Data Analysis

CMM - 703

D.N.H Weerasinghe
IIT ID - 20240474
RGU ID - 2417932

Table Of Contents

Load Packages

```
package_list <-
c("reshape2","plotly","caret","smotefamily","glmnet","randomForest","tinytex"
,"webshot2","shiny","DT")
load_packages <- function(package_name){
  #check if packages are installed. if not install them.
  if (!require(package_name, character.only = TRUE))
install.packages(package_name, dependencies = TRUE)
  #load libraries
  library(package_name,character.only = TRUE)
}

lapply(package_list,load_packages)

## Loading required package: reshape2

## Loading required package: plotly

## Loading required package: ggplot2

##
## Attaching package: 'plotly'
```

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

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

## Loading required package: caret

## Loading required package: lattice

## Loading required package: smotefamily

## Loading required package: glmnet

## Loading required package: Matrix

## Loaded glmnet 4.1-8

## Loading required package: randomForest

## randomForest 4.7-1.2

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

##
## Attaching package: 'randomForest'

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

## Loading required package: tinytex

## Loading required package: webshot2

## Loading required package: shiny

## Loading required package: DT

##
## Attaching package: 'DT'
```

```
## The following objects are masked from 'package:shiny':
##
##     dataTableOutput, renderDataTable

## [[1]]
## [1] "reshape2"  "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [7] "methods"   "base"
##
## [[2]]
##  [1] "plotly"    "ggplot2"   "reshape2"  "stats"     "graphics"
"grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
## [[3]]
##  [1] "caret"     "lattice"   "plotly"    "ggplot2"   "reshape2"  "stats"
##  [7] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[4]]
##  [1] "smotefamily" "caret"       "lattice"     "plotly"      "ggplot2"
##  [6] "reshape2"    "stats"       "graphics"    "grDevices"   "utils"
## [11] "datasets"    "methods"     "base"
##
## [[5]]
##  [1] "glmnet"      "Matrix"      "smotefamily" "caret"       "lattice"
##  [6] "plotly"      "ggplot2"     "reshape2"    "stats"       "graphics"
## [11] "grDevices"   "utils"       "datasets"    "methods"     "base"
##
## [[6]]
##  [1] "randomForest" "glmnet"       "Matrix"       "smotefamily"  "caret"
##  [6] "lattice"      "plotly"       "ggplot2"      "reshape2"     "stats"
## [11] "graphics"     "grDevices"    "utils"        "datasets"     "methods"
## [16] "base"
##
## [[7]]
##  [1] "tinytex"      "randomForest" "glmnet"       "Matrix"
"smotefamily"
##  [6] "caret"        "lattice"      "plotly"       "ggplot2"
"reshape2"
## [11] "stats"        "graphics"     "grDevices"    "utils"
"datasets"
## [16] "methods"      "base"
##
## [[8]]
##  [1] "webshot2"     "tinytex"      "randomForest" "glmnet"       "Matrix"
```

```
##  [6] "smotefamily"  "caret"        "lattice"      "plotly"       "ggplot2"
## [11] "reshape2"     "stats"        "graphics"     "grDevices"    "utils"
## [16] "datasets"     "methods"      "base"
##
## [[9]]
##  [1] "shiny"        "webshot2"     "tinytex"      "randomForest" "glmnet"
##  [6] "Matrix"       "smotefamily"  "caret"        "lattice"      "plotly"
## [11] "ggplot2"      "reshape2"     "stats"        "graphics"
"grDevices"
## [16] "utils"        "datasets"     "methods"      "base"
##
## [[10]]
##  [1] "DT"           "shiny"        "webshot2"     "tinytex"
"randomForest"
##  [6] "glmnet"       "Matrix"       "smotefamily"  "caret"        "lattice"
## [11] "plotly"       "ggplot2"      "reshape2"     "stats"
"graphics"
## [16] "grDevices"    "utils"        "datasets"     "methods"      "base"
```

Set image size for knitted file.

```
suppressMessages(suppressWarnings(webshot::install_phantomjs()))
knitr::opts_chunk$set(
  dev = "png",      # Render figures as PNG images
  dpi = 250,
  fig.retina = 2,
  fig.width = 4,    # Adjust these values as needed (in inches)
  fig.height = 5

)
```

TASK 01

1.1 Generate two important plots

```r
#checking current working directory
getwd()
```

```
## [1] "/Users/naduniweerasinghe"
```

```r
#getall docs in current directory
dir()
```

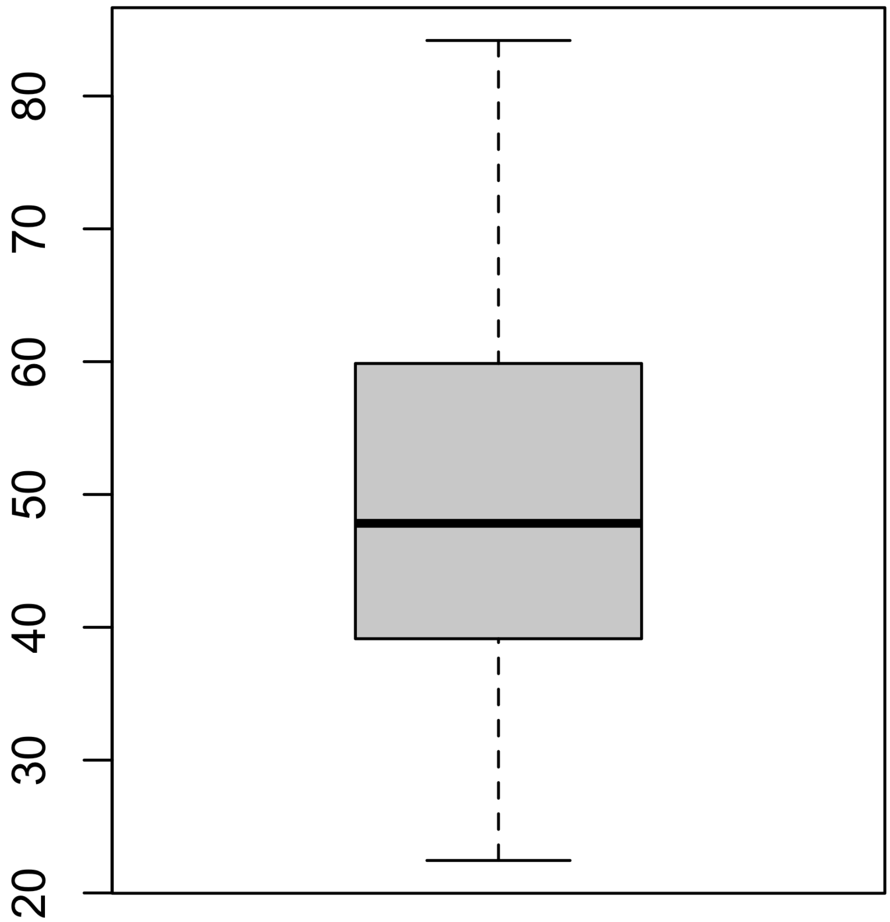
```
##  [1] "2025_day_06.R"
##  [2] "2025_day01.R"
##  [3] "2025_day02.R"
##  [4] "2025_day03.R"
##  [5] "2025_day04.R"
##  [6] "2025_day05.R"
##  [7] "Applications"
##  [8] "cassandra-docker-compose"
##  [9] "Cloud-CMM-707"
## [10] "CMM-702"
## [11] "CMM-703"
## [12] "CMM-703-Data-Analysis23.R"
## [13] "CMM-Data-Analysis-new_files"
## [14] "CMM-Data-Analysis-new.docx"
## [15] "CMM-Data-Analysis-new.pdf"
## [16] "CMM-Data-Analysis-new.Rmd"
## [17] "CMM-Data-Analysis-new.tex"
## [18] "CMM-Data-Analysis-Task-02_files"
## [19] "CMM-Data-Analysis-Task-02.html"
## [20] "CMM-Data-Analysis-Task-02.Rmd"
## [21] "Day03_objects.RData"
## [22] "Desktop"
## [23] "Documents"
## [24] "Downloads"
## [25] "hadoop-docker-compose"
## [26] "IdeaProjects"
## [27] "Library"
## [28] "mapreduce-design-intro"
## [29] "Movies"
## [30] "mt-cars.numbers"
```

```
## [31] "Music"
## [32] "NbaAnalysis"
## [33] "Pictures"
## [34] "Postman"
## [35] "Public"
## [36] "rmd_51a3c41eeaee10bbbd3e7b1f1816ef2e.log"
## [37] "rmd_94f291fdda1932a53ccc17a746ea0678.log"
## [38] "Terminal Saved Output.txt"
## [39] "test_files"
## [40] "test.pdf"
## [41] "test.Rmd"
## [42] "test.tex"
## [43] "Untitled.R"
```

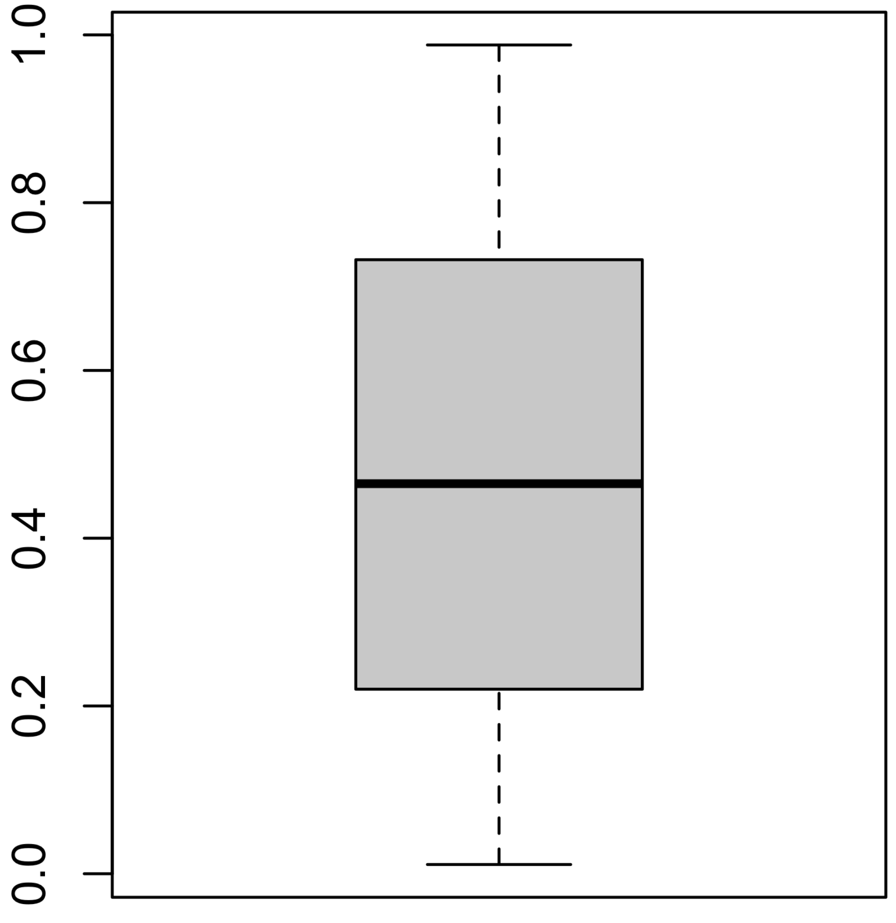
```r
#read csv to view data
candy_data = read.table("/Users/naduniweerasinghe/CMM-703/candy-data.csv",
sep = ",", header = TRUE , quote = "\"", stringsAsFactors = FALSE, na.strings
= c("", "NA"))

