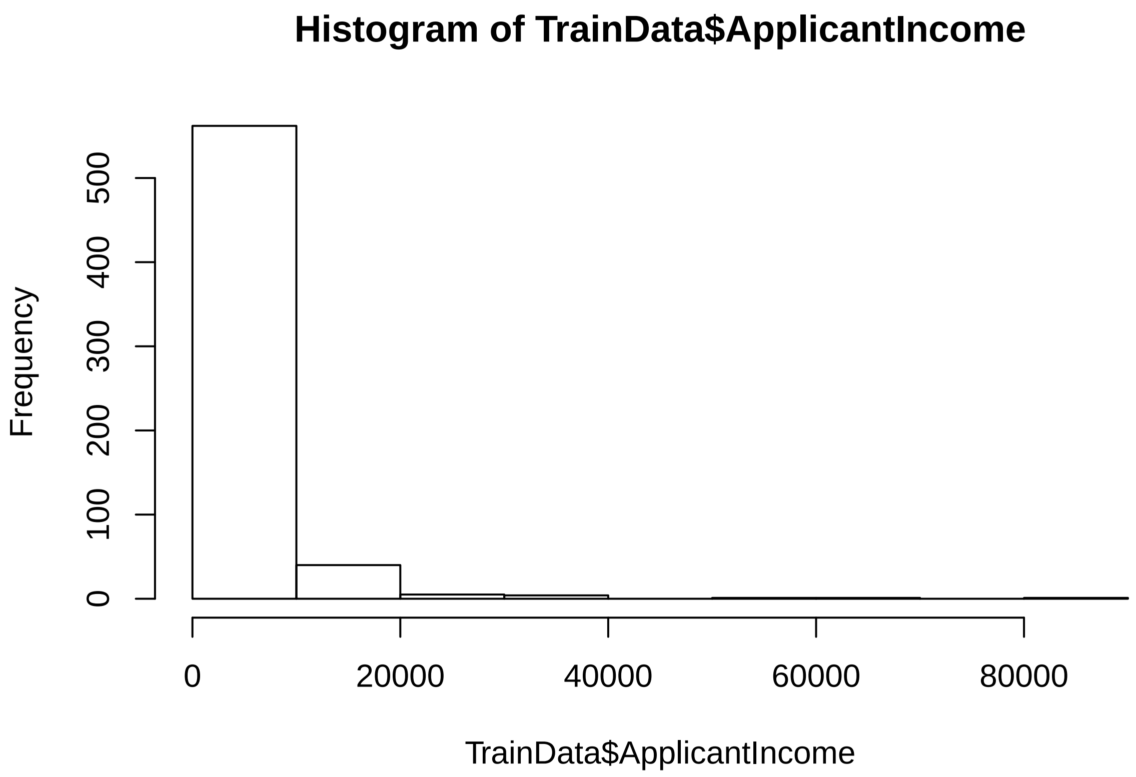
Load Data

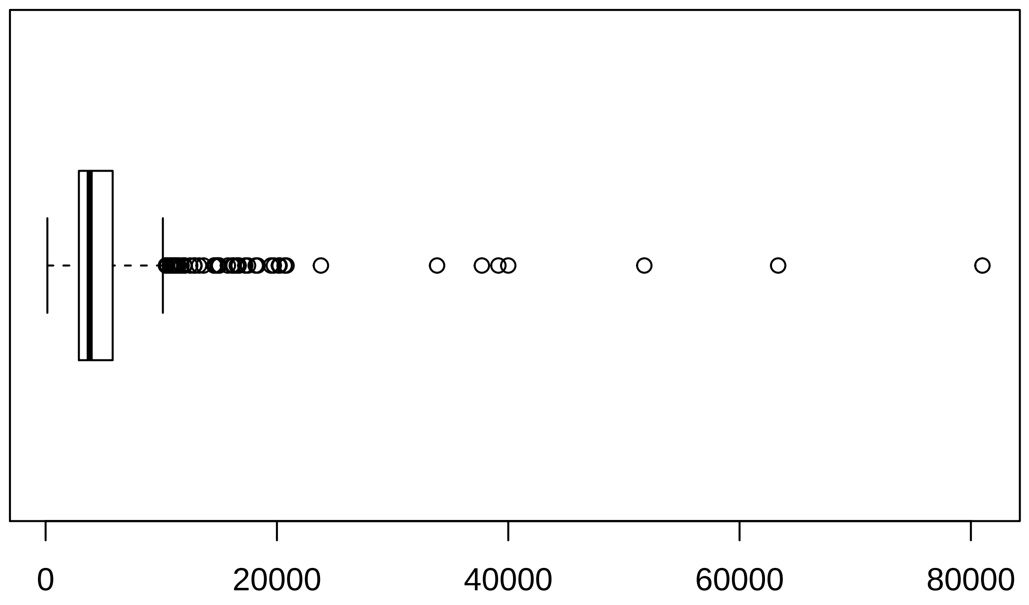
Initial Investigation

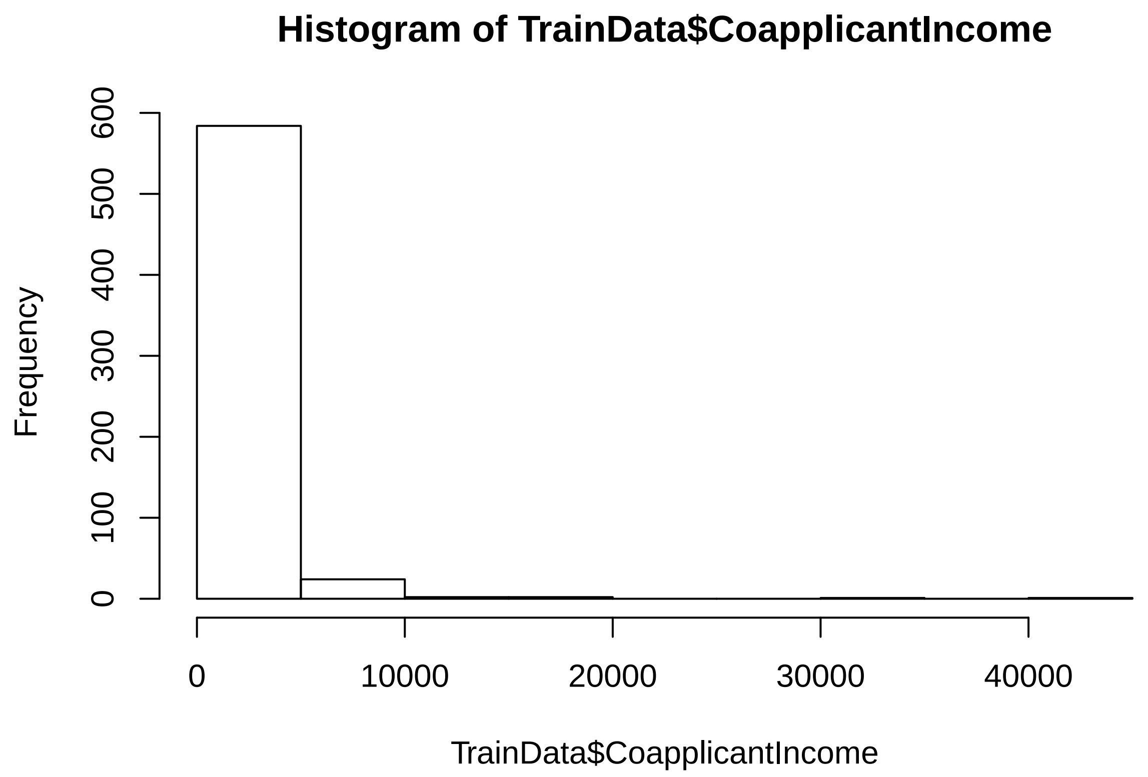
The initial investigation into the TrainData set involves plotting of histogram and boxplots for - Applicant Income - Coapplicant Income _ Loan Amount - Loan Amount Term

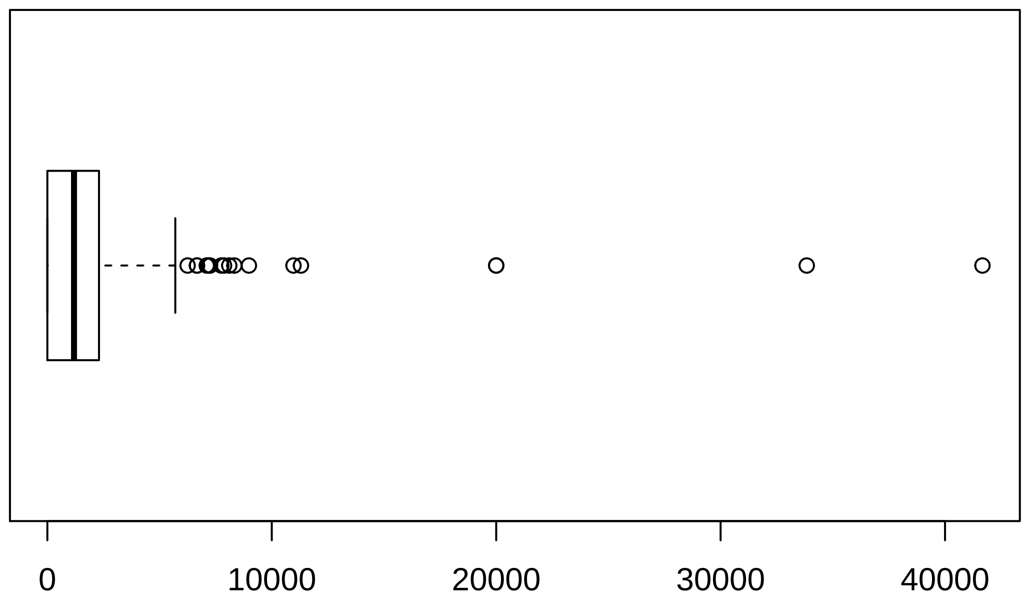
```
##      Loan_ID       Gender    Married    Dependents        Education
##  LP001002:  1   Female:112   No  :213   0   :345   Graduate    :480
##  LP001003:  1   Male  :489   Yes :398   1   :102   Not Graduate:134
##  LP001005:  1   NA's  : 13   NA's:  3   2   :101
##  LP001006:  1                           3+  : 51
##  LP001008:  1                           NA's: 15
##  LP001011:  1
##  (Other) :608
##  Self_Employed ApplicantIncome CoapplicantIncome   LoanAmount
##  No  :500      Min.   :  150   Min.   :    0     Min.   :  9.0
##  Yes : 82      1st Qu.: 2878   1st Qu.:    0     1st Qu.:100.0
##  NA's: 32      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                                  
```