summary(candy_data)
```

```
##  competitorname       chocolate          fruity          caramel
##  Length:85          Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  Class :character   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Mode  :character   Median :0.0000   Median :0.0000   Median :0.0000
##                     Mean   :0.4353   Mean   :0.4471   Mean   :0.1647
##                     3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
##                     Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  peanutyalmondy       nougat        crispedricewafer       hard
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.0000   Median :0.00000   Median :0.00000   Median :0.0000
##  Mean   :0.1647   Mean   :0.08235   Mean   :0.08235   Mean   :0.1765
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##       bar            pluribus      sugarpercent     pricepercent
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0110   Min.   :0.0110
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.2200   1st Qu.:0.2550
##  Median :0.0000   Median :1.0000   Median :0.4650   Median :0.4650
##  Mean   :0.2471   Mean   :0.5176   Mean   :0.4786   Mean   :0.4689
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.7320   3rd Qu.:0.6510
##  Max.   :1.0000   Max.   :1.0000   Max.   :0.9880   Max.   :0.9760
##    winpercent
##  Min.   :22.45
```

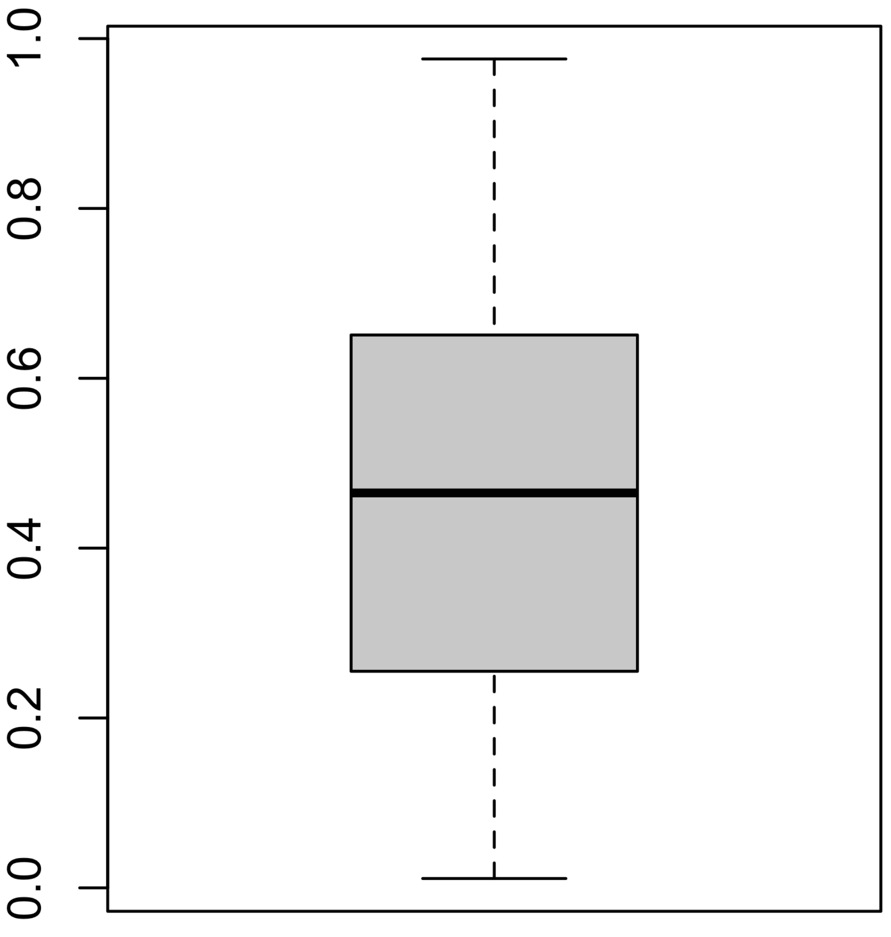
```
##  1st Qu.:39.14
##  Median :47.83
##  Mean   :50.32
##  3rd Qu.:59.86
##  Max.   :84.18

#check if data has missing values
colSums(is.na(candy_data))

##   competitorname        chocolate           fruity          caramel
##                0                0                0                0
##   peanutyalmondy           nougat crispedricewafer             hard
##                0                0                0                0
##              bar         pluribus     sugarpercent     pricepercent
##                0                0                0                0
##       winpercent
##                0

#1. create a boxplot chart using data
boxplot(candy_data$winpercent)
```



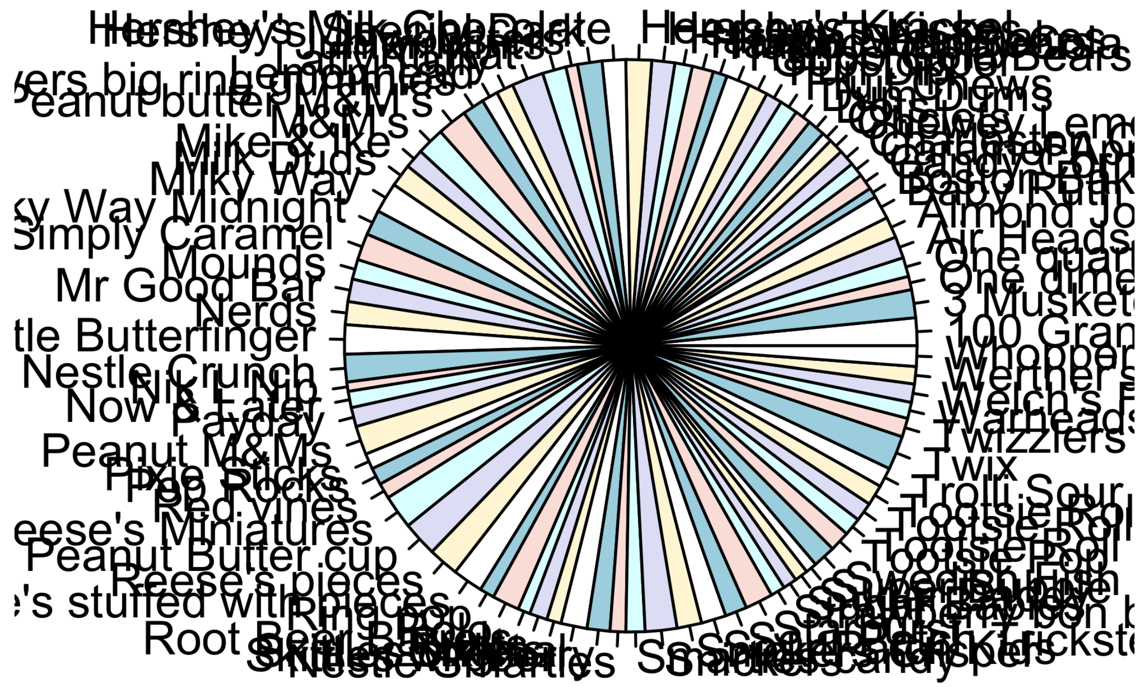

```r
boxplot(candy_data$sugarpercent)
```




```r
boxplot(candy_data$pricepercent)
```




```
#2. create a pie chart using data
pie(candy_data$winpercent,labels = candy_data$competitorname)
```



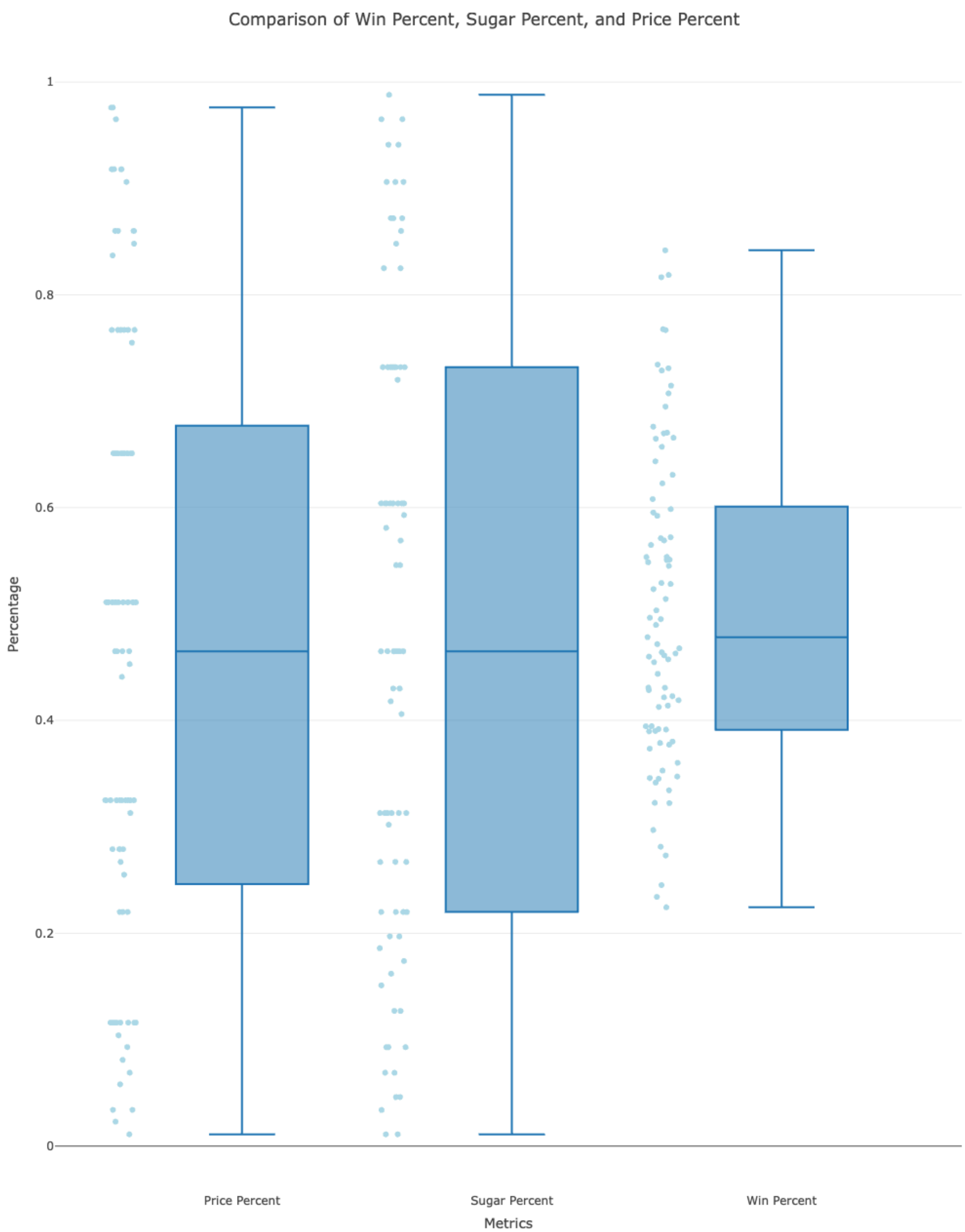
M&M's
Mike & Ike
Milk Duds
Milky Way
Simply Caramel
Mounds
Mr Good Bar
Nerds
Nestle Crunch
Now & Later
Payday
Peanut M&Ms
Pop Rocks
Red vines
Peanut Butter cup
Reese's pieces
Root Beer Barrels
Fruit Chews
Dum Dums
Baby Ruth
Air Heads
One dime
Whopper
Warheads
Twizzlers
Twix
Swedish Fish
Smarties

1.2 Discussion of how the plots can be improved and improved plots.

1.2.1 plot 01- improved version of first plot

To improve we can add y-axis and x-axis names for box plot view. All box plots can be view in same page to make the comparison easy. Add a suitable title to understand the comparison.Other than that we can use a library like plotly to get animated results.Most importantly need to normalize price percent since it is in 0-100 range and the other two are in 0-1 range.

```
# 1. Box plot
# Normalize win percent
candy_data$winpercent <- candy_data$winpercent / 100 # Divide by 100

# Reshape data to long format for multiple box plots
data_long <- data.frame(
  Category = rep(c("Win Percent", "Sugar Percent", "Price Percent"), each =
nrow(candy_data)),
  Value = c(candy_data$winpercent, candy_data$sugarpercen,
candy_data$pricepercent)
)

# Create the box plot
boxplot_candy <- plot_ly(data_long,
  x = ~Category, y = ~Value, type = "box",
  boxpoints = "all", jitter = 0.3, pointpos = -1.8,
  marker = list(color = "lightblue")
) %>% layout(
  title = "Comparison of Win Percent, Sugar Percent, and Price Percent",
  xaxis = list(title = "Metrics"),
  yaxis = list(title = "Percentage")
)

boxplot_candy

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca27312b704/widgetca251ada2a1.html screenshot completed
```


Comparison of Win Percent, Sugar Percent, and Price Percent
1
0.8
0.6
0.4
0.2
0
Percentage
Price Percent
Sugar Percent
Win Percent
Metrics

1.2.2 plot 02- improved version of second plot

In the second plot, unable to identify the each competitor since there are so many. Names are not clear. Percentages cannot be viewed & need a proper title as well.Most importantly the pie chart is not suitable to view the win percentage against the competitor since winning percentage is not a part of the whole. We can use a bar plot instead of that. Can add the types of ingredients they have used or other details.

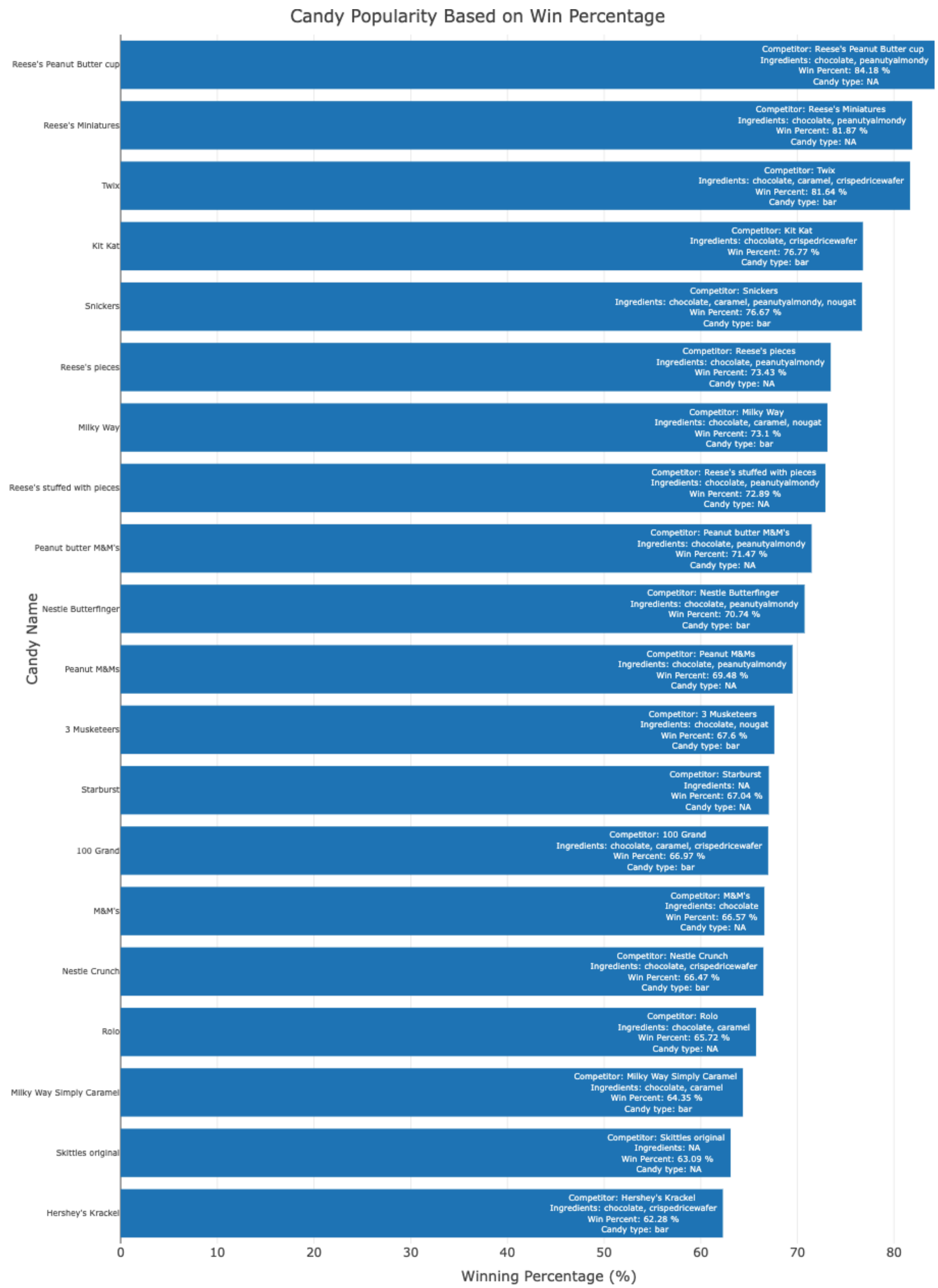
```
#2. Bar plot

ingredients <-
c("chocolate","caramel","peanutyalmondy","nougat","crispedricewafer")
candy_type <- c("hard","bar")
colnames(candy_data)

##  [1] "competitorname"   "chocolate"        "fruity"           "caramel"
##  [5] "peanutyalmondy"   "nougat"           "crispedricewafer" "hard"
##  [9] "bar"              "pluribus"         "sugarpercent"
"pricepercent"
## [13] "winpercent"

setdiff(ingredients,colnames(candy_data))

## character(0)

candy_data$ingred_list <- apply(candy_data[,ingredients],1,function(candy){
  in_list <- as.array(names(candy)[candy == 1])
  return(if (length(in_list) == 0)  "NA" else paste(in_list, collapse = ",
"))
})


candy_data$candy_type <- apply(candy_data[,candy_type],1,function(candy){
  tp_list <- as.array(names(candy)[candy == 1])
  return(if (length(tp_list) == 0)  "NA" else paste(tp_list, collapse = ",
"))
})

candy_data$hover_text <- paste("Competitor:", candy_data$competitorname,
"<br>","Ingredients:", candy_data$ingred_list, "<br>","Win Percent:",
round(candy_data$winpercent*100,2), "%","<br>", "Candy
type:",candy_data$candy_type)

sorted_cndy_data <- candy_data[order(-candy_data$winpercent), ][1:20, ]
```

```
sorted_cndy_data

##                  competitorname chocolate fruity caramel peanutyalmondy
nougat
## 53   Reese's Peanut Butter cup         1      0       0              1
0
## 52          Reese's Miniatures         1      0       0              1
0
## 80                        Twix         1      0       1              0
0
## 29                     Kit Kat         1      0       0              0
0
## 65                    Snickers         1      0       1              1
1
## 54              Reese's pieces         1      0       0              1
0
## 37                   Milky Way         1      0       1              0
1
## 55 Reese's stuffed with pieces         1      0       0              1
0
## 33         Peanut butter M&M's         1      0       0              1
0
## 43         Nestle Butterfinger         1      0       0              1
0
## 48                 Peanut M&Ms         1      0       0              1
0
## 2                 3 Musketeers         1      0       0              0
1
## 69                   Starburst         0      1       0              0
0
## 1                    100 Grand         1      0       1              0
0
## 34                       M&M's         1      0       0              0
0
## 44               Nestle Crunch         1      0       0              0
0
## 57                        Rolo         1      0       1              0
0
## 39    Milky Way Simply Caramel         1      0       1              0
0
## 61           Skittles original         0      1       0              0
0
## 24           Hershey's Krackel         1      0       0              0
```

```
0
##    crispedricewafer hard bar pluribus sugarpercent pricepercent winpercent
## 53                0    0   0        0        0.720        0.651  0.8418029
## 52                0    0   0        0        0.034        0.279  0.8186626
## 80                1    0   1        0        0.546        0.906  0.8164291
## 29                1    0   1        0        0.313        0.511  0.7676860
## 65                0    0   1        0        0.546        0.651  0.7667378
## 54                0    0   0        1        0.406        0.651  0.7343499
## 37                0    0   1        0        0.604        0.651  0.7309956
## 55                0    0   0        0        0.988        0.651  0.7288790
## 33                0    0   0        1        0.825        0.651  0.7146505
## 43                0    0   1        0        0.604        0.767  0.7073564
## 48                0    0   0        1        0.593        0.651  0.6948379
## 2                 0    0   1        0        0.604        0.511  0.6760294
## 69                0    0   0        1        0.151        0.220  0.6703763
## 1                 1    0   1        0        0.732        0.860  0.6697173
## 34                0    0   0        1        0.825        0.651  0.6657458
## 44                1    0   1        0        0.313        0.767  0.6647068
## 57                0    0   0        1        0.860        0.860  0.6571629
## 39                0    0   1        0        0.965        0.860  0.6435334
## 61                0    0   0        1        0.941        0.220  0.6308514
## 24                1    0   1        0        0.430        0.918  0.6228448
##                                 ingred_list candy_type
## 53                 chocolate, peanutyalmondy         NA
## 52                 chocolate, peanutyalmondy         NA
## 80       chocolate, caramel, crispedricewafer        bar
## 29                chocolate, crispedricewafer        bar
## 65 chocolate, caramel, peanutyalmondy, nougat        bar
## 54                 chocolate, peanutyalmondy         NA
## 37                 chocolate, caramel, nougat        bar
## 55                 chocolate, peanutyalmondy         NA
## 33                 chocolate, peanutyalmondy         NA
## 43                 chocolate, peanutyalmondy        bar
## 48                 chocolate, peanutyalmondy         NA
## 2                          chocolate, nougat        bar
## 69                                        NA         NA
## 1        chocolate, caramel, crispedricewafer        bar
## 34                                 chocolate         NA
## 44                chocolate, crispedricewafer        bar
## 57                        chocolate, caramel         NA
## 39                        chocolate, caramel        bar
## 61                                        NA         NA
## 24                chocolate, crispedricewafer        bar
##
```

```
hover_text
## 53   Competitor: Reese's Peanut Butter cup <br> Ingredients: chocolate,
peanutyalmondy <br> Win Percent: 84.18 % <br> Candy type: NA
## 52          Competitor: Reese's Miniatures <br> Ingredients: chocolate,
peanutyalmondy <br> Win Percent: 81.87 % <br> Candy type: NA
## 80            Competitor: Twix <br> Ingredients: chocolate, caramel,
crispedricewafer <br> Win Percent: 81.64 % <br> Candy type: bar
## 29                  Competitor: Kit Kat <br> Ingredients: chocolate,
crispedricewafer <br> Win Percent: 76.77 % <br> Candy type: bar
## 65  Competitor: Snickers <br> Ingredients: chocolate, caramel,
peanutyalmondy, nougat <br> Win Percent: 76.67 % <br> Candy type: bar
## 54              Competitor: Reese's pieces <br> Ingredients: chocolate,
peanutyalmondy <br> Win Percent: 73.43 % <br> Candy type: NA
## 37                  Competitor: Milky Way <br> Ingredients: chocolate,
caramel, nougat <br> Win Percent: 73.1 % <br> Candy type: bar
## 55 Competitor: Reese's stuffed with pieces <br> Ingredients: chocolate,
peanutyalmondy <br> Win Percent: 72.89 % <br> Candy type: NA
## 33         Competitor: Peanut butter M&M's <br> Ingredients: chocolate,
peanutyalmondy <br> Win Percent: 71.47 % <br> Candy type: NA
## 43        Competitor: Nestle Butterfinger <br> Ingredients: chocolate,
peanutyalmondy <br> Win Percent: 70.74 % <br> Candy type: bar
## 48                 Competitor: Peanut M&Ms <br> Ingredients: chocolate,
peanutyalmondy <br> Win Percent: 69.48 % <br> Candy type: NA
## 2                         Competitor: 3 Musketeers <br> Ingredients:
chocolate, nougat <br> Win Percent: 67.6 % <br> Candy type: bar
## 69                                          Competitor: Starburst <br>
Ingredients: NA <br> Win Percent: 67.04 % <br> Candy type: NA
## 1        Competitor: 100 Grand <br> Ingredients: chocolate, caramel,
crispedricewafer <br> Win Percent: 66.97 % <br> Candy type: bar
## 34                                          Competitor: M&M's <br>
Ingredients: chocolate <br> Win Percent: 66.57 % <br> Candy type: NA
## 44            Competitor: Nestle Crunch <br> Ingredients: chocolate,
crispedricewafer <br> Win Percent: 66.47 % <br> Candy type: bar
## 57                              Competitor: Rolo <br> Ingredients:
chocolate, caramel <br> Win Percent: 65.72 % <br> Candy type: NA
## 39          Competitor: Milky Way Simply Caramel <br> Ingredients:
chocolate, caramel <br> Win Percent: 64.35 % <br> Candy type: bar
## 61                                 Competitor: Skittles original <br>
Ingredients: NA <br> Win Percent: 63.09 % <br> Candy type: NA
## 24        Competitor: Hershey's Krackel <br> Ingredients: chocolate,
crispedricewafer <br> Win Percent: 62.28 % <br> Candy type: bar