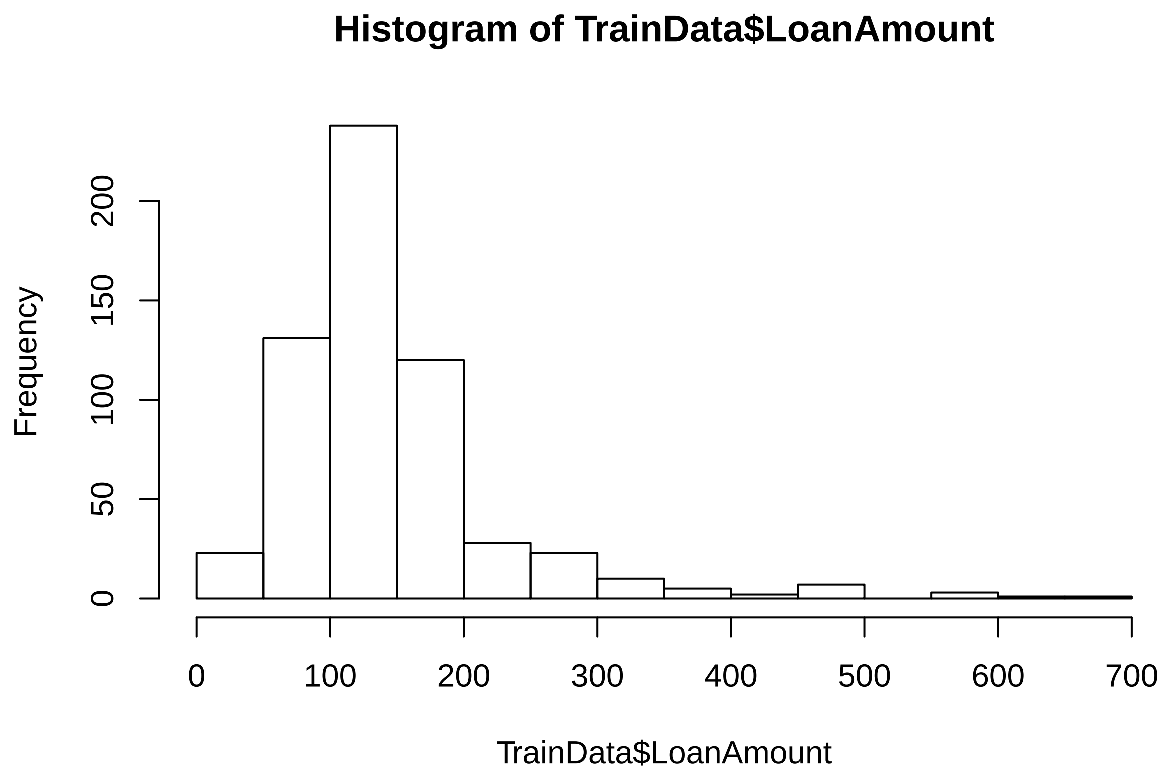
```
##      Loan_ID       Gender    Married   Dependents        Education  
##  LP001015:  1   Female: 70   No :134   0   :200   Graduate    :283  
##  LP001022:  1   Male  :286   Yes:233   1   : 58   Not Graduate: 84  
##  LP001031:  1   NA's  : 11             2   : 59                     
##  LP001035:  1                          3+  : 40                     
##  LP001051:  1                          NA's: 10                     
##  LP001054:  1                                                       
##  (Other) :361                                                       
##  Self_Employed ApplicantIncome CoapplicantIncome   LoanAmount   
##  No  :307      Min.   :    0   Min.   :    0     Min.   : 28.0  
##  Yes : 37      1st Qu.: 2864   1st Qu.:    0     1st Qu.:100.2  
##  NA's: 23      Median : 3786   Median : 1025     Median :125.0  
##                Mean   : 4806   Mean   : 1570     Mean   :136.1  
##                3rd Qu.: 5060   3rd Qu.: 2430     3rd Qu.:158.0  
##                Max.   :72529   Max.   :24000     Max.   :550.0  
##                                                  NA's   :5      
##  Loan_Amount_Term Credit_History     Property_Area
##  Min.   :  6.0    Min.   :0.0000   Rural    :111  
##  1st Qu.:360.0    1st Qu.:1.0000   Semiurban:116  
##  Median :360.0    Median :1.0000   Urban    :140  
##  Mean   :342.5    Mean   :0.8254                  
##  3rd Qu.:360.0    3rd Qu.:1.0000                  
##  Max.   :480.0    Max.   :1.0000                  
##  NA's   :6        NA's   :29                      
```