fig <- plot_ly(data.frame(sorted_cndy_data),
  x = ~winpercent*100,
```

```
  y = ~reorder(competitorname,winpercent),#sortcompetitor names based on the
winning %
  type = "bar",
  text = ~hover_text,
  hoverinfo = "text"
) %>% layout(
  title = "Candy Popularity Based on Win Percentage",
  xaxis = list(title = "Winning Percentage (%)"),
  yaxis = list(title = "Candy Name", tickfont = list(size = 8)),
  margin = list(l = 150)  # Adjust left margin for readability
)

fig

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca231b33566/widgetca26c76d507.html screenshot completed
```


Candy Popularity Based on Win Percentage
Candy Name
Reese's Peanut Butter cup
Competitor: Reese's Peanut Butter cup
Ingredients: chocolate, peanutyalmondy
Win Percent: 84.18 %
Candy type: NA
Reese's Miniatures
Competitor: Reese's Miniatures
Ingredients: chocolate, peanutyalmondy
Win Percent: 81.87 %
Candy type: NA
Twix
Competitor: Twix
Ingredients: chocolate, caramel, crispedricewafer
Win Percent: 81.64 %
Candy type: bar
Kit Kat
Competitor: Kit Kat
Ingredients: chocolate, crispedricewafer
Win Percent: 76.77 %
Candy type: bar
Snickers
Competitor: Snickers
Ingredients: chocolate, caramel, peanutyalmondy, nougat
Win Percent: 76.67 %
Candy type: bar
Reese's pieces
Competitor: Reese's pieces
Ingredients: chocolate, peanutyalmondy
Win Percent: 73.43 %
Candy type: NA
Milky Way
Competitor: Milky Way
Ingredients: chocolate, caramel, nougat
Win Percent: 73.1 %
Candy type: bar
Reese's stuffed with pieces
Competitor: Reese's stuffed with pieces
Ingredients: chocolate, peanutyalmondy
Win Percent: 72.89 %
Candy type: NA
Peanut butter M&M's
Competitor: Peanut butter M&M's
Ingredients: chocolate, peanutyalmondy
Win Percent: 71.47 %
Candy type: NA
Nestle Butterfinger
Competitor: Nestle Butterfinger
Ingredients: chocolate, peanutyalmondy
Win Percent: 70.74 %
Candy type: bar
Peanut M&Ms
Competitor: Peanut M&Ms
Ingredients: chocolate, peanutyalmondy
Win Percent: 69.48 %
Candy type: NA
3 Musketeers
Competitor: 3 Musketeers
Ingredients: chocolate, nougat
Win Percent: 67.6 %
Candy type: bar
Starburst
Competitor: Starburst
Ingredients: NA
Win Percent: 67.04 %
Candy type: NA
100 Grand
Competitor: 100 Grand
Ingredients: chocolate, caramel, crispedricewafer
Win Percent: 66.97 %
Candy type: bar
M&M's
Competitor: M&M's
Ingredients: chocolate
Win Percent: 66.57 %
Candy type: NA
Nestle Crunch
Competitor: Nestle Crunch
Ingredients: chocolate, crispedricewafer
Win Percent: 66.47 %
Candy type: bar
Rolo
Competitor: Rolo
Ingredients: chocolate, caramel
Win Percent: 65.72 %
Candy type: NA
Milky Way Simply Caramel
Competitor: Milky Way Simply Caramel
Ingredients: chocolate, caramel
Win Percent: 64.35 %
Candy type: bar
Skittles original
Competitor: Skittles original
Ingredients: NA
Win Percent: 63.09 %
Candy type: NA
Hershey's Krackel
Competitor: Hershey's Krackel
Ingredients: chocolate, crispedricewafer
Win Percent: 62.28 %
Candy type: bar
0
10
20
30
40
50
60
70
80
Winning Percentage (%)

TASK 02

Task 2.1

2.1.1 Load the data set to the notebook

```
#load the bank churn dataset
bank_churn  <- read.table("/Users/naduniweerasinghe/CMM-703/Bank_Churn.csv",
sep = "," , header = TRUE , quote = "\"", stringsAsFactors = FALSE,
na.strings = c("", "NA"))
```

2.1.2 View first few records of data in the data set

```
#view first few data rows in data set
head(bank_churn)

##   CustomerId  Surname CreditScore Geography Gender Age Tenure   Balance
## 1   15634602 Hargrave         619    France Female  42      2      0.00
## 2   15647311     Hill         608     Spain Female  41      1  83807.86
## 3   15619304     Onio         502    France Female  42      8 159660.80
## 4   15701354     Boni         699    France Female  39      1      0.00
## 5   15737888 Mitchell         850     Spain Female  43      2 125510.82
## 6   15574012      Chu         645     Spain   Male  44      8 113755.78
##   NumOfProducts HasCrCard IsActiveMember EstimatedSalary Exited
## 1             1         1              1       101348.88      1
## 2             1         0              1       112542.58      0
## 3             3         1              0       113931.57      1
## 4             2         0              0        93826.63      0
## 5             1         1              1        79084.10      0
## 6             2         1              0       149756.71      1
```

2.1.3 View last few records in the data set

```
#view last few data rows in dataset
tail(bank_churn)

##       CustomerId   Surname CreditScore Geography Gender Age Tenure
Balance
## 9995    15719294      Wood         800    France Female  29      2
0.00
## 9996    15606229  Obijiaku         771    France   Male  39      5
0.00
## 9997    15569892 Johnstone         516    France   Male  35     10
57369.61
## 9998    15584532       Liu         709    France Female  36      7
0.00
## 9999    15682355 Sabbatini         772   Germany   Male  42      3
75075.31
## 10000   15628319    Walker         792    France Female  28      4
130142.79
##       NumOfProducts HasCrCard IsActiveMember EstimatedSalary Exited
## 9995              2         0              0       167773.55      0
## 9996              2         1              0        96270.64      0
## 9997              1         1              1       101699.77      0
## 9998              1         0              1        42085.58      1
## 9999              2         1              0        92888.52      1
## 10000             1         1              0        38190.78      0
```

2.1.4 View summary of data

```r
#view summary of data
summary(bank_churn)
```

```
##    CustomerId         Surname           CreditScore     Geography        
##  Min.   :15565701   Length:10000       Min.   :350.0   Length:10000      
##  1st Qu.:15628528   Class :character   1st Qu.:584.0   Class :character  
##  Median :15690738   Mode  :character   Median :652.0   Mode  :character  
##  Mean   :15690941                      Mean   :650.5                     
##  3rd Qu.:15753234                      3rd Qu.:718.0                     
##  Max.   :15815690                      Max.   :850.0                     
##     Gender               Age            Tenure          Balance      
##  Length:10000       Min.   :18.00   Min.   : 0.000   Min.   :     0  
##  Class :character   1st Qu.:32.00   1st Qu.: 3.000   1st Qu.:     0  
##  Mode  :character   Median :37.00   Median : 5.000   Median : 97199  
##                     Mean   :38.92   Mean   : 5.013   Mean   : 76486  
##                     3rd Qu.:44.00   3rd Qu.: 7.000   3rd Qu.:127644  
##                     Max.   :92.00   Max.   :10.000   Max.   :250898  
##  NumOfProducts    HasCrCard      IsActiveMember   EstimatedSalary    
##  Min.   :1.00   Min.   :0.0000   Min.   :0.0000   Min.   :    11.58  
##  1st Qu.:1.00   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 51002.11  
##  Median :1.00   Median :1.0000   Median :1.0000   Median :100193.91  
##  Mean   :1.53   Mean   :0.7055   Mean   :0.5151   Mean   :100090.24  
##  3rd Qu.:2.00   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:149388.25  
##  Max.   :4.00   Max.   :1.0000   Max.   :1.0000   Max.   :199992.48  
##      Exited      
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :0.0000  
##  Mean   :0.2037  
##  3rd Qu.:0.0000  
##  Max.   :1.0000  
```

2.1.5 Check if any feature of data has missing values

```
#check for missing values in dataset
colSums(is.na(bank_churn))

##      CustomerId         Surname     CreditScore       Geography
Gender
##               0               0               0               0
0
##             Age          Tenure         Balance   NumOfProducts
HasCrCard
##               0               0               0               0
0
##  IsActiveMember EstimatedSalary          Exited
##               0               0               0
```

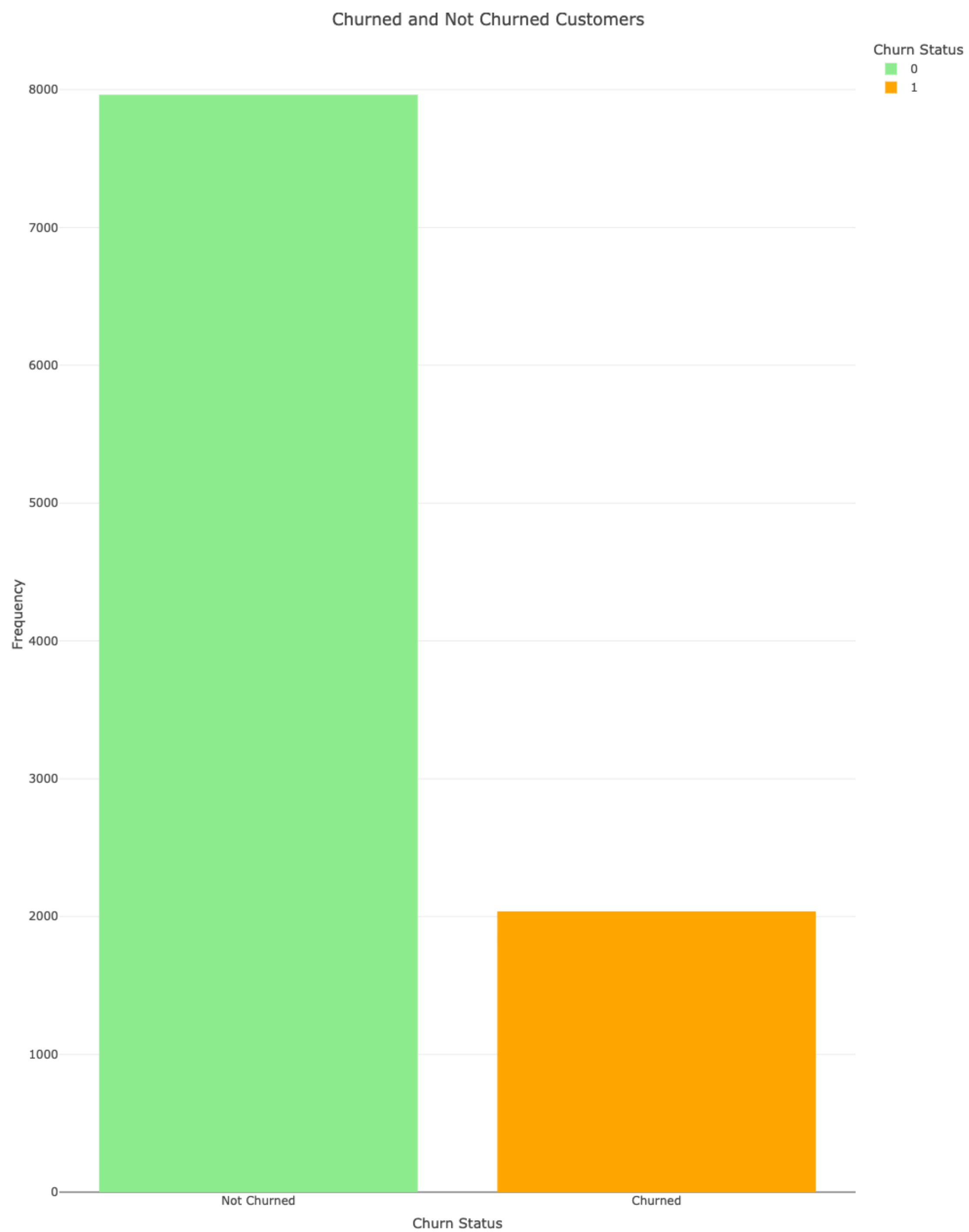
Since in above code result data shows that there aren't any missing values, visualize all data in data set to understand the spread.

2.1.6 View features in plots to get an idea about data

```
#since in above code result data shows that there aren't any missing values,
visualize all data in dataset to understand the spread.

#view all customer account balance for who customers not churned
colnames(bank_churn)

##  [1] "CustomerId"      "Surname"         "CreditScore"     "Geography"
##  [5] "Gender"          "Age"             "Tenure"          "Balance"
##  [9] "NumOfProducts"   "HasCrCard"       "IsActiveMember"
"EstimatedSalary"
## [13] "Exited"
```

2.1.6.1 View Churned and Not Churned Customers

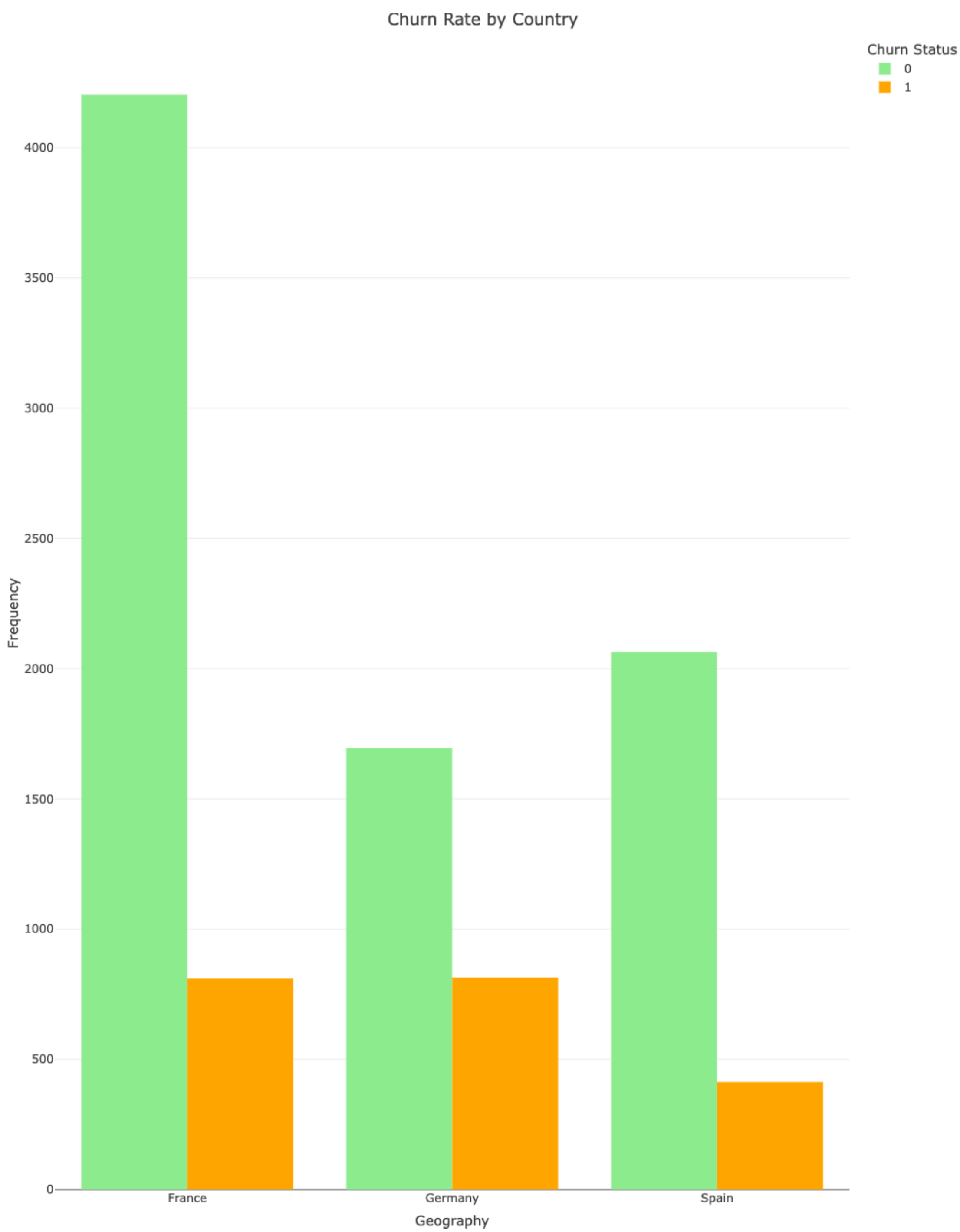
```
#barplot for churn and not churn
churn_not_churned <- plot_ly(
  as.data.frame(table(bank_churn$Exited)),
  x = ~Var1 ,
  y = ~Freq,
  type = "bar",
  color = ~factor(Var1),
  colors = c("1" = "orange", "0" = "lightgreen")
) %>%
  layout(
    title = "Churned and Not Churned Customers",
    xaxis = list(
      title = "Churn Status",
      tickvals = c(0, 1),
      ticktext = c("Not Churned", "Churned")
    ),
    yaxis = list(title = "Frequency"),
    legend = list(title = list(text = 'Churn Status'))
  )

#view the plot
churn_not_churned

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca25e3f63da/widgetca213b4ad8d.html screenshot completed
```



Churned and Not Churned Customers
Churn Status
0
1
8000
7000
6000
5000
4000
3000
2000
1000
0
Frequency
Not Churned
Churned
Churn Status

2.1.6.2 View Churn Customers Percentage By Country

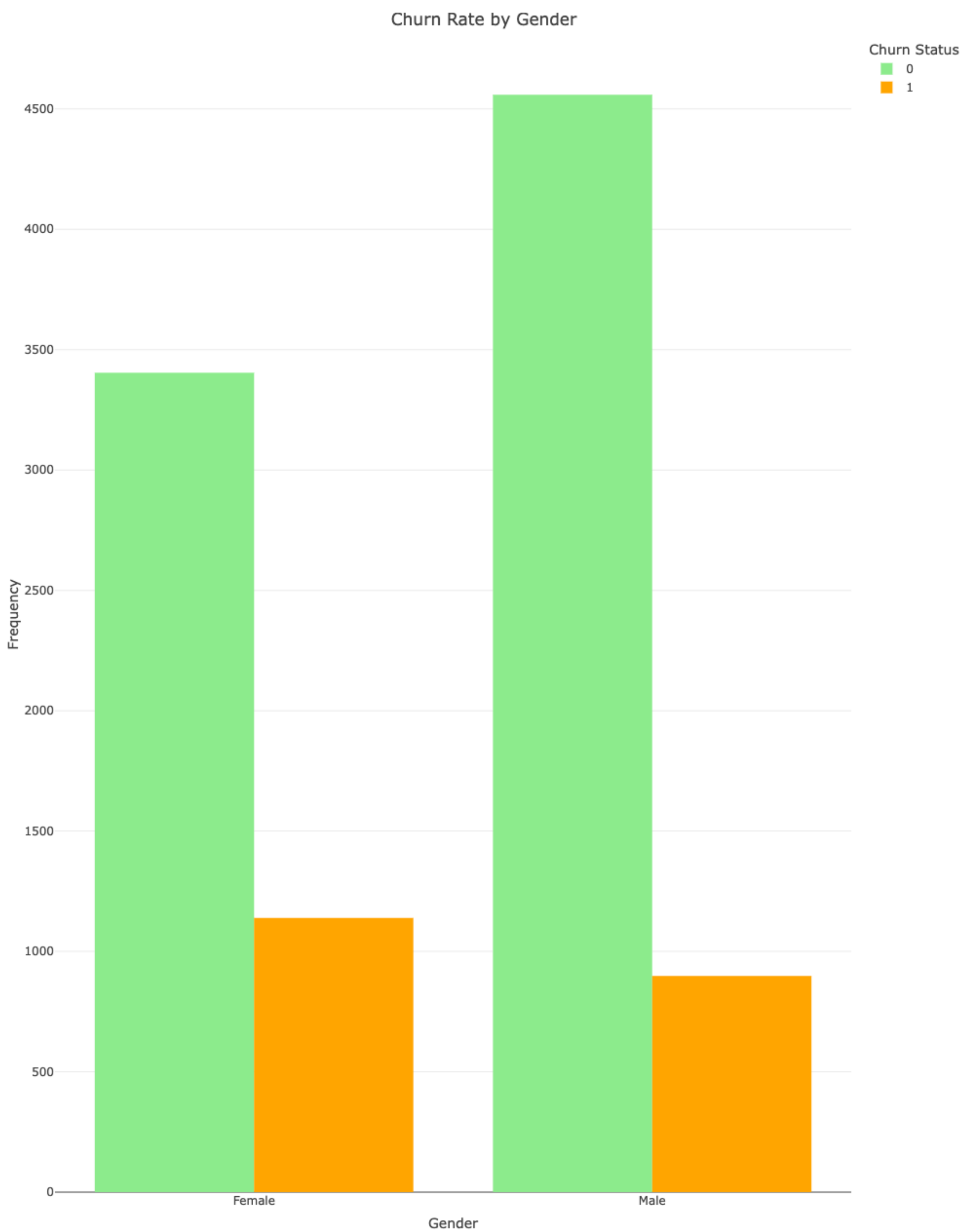
```r
#count churned customer data by country

#barplot for churn and not churn data by country
ch_by_country <- plot_ly(
  as.data.frame(table(bank_churn$Exited , bank_churn$Geography)),
  x = ~ Var2,
  y = ~ Freq,
  type = "bar",
  color = ~ factor(Var1),
  colors = c("1" = "orange", "0" = "lightgreen")
) %>%
  layout(
    title = "Churn Rate by Country",
    xaxis = list(title = "Geography"),
    yaxis = list(title = "Frequency"),
    legend = list(title = list(text = 'Churn Status'))
  )