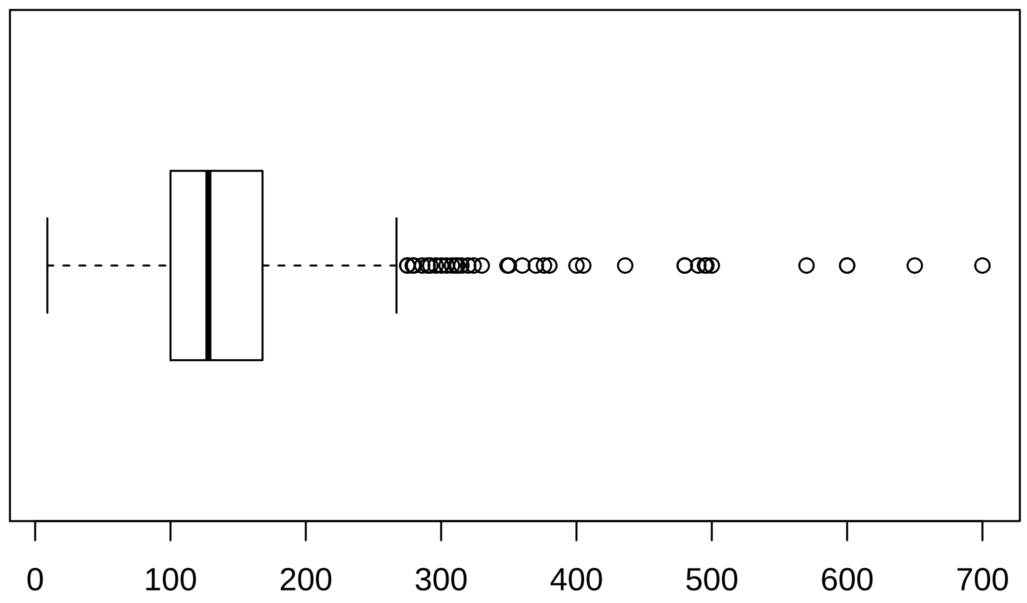

Histogram of TrainData$ApplicantIncome
Frequency
0
100
200
300
400
500
0
20000
40000
60000
80000
TrainData$ApplicantIncome

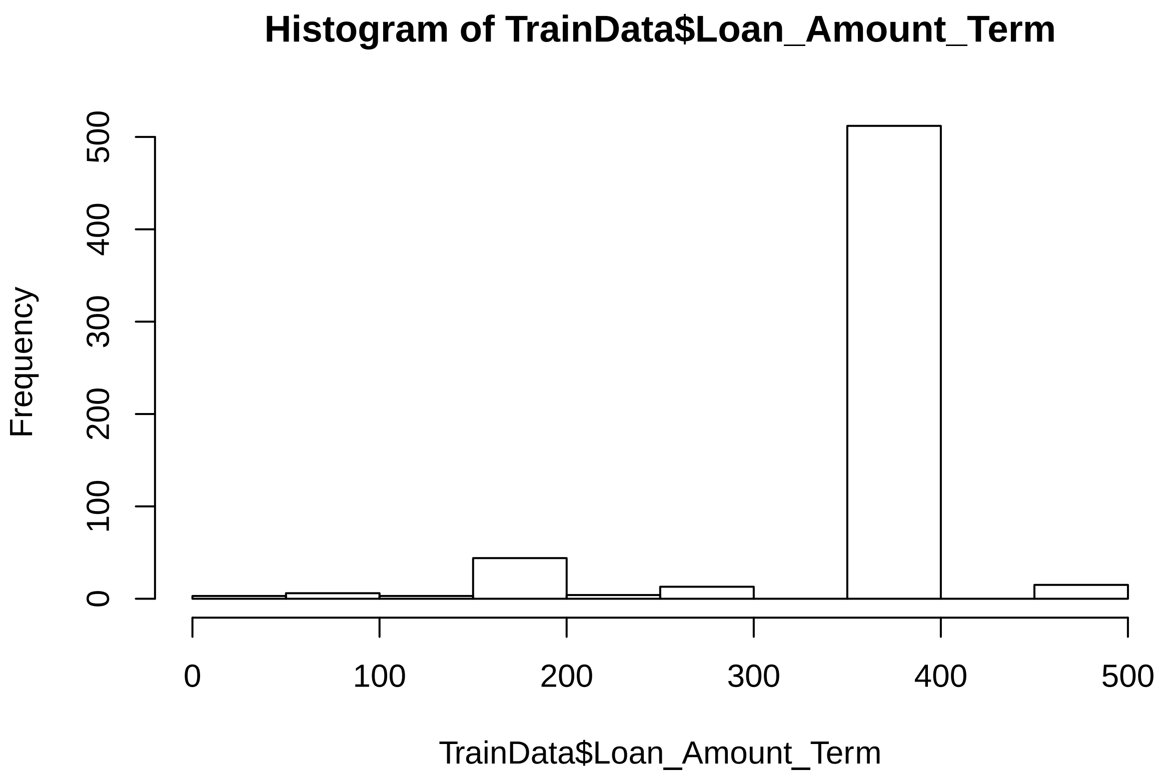

0
20000
40000
60000
80000

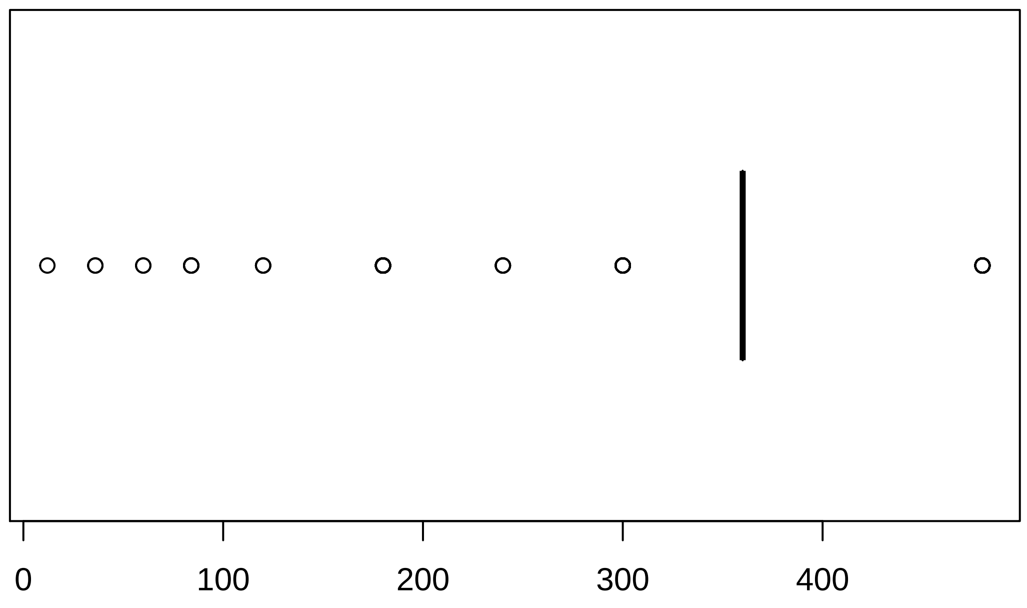

Histogram of TrainData$CoapplicantIncome
Frequency
0
100
200
300
400
500
600
0
10000
20000
30000
40000
TrainData$CoapplicantIncome