#view the plot
ch_by_country

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca2604c9037/widgetca242903c45.html screenshot completed
```



Churn Rate by Country
Churn Status
0
1
Frequency
4000
3500
3000
2500
2000
1500
1000
500
0
France
Germany
Spain
Geography

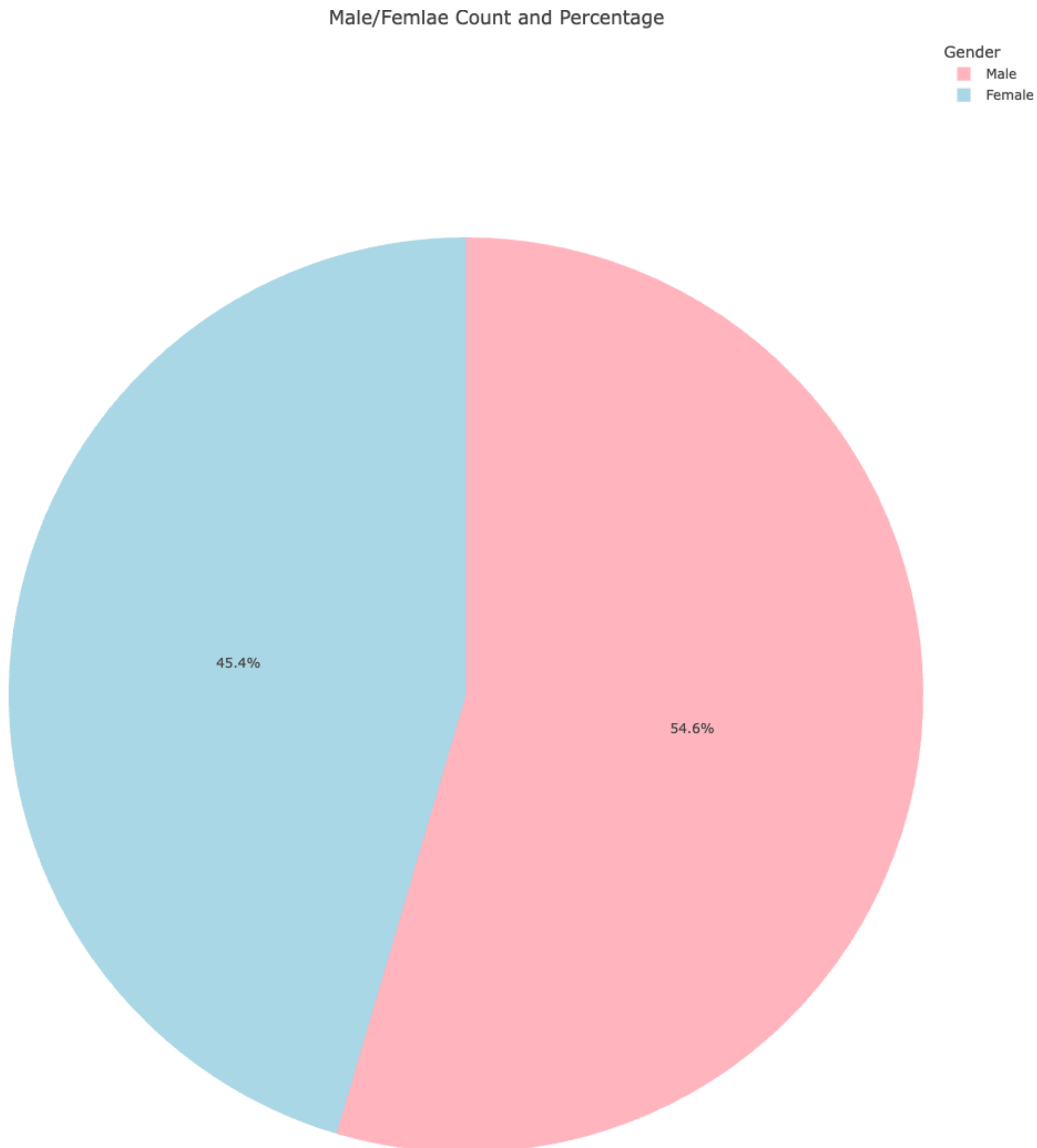
2.1.6.3 View Churn Customers By Gender

```
#count churned customer data by Gender
table(bank_churn$Exited , bank_churn$Gender)

##
##      Female Male
##   0    3404 4559
##   1    1139  898

#barplot for churn and not churn rate by gender
ch_by_gender <- plot_ly(
  as.data.frame(table(bank_churn$Exited , bank_churn$Gender)),
  x = ~ Var2,
  y = ~ Freq,
  type = "bar",
  color = ~ factor(Var1),
  colors = c("1" = "orange", "0" = "lightgreen")
) %>%
  layout(
    title = "Churn Rate by Gender",
    xaxis = list(title = "Gender"),
    yaxis = list(title = "Frequency"),
    legend = list(title = list(text = 'Churn Status'))
  )

#view the plot
ch_by_gender

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca268ff68b3/widgetca25832f59f.html screenshot completed
```


Churn Rate by Gender
Churn Status
0
1
Frequency
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Female
Male
Gender

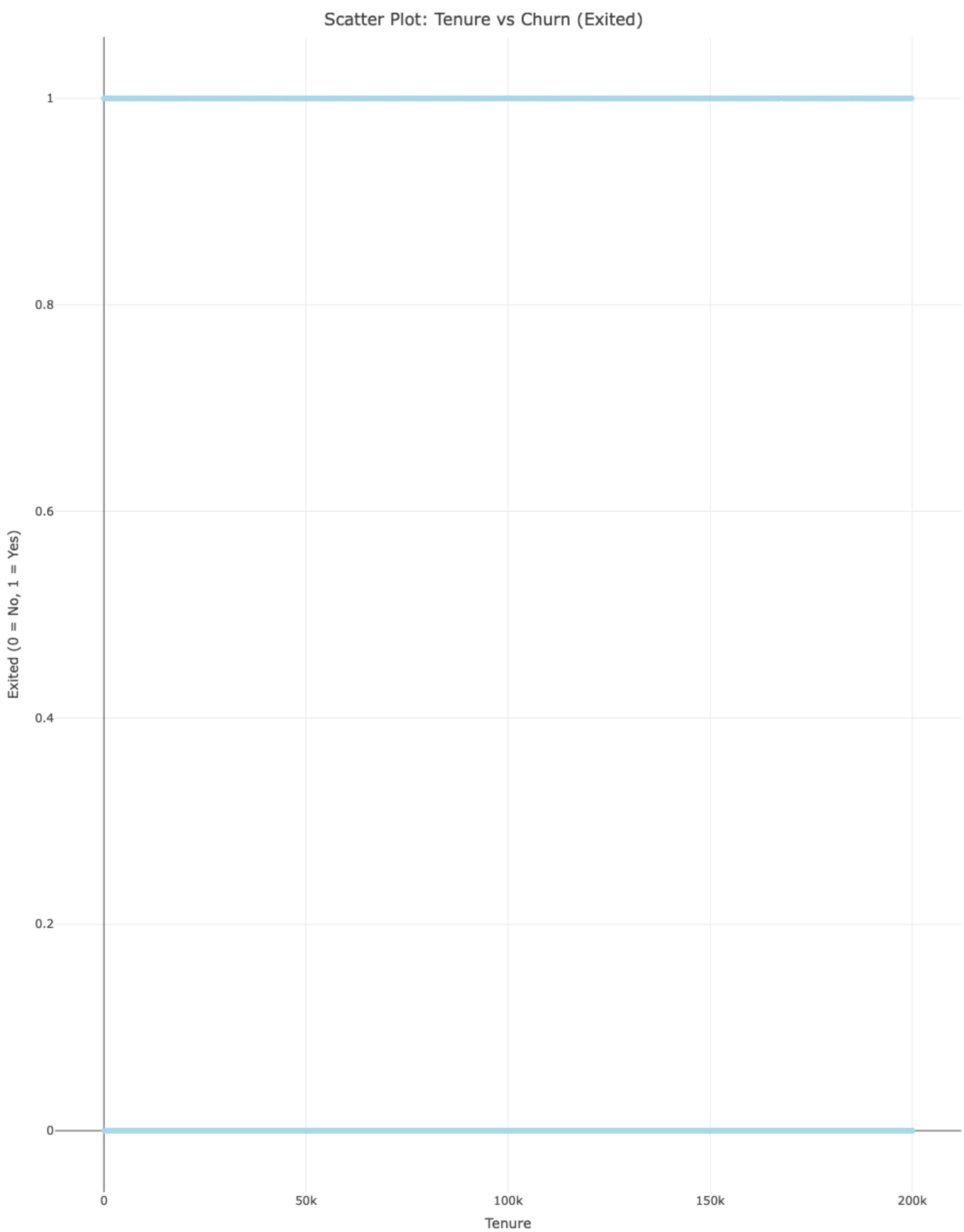
2.1.6.4 View Customers Percentages By Gender

```
#pie chart to view customer male female percentage
ch_by_gender <- plot_ly(
  as.data.frame(table(bank_churn$Gender)),
  labels = ~ Var1,
  values = ~ Freq,
  type = "pie",
  marker = list(colors = c("lightblue", "lightpink"))
) %>%
  layout(title = "Male/Femlae Count and Percentage", legend = list(title =
list(text = 'Gender')))

#view the plot
ch_by_gender

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca225e1b82f/widgetca26032917.html screenshot completed
```


Male/Femlae Count and Percentage
Gender
Male
Female
45.4%
54.6%

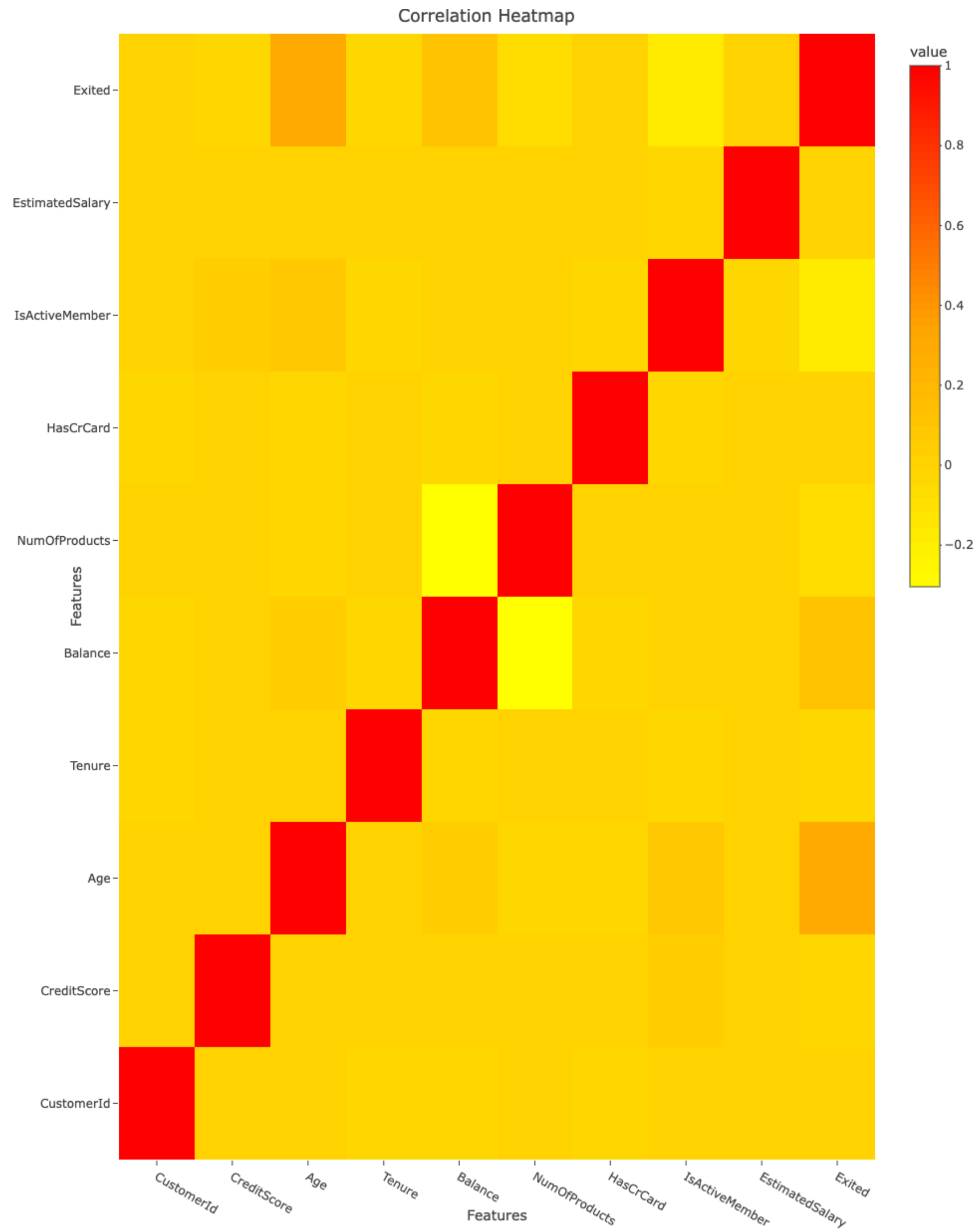
2.1.6.5 Churn Status Against Estimated Salary

```
plot_ly(data = bank_churn, x = ~EstimatedSalary, y = ~Exited, type =
'scatter', mode = 'markers',
        marker = list(color = 'lightblue')) %>%
  layout(title = 'Scatter Plot: Tenure vs Churn (Exited)',
         xaxis = list(title = 'Tenure'),
         yaxis = list(title = 'Exited (0 = No, 1 = Yes)'))

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca24c955256/widgetca25f50b361.html screenshot completed
```


Scatter Plot: Tenure vs Churn (Exited)
1
0.8
0.6
0.4
0.2
0
Exited (0 = No, 1 = Yes)
0
50k
100k
150k
200k
Tenure

2.1.6.6 View Correlation Between Features

```
#view correlation between features

#get only numeric columns
num_bank_churned <- bank_churn[sapply(bank_churn, is.numeric)]

#calculate the correlation
corelation_m <- cor(num_bank_churned, use = "complete.obs")

#convert values to long format for Plotly
corelation_d <- melt(corelation_m)

correlation_plot <- plot_ly(
  data = corelation_d,
  #corelation data
  x = ~ Var1,
  #feature
  y = ~ Var2,
  #feature
  z = ~ value,
  # corelation value
  type = "heatmap",
  #plot type
  colorscale = list(c(0, 0.5, 1), #position of colors (0 = lowest, 1 = 
highest)
                    c("yellow", "orange", "red")  #color progression)
  )) %>%
    layout(
      title = "Correlation Heatmap",
      #title of plot
      xaxis = list(title = "Features"),
      yaxis = list(title = "Features")
    )

  #view correlation plot
  correlation_plot

## 
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca2638e5318/widgetca214f17fb8.html screenshot completed
```



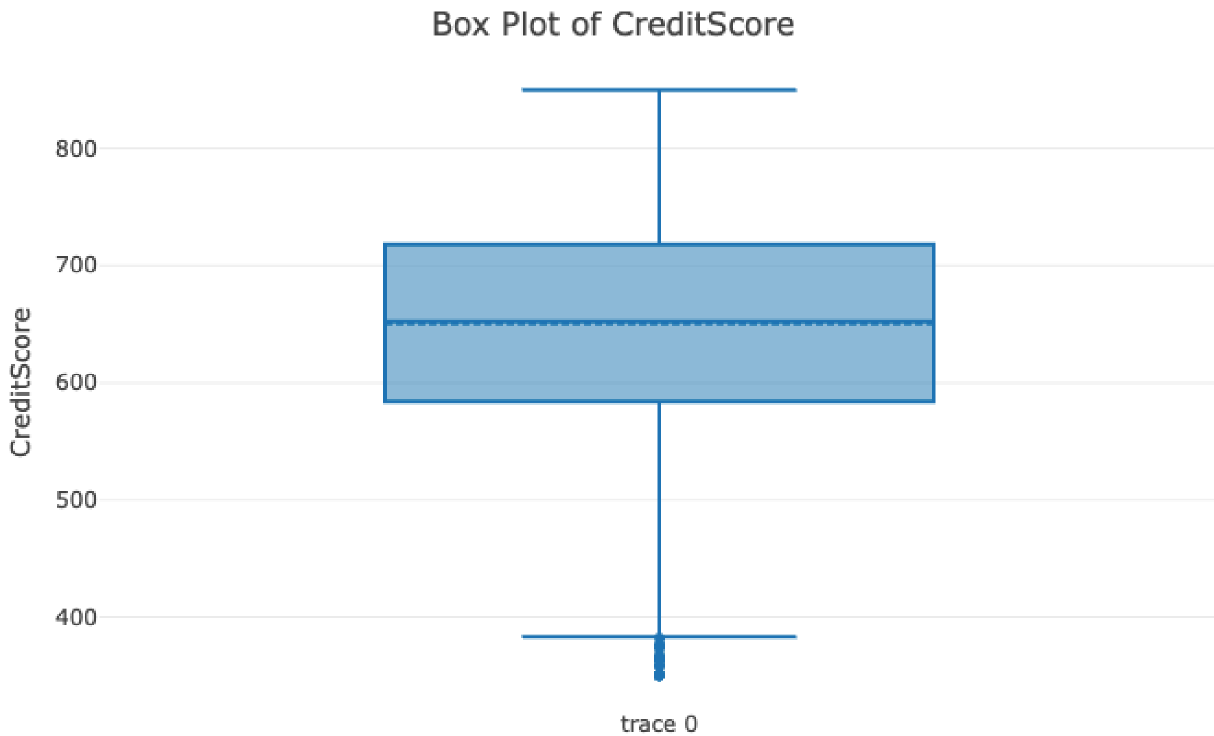
Correlation Heatmap
value
1
0.8
0.6
0.4
0.2
0
−0.2
Exited
EstimatedSalary
IsActiveMember
HasCrCard
NumOfProducts
Balance
Tenure
Age
CreditScore
CustomerId
Features
CustomerId
CreditScore
Age
Tenure
Balance
NumOfProducts
HasCrCard
IsActiveMember
EstimatedSalary
Exited
Features

2.1.7 Remove irrelevant features like customer name and customerId

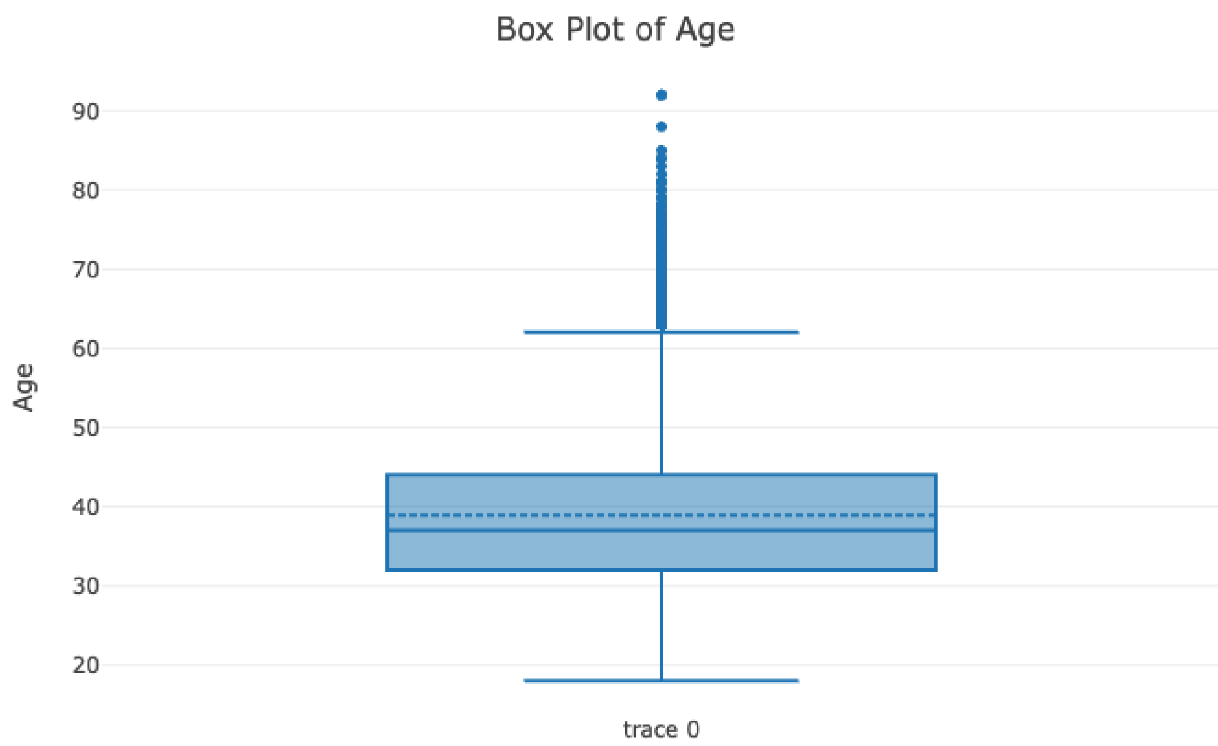
```
bank_churn <- bank_churn[, c(
  "Geography",
  "CreditScore",
  "Gender",
  "Age",
  "Tenure",
  "Balance",
  "NumOfProducts",
  "HasCrCard",
  "IsActiveMember",
  "EstimatedSalary",
  "Exited"
)]
```

2.1.8 Check for Outliers in Numerical Data

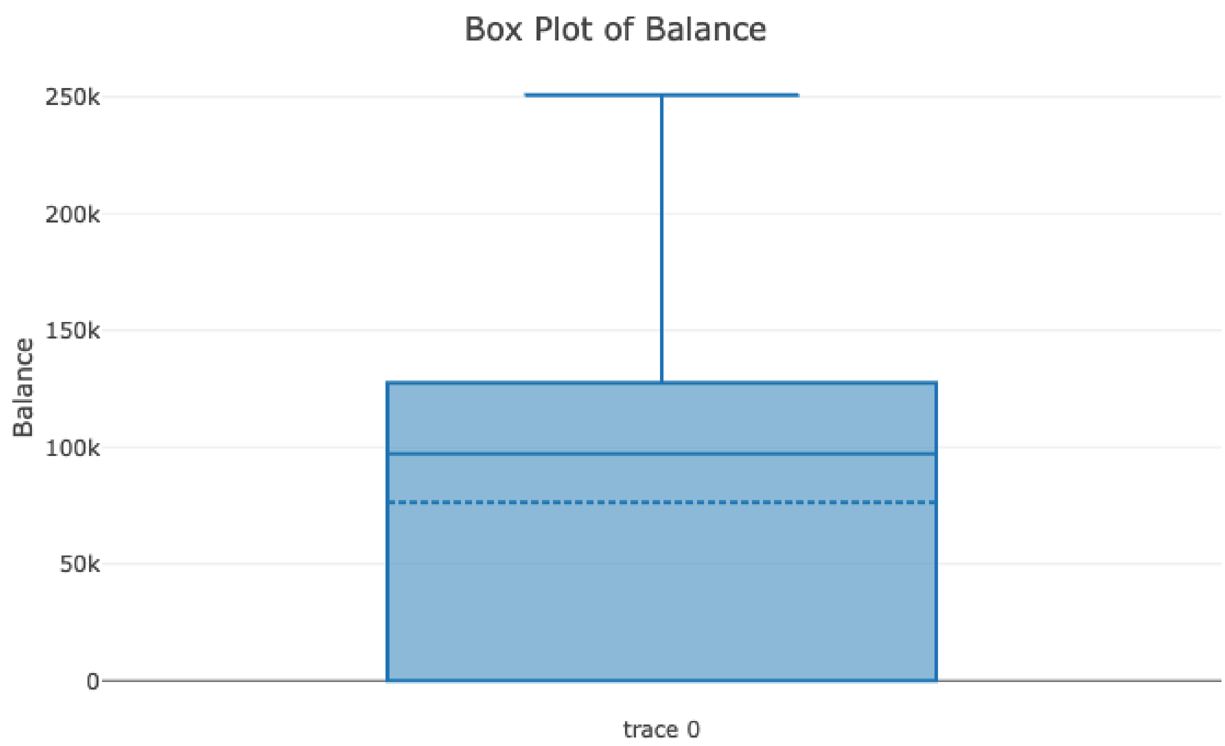
```
#check for outliers in numerical data