0
10000
20000
30000
40000


Histogram of TrainData$LoanAmount
Frequency
0
50
100
150
200
0
100
200
300
400
500
600
700
TrainData$LoanAmount


0
100
200
300
400
500
600
700


Histogram of TrainData$Loan_Amount_Term
Frequency
0
100
200
300
400
500
0
100
200
300
400
500
TrainData$Loan_Amount_Term



The first peek at the datasets provides following details:-

1. TrainData has 614 records with 13 variables.
2. TestData ahs 367 records with 13 variable.
3. The histogram and boxplot of ApplicantIncome shows their are outliers as there is a huge difference between the mean and the median.
4. The histogram and boxplot for CoapplicantIncome also shows that their are outliers because of the difference in mean and median.
5. The histogram and boxplot for LoanAmount also shows that their are outliers as the mean and median are different.
6. Similarly histogram and boxplot for Loan_Amount_Term also shows that their are outliers as the mean and median are different.

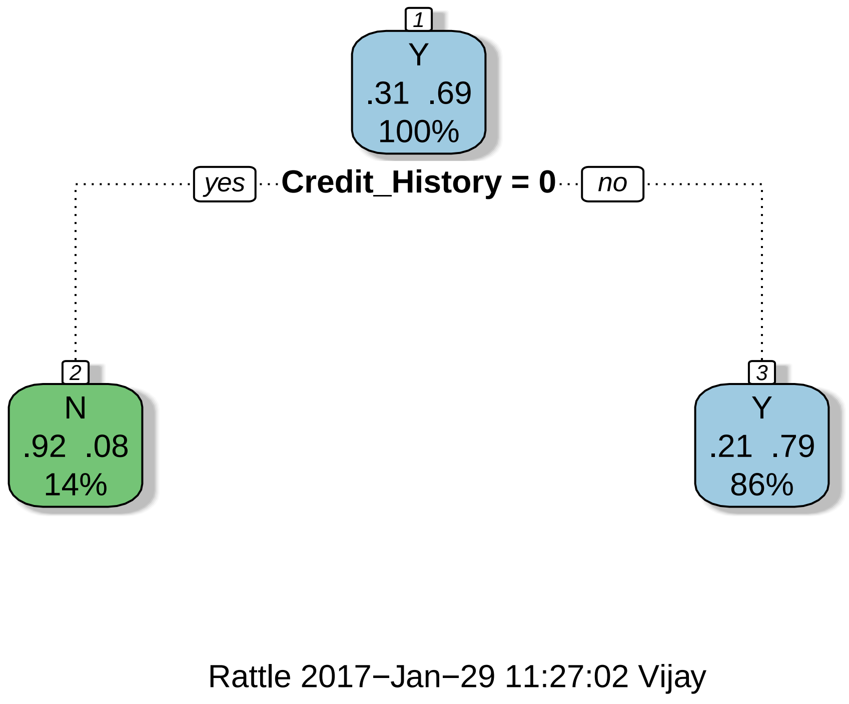
Transformation of TrainData Dataset

Transform the TestData dataset

Cleaning and Exploratory Data Analysis

Impute NAs in TestData

Build Models uing Machine Learning



```
##   N   Y
##  66 301
```

```
## Linear Discriminant Analysis
##
## 614 samples
##  12 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 489, 488, 489, 489, 488, 489, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8195286  0.5096318
```

```
##
##

## Quadratic Discriminant Analysis
##
## 614 samples
##  12 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 489, 489, 488, 488, 490, 489, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8194498  0.5098399
##
##

## CART
##
## 614 samples
##  12 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 489, 489, 488, 489, 489, 489, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00295858  0.7807407  0.4604251
##   0.00591716  0.7880808  0.4571333
##   0.42011834  0.7367340  0.1932099
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.00591716.

## k-Nearest Neighbors
##
## 614 samples
##  12 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 488, 488, 488, 489, 489, 489, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6593361  0.11104767
##   7  0.6483610  0.06520746
##   9  0.6723702  0.08959621
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was k = 9.
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 614 samples
##  12 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 488, 489, 488, 489, 489, 490, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.8028804  0.4552724
##   0.50  0.8211969  0.5151854
##   1.00  0.8211632  0.5184578
##
## Tuning parameter 'sigma' was held constant at a value of 0.9318165
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were sigma = 0.9318165 and C = 0.5.
```

```
## Random Forest
##
## 614 samples
##  12 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 488, 488, 488, 489, 490, 489, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8229490  0.5392694
##   3     0.8006918  0.4985474
##   4     0.7970555  0.4923830
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
##
## H2O is not running yet, starting it now...
```

```
## Warning in .h2o.startJar(nthreads = nthreads, max_memory = max_mem_size, : You have a 32-bit version
## Please download the latest Java SE JDK 7 from the following URL:
## http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
```

```
##
## Note:  In case of errors look at the following log files:
##     C:\Users\Vijay\AppData\Local\Temp\RtmpGMQJwN/h2o_Vijay_started_from_r.out
##     C:\Users\Vijay\AppData\Local\Temp\RtmpGMQJwN/h2o_Vijay_started_from_r.err
##
##
## Starting H2O JVM and connecting: .. Connection successful!
##
```

```
## R is connected to the H2O cluster:
##     H2O cluster uptime:         5 seconds 219 milliseconds
##     H2O cluster version:        3.10.0.8
##     H2O cluster version age:    3 months and 18 days !!!
##     H2O cluster name:           H2O_started_from_R_Vijay_fma179
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   0.96 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     R Version:                  R version 3.2.5 (2016-04-14)
```

```
## Warning in h2o.clusterInfo():
## Your H2O cluster version is too old (3 months and 18 days)!
## Please download and install the latest version from http://h2o.ai/download/
```

```
## function (ip = "localhost", port = 54321, startH2O = TRUE, forceDL = FALSE,
##     enable_assertions = TRUE, license = NULL, nthreads = -2,
##     max_mem_size = NULL, min_mem_size = NULL, ice_root = tempdir(),
##     strict_version_check = TRUE, proxy = NA_character_, https = FALSE,
##     insecure = FALSE, username = NA_character_, password = NA_character_,
##     cluster_id = NA_integer_, cookies = NA_character_)
## {
##     if (!is.character(ip) || length(ip) != 1L || is.na(ip) ||
##         !nzchar(ip))
##         stop("`ip` must be a non-empty character string")
##     if (!is.numeric(port) || length(port) != 1L || is.na(port) ||
##         port < 0 || port > 65536)
##         stop("`port` must be an integer ranging from 0 to 65536")
##     if (!is.logical(startH2O) || length(startH2O) != 1L || is.na(startH2O))
##         stop("`startH2O` must be TRUE or FALSE")
##     if (!is.logical(forceDL) || length(forceDL) != 1L || is.na(forceDL))
##         stop("`forceDL` must be TRUE or FALSE")
##     if (!is.numeric(nthreads) || length(nthreads) != 1L || is.na(nthreads) ||
##         nthreads < -2)
##         stop("`nthreads` must an integer value greater than or equal to -2")
##     if (!is.null(max_mem_size) && !(is.character(max_mem_size) &&
##         length(max_mem_size) == 1L && !is.na(max_mem_size) &&
##         nzchar(max_mem_size)))
##         stop("`max_mem_size` must be NULL or a non-empty character string")
##     if (!is.null(max_mem_size) && !regexpr("^[1-9][0-9]*[gGmM]$",
##         max_mem_size))
##         stop("`max_mem_size` option must be like 1g or 1024m")
##     if (!is.null(min_mem_size) && !(is.character(min_mem_size) &&
##         length(min_mem_size) == 1L && !is.na(min_mem_size) &&
##         nzchar(min_mem_size)))
##         stop("`min_mem_size` must be NULL or a non-empty character string")
##     if (!is.null(min_mem_size) && !regexpr("^[1-9][0-9]*[gGmM]$",
##         min_mem_size))
##         stop("`min_mem_size` option must be like 1g or 1024m")
##     if (!is.logical(enable_assertions) || length(enable_assertions) !=
##         1L || is.na(enable_assertions))
##         stop("`enable_assertions` must be TRUE or FALSE")
##     if (!is.null(license) && !is.character(license))
##         stop("`license` must be of class character")
##     if (!is.character(ice_root) || length(ice_root) != 1L ||
##         is.na(ice_root) || !nzchar(ice_root))
##         stop("`ice_root` must be a non-empty character string")
##     if (!is.logical(strict_version_check) || length(strict_version_check) !=
##         1L || is.na(strict_version_check))
##         stop("`strict_version_check` must be TRUE or FALSE")
##     if (!is.character(proxy) || !nzchar(proxy))
##         stop("`proxy` must be a character string or NA_character_")
##     if (!is.logical(https) || length(https) != 1L || is.na(https))
##         stop("`https` must be TRUE or FALSE")
##     if (!is.logical(insecure) || length(insecure) != 1L || is.na(insecure))
##         stop("`insecure` must be TRUE or FALSE")
##     if (https != insecure)
##         stop("`https` and `insecure` must both be TRUE to enable HTTPS")
##     if (!is.character(username) || !nzchar(username))
##         stop("`username` must be a character string or NA_character_")
##     if (!is.character(password) || !nzchar(password))
##         stop("`password` must be a character string or NA_character_")
##     if (is.na(username) != is.na(password))
##         stop("Must provide both `username` and `password`")
##     if (!is.na(cluster_id) && (!is.numeric(cluster_id) || cluster_id <
##         0))
##         stop("`cluster_id` must be an integer value greater than 0")
##     if (!is.na(cookies) && (!is.vector(cookies)))
##         stop("`cookies` must be a vector of cookie values")
##     if ((R.Version()$major == "3") && (R.Version()$minor == "1.0")) {
##         stop("H2O is not compatible with R 3.1.0\n", "Please change to a newer or older version of R
##             "(For technical details, search the r-devel mailing list\n",
##             "for type.convert changes in R 3.1.0.)")
##     }
##     doc_ip <- Sys.getenv("H2O_R_CMD_CHECK_DOC_EXAMPLES_IP")
##     doc_port <- Sys.getenv("H2O_R_CMD_CHECK_DOC_EXAMPLES_PORT")
##     if (nchar(doc_ip))
##         ip <- doc_ip
##     if (nchar(doc_port))
##         port <- as.numeric(doc_port)
##     warnNthreads <- FALSE
##     tmpConn <- new("H2OConnection", ip = ip, port = port, proxy = proxy,
##         https = https, insecure = insecure, username = username,
##         password = password, cluster_id = cluster_id, cookies = cookies)
##     if (!h2o.clusterIsUp(tmpConn)) {
##         if (!startH2O)
##             stop("Cannot connect to H2O server. Please check that H2O is running at ",
##                 h2o.getBaseURL(tmpConn))
##         else if (ip == "localhost" || ip == "127.0.0.1") {
##             cat("\nH2O is not running yet, starting it now...\n")
##             if (nthreads == -2) {
##                 warnNthreads <- TRUE
##                 nthreads <- 2
##             }
##             stdout <- .h2o.getTmpFile("stdout")
##             .h2o.startJar(nthreads = nthreads, max_memory = max_mem_size,
##                 min_memory = min_mem_size, enable_assertions = enable_assertions,
##                 forceDL = forceDL, license = license, ice_root = ice_root,
##                 stdout = stdout)
##             count <- 0L
##             cat("Starting H2O JVM and connecting: ")
##             while (!h2o.clusterIsUp(conn = tmpConn) && (count <
##                 60L)) {
##                 cat(".")
##                 Sys.sleep(1L)
##                 count <- count + 1L
##             }
##             if (!h2o.clusterIsUp(conn = tmpConn)) {
##                 cat(paste(readLines(stdout), collapse = "\n"),
##                   "\n")
##                 print(tmpConn@ip)
##                 print(tmpConn@port)
##                 rv <- .h2o.doRawGET(conn = tmpConn, urlSuffix = "")
##                 print(rv$curlError)
##                 print(rv$httpStatusCode)
##                 print(rv$curlErrorMessage)
##                 print(system("curl 'http://localhost:54321'"))
##                 stop("H2O failed to start, stopping execution.")
##             }
##         }
##         else stop("Can only start H2O launcher if IP address is localhost.")
##     }
##     conn <- new("H2OConnection", ip = ip, port = port, proxy = proxy,
##         https = https, insecure = insecure, username = username,
##         password = password, cluster_id = cluster_id, cookies = cookies)
##     assign("SERVER", conn, .pkg.env)
##     cat(" Connection successful!\n\n")
##     h2o.clusterInfo()
##     cat("\n")
##     if (strict_version_check && !nchar(Sys.getenv("H2O_DISABLE_STRICT_VERSION_CHECK"))) {
##         verH2O <- h2o.getVersion()
##         verPkg <- packageVersion("h2o")
##         if (verH2O != verPkg) {
##             build_number_H2O <- h2o.getBuildNumber()
##             branch_name_H2O <- h2o.getBranchName()
##             if (is.null(build_number_H2O)) {
##                 stop(sprintf("Version mismatch! H2O is running version %s but h2o-R package is versi
##                   verH2O, toString(verPkg)))
##             }
##             else if (build_number_H2O == "unknown") {
##                 stop(sprintf("Version mismatch! H2O is running version %s but h2o-R package is versi
##                   verH2O, toString(verPkg)))
##             }
##             else if (build_number_H2O == "99999") {
##                 stop((sprintf("Version mismatch! H2O is running version %s but h2o-R package is vers
##                   verH2O, toString(verPkg))))
##             }
##             else {
##                 stop(sprintf("Version mismatch! H2O is running version %s but h2o-R package is versi
##                     verH2O, toString(verPkg), branch_name_H2O,
##                     build_number_H2O))
##              }
##          }
##      }
##      if (warnNthreads) {
##          cat("Note:  As started, H2O is limited to the CRAN default of 2 CPUs.\n")
##          cat("       Shut down and restart H2O as shown below to use all your CPUs.\n")
##          cat("           > h2o.shutdown()\n")
##          cat("           > h2o.init(nthreads = -1)\n")
##          cat("\n")
##      }
##      conn@mutable$session_id <- .init.session_id()
##      invisible(conn)
## }
## <environment: namespace:h2o>
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
##  [1] "Gender"            "Married"           "Dependents"
##  [4] "Education"         "Self_Employed"     "ApplicantIncome"
##  [7] "CoapplicantIncome" "LoanAmount"        "Loan_Amount_Term"
## [10] "Credit_History"    "Property_Area"     "Loan_Status"
```

```
##
  |
  |                                                                 |   0%
  |
  |=                                                                |   2%
  |
  |==                                                               |   3%
  |
  |=====                                                            |   8%
  |
  |================                                                |  24%
  |
  |==========================                                      |  41%
  |
  |=====================================                           |  58%
  |
  |================================================                |  75%
  |
  |===========================================================     |  90%
  |
  |=================================================================| 100%
```

```
##    user  system elapsed
##    0.82    0.00   16.87
```

```
## H2OBinomialMetrics: gbm
## ** Reported on training data. **
##
## MSE:  0.07671854
## RMSE:  0.2769811
## LogLoss:  0.2678692
## Mean Per-Class Error:  0.09329335
## AUC:  0.9759701
## Gini:  0.9519402
##
## Confusion Matrix for F1-optimal threshold:
##          N   Y    Error     Rate
## N      163  29 0.151042  =29/192
## Y       15 407 0.035545  =15/422
## Totals 178 436 0.071661  =44/614
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.630172 0.948718 252
## 2                       max f2  0.514981 0.968188 293
## 3                 max f0point5  0.691076 0.955954 232
## 4                 max accuracy  0.691076 0.929967 232
## 5                max precision  0.980798 1.000000   0
## 6                   max recall  0.426231 1.000000 311
## 7              max specificity  0.980798 1.000000   0
## 8             max absolute_mcc  0.691076 0.840285 232
## 9   max min_per_class_accuracy  0.705909 0.917062 224
## 10 max mean_per_class_accuracy  0.691076 0.926343 232
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
```