#check outliers for Age,balance, credit score,estimated salary,tenture

numerical_fr <- c("CreditScore",
                  "Age",
                  "Balance",
                  "EstimatedSalary",
                  "Tenure",
                  "NumOfProducts")




outlier_ck_nr <- sapply(numerical_fr,function(fr){
  plot_ly(
    data = bank_churn,
    y = ~ bank_churn[[fr]],
    type = 'box',
    boxmean = TRUE
  ) %>% layout(title = paste("Box Plot of", fr),
               yaxis = list(title = fr))},simplify = FALSE)

  outlier_ck_nr

## $CreditScore
##
## $Age
##
## $Balance
##
## $EstimatedSalary
##
## $Tenure
##
## $NumOfProducts
```

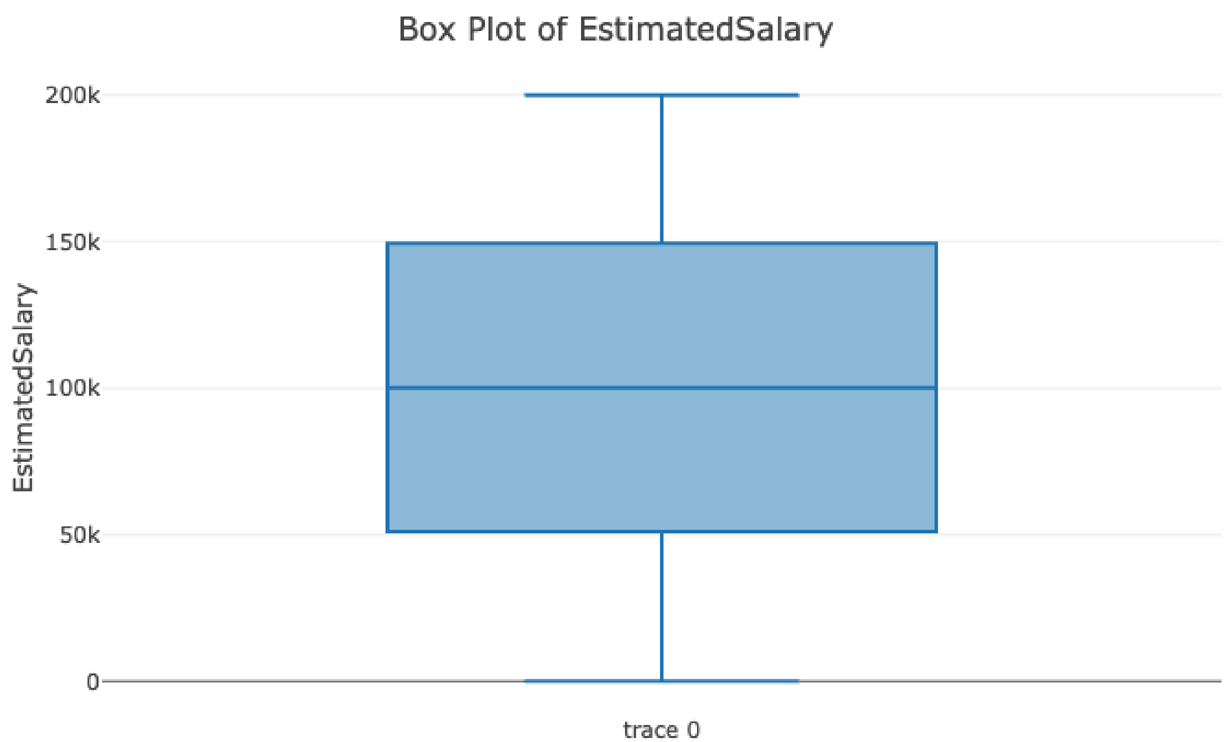


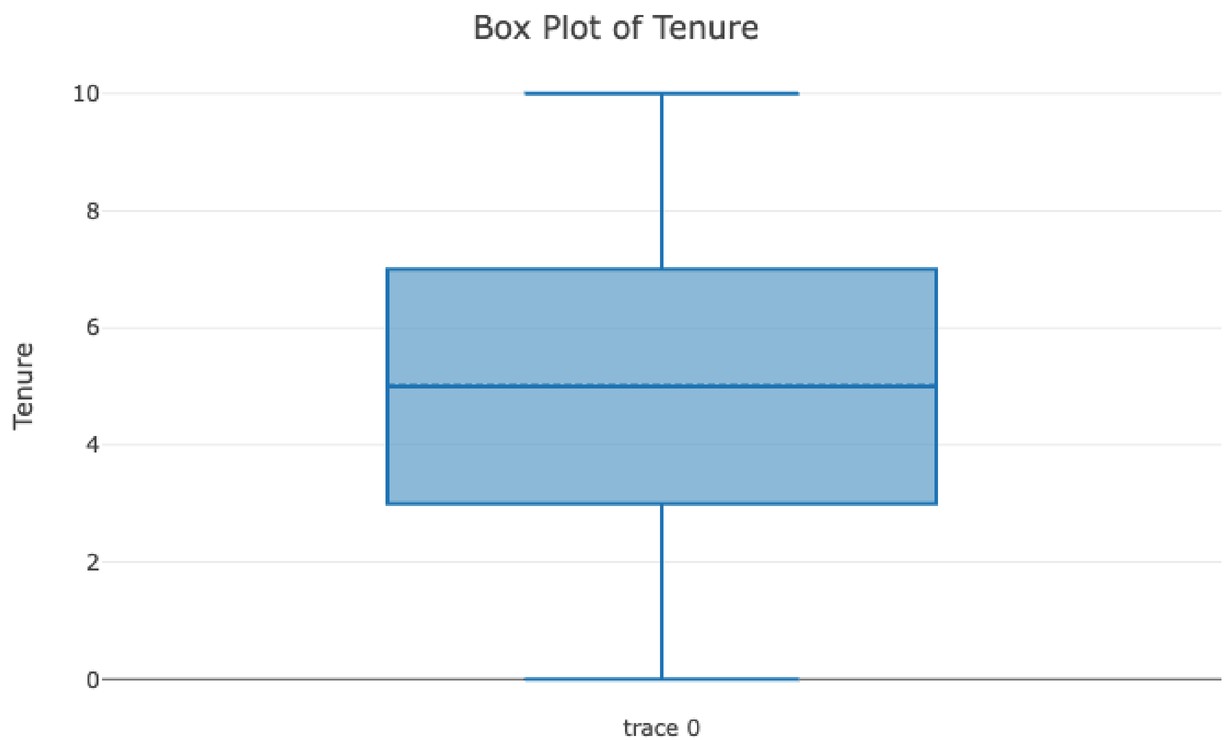
Box Plot of CreditScore
800
700
600
500
400
CreditScore
trace 0

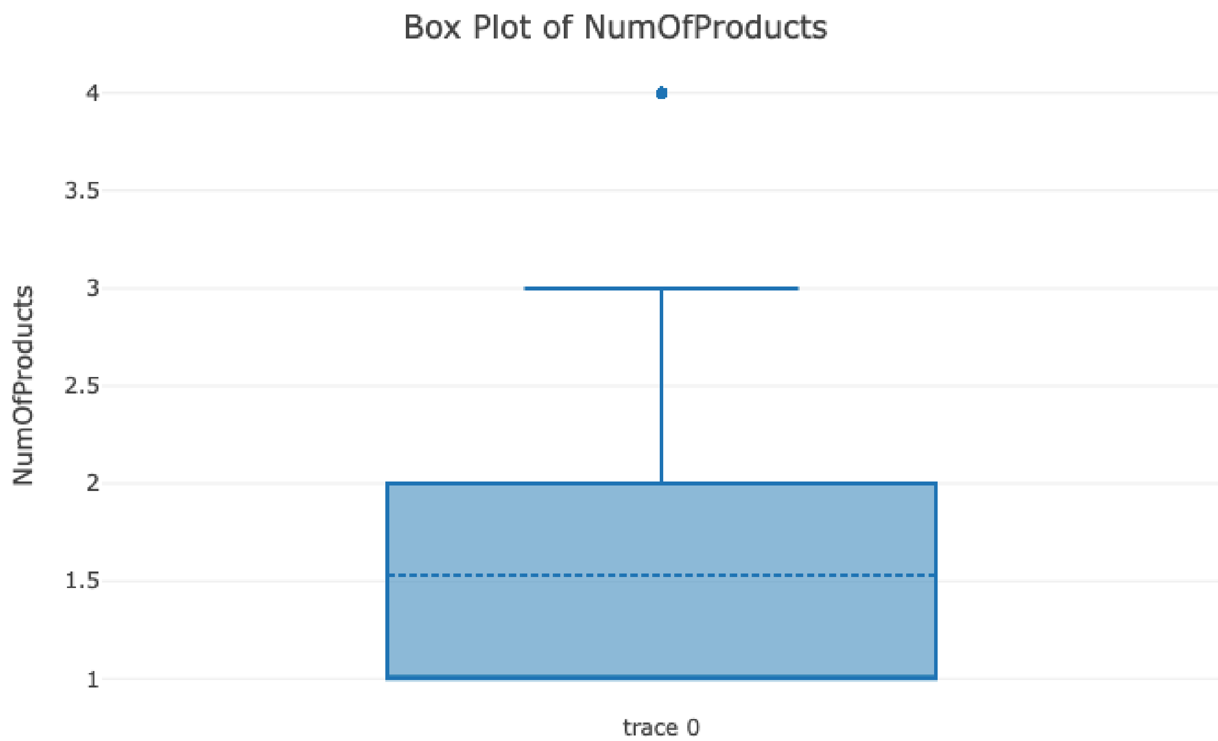


Box Plot of Age
Age
90
80
70
60
50
40
30
20
trace 0


Box Plot of Balance
250k
200k
150k
100k
50k
0
Balance
trace 0


Box Plot of EstimatedSalary
200k
150k
100k
50k
0
EstimatedSalary
trace 0


Box Plot of Tenure
10
8
6
4
2
0
Tenure
trace 0



Box Plot of NumOfProducts
NumOfProducts
4
3.5
3
2.5
2
1.5
1
trace 0

2.1.9 Define Methods to find Outliers & Find the Data Percentage to be Removed

```
#As the above depicts, Age, NumOfProducts, and CreditScore have outliers.
Next, we need to check if removing those values is safe by finding the number
of rows that get removed from the dataset( if it is 5% or less it is safe &
no data loss)

#calculate IQR value



find_lower_upper_bound <- function(dataset, column) {
  Q1 <- quantile(dataset[[column]], 0.25, na.rm = TRUE)
  Q3 <- quantile(dataset[[column]], 0.75, na.rm = TRUE)
  IQR_value <- Q3 - Q1

  #find the lower bound and the upper bound
  lower_bound <- Q1 - 1.5 * IQR_value
  upper_bound <- Q3 + 1.5 * IQR_value
  return(list(lower_bound = lower_bound, upper_bound = upper_bound))
}

# Function to find outliers using IQR
find_outliers <- function(dataset, column) {
  bound_data = find_lower_upper_bound(dataset, column)
  # Count number of outliers
  sum(dataset[[column]] < bound_data$lower_bound |
        dataset[[column]] > bound_data$upper_bound,
      na.rm = TRUE)
}

#check outlier count for each feature
outlier_ct <- sapply(c("Age", "