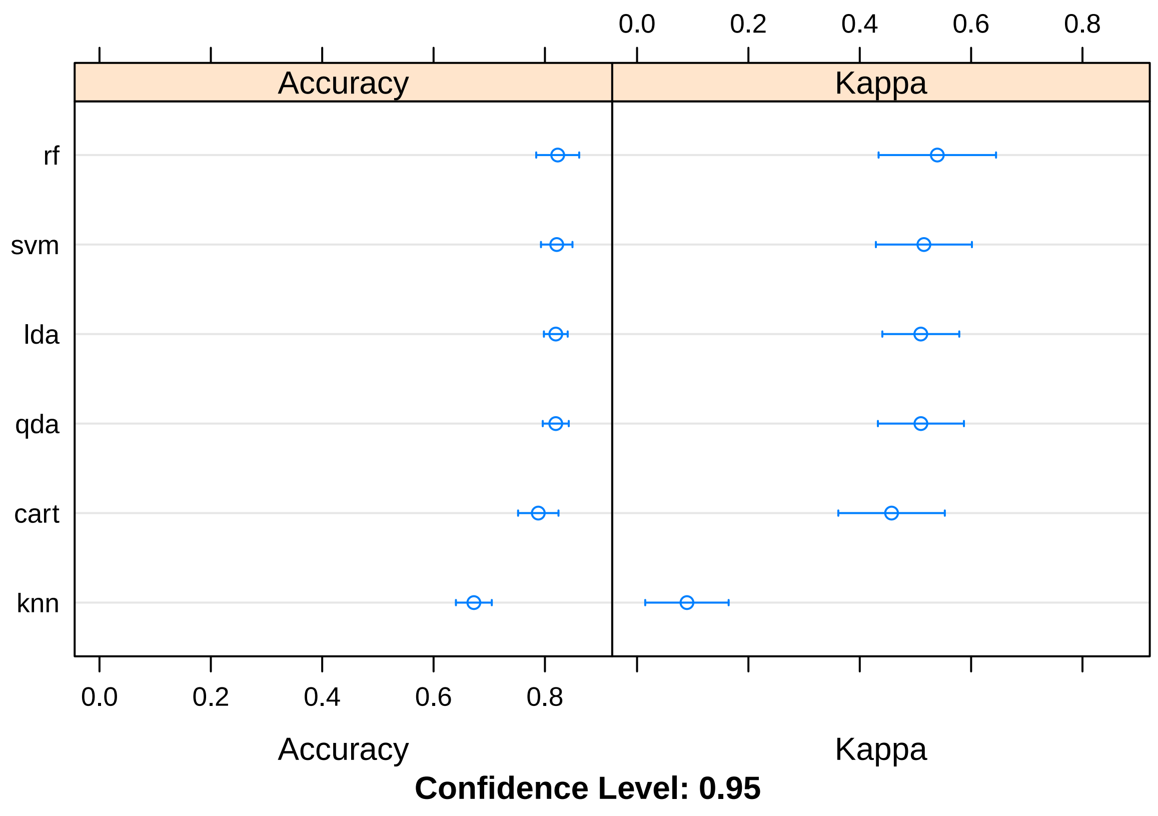
```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
## [1] 0.9759701
```

Select Best Model

We now have 6 models and accuracy estimates for each. Now we will resample all models and plot the accuracy of models. The plot shows that SVM is the most accurate model.

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, knn, svm, rf, qda, cart
## Number of resamples: 10
##
## Accuracy
##         Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## lda  0.7778  0.8037 0.8165 0.8195  0.8356 0.8704    0
## knn  0.6000  0.6481 0.6727 0.6724  0.6918 0.7407    0
## svm  0.7636  0.7870 0.8258 0.8212  0.8472 0.8727    0
## rf   0.7037  0.7989 0.8348 0.8229  0.8539 0.8909    0
## qda  0.7407  0.8075 0.8333 0.8194  0.8356 0.8545    0
## cart 0.7037  0.7465 0.7870 0.7881  0.8356 0.8519    0
##
## Kappa
##           Min. 1st Qu.  Median   Mean 3rd Qu.   Max. NA's
## lda   0.36350 0.44530 0.51680 0.5096  0.5614 0.6736    0
## knn  -0.07651 0.02431 0.08366 0.0896  0.1183 0.2574    0
## svm   0.32610 0.42210 0.52870 0.5152  0.6000 0.6638    0
## rf    0.24080 0.43820 0.58160 0.5393  0.6319 0.7360    0
## qda   0.22700 0.48870 0.54920 0.5098  0.5618 0.6085    0
## cart  0.26660 0.35340 0.44790 0.4571  0.5825 0.6230    0
```



Conclusion

1. Based on various prediction models used Random Forest tops the list with an accuracy of 82%. The dotplot of various models based on accuarcy also confirms that RF tops the list.
2. The visualization of data using decission tree provides following valuable insights which can help the housing finance company in automation of loan eligibility process:

- The top node shows that the basic eligibility criterion for loan eligibility should be customers credit

history. It also displays that 31% have a credit history while 69% do not have credit history.
- Those customers who have credit history of ‘0’, 92% of them will not go for the loan. While only 8% will go ahead with the loan.
- Those customers who do not have a credit history of ‘0’, 21% will go for the loan while 79% will not go for the loan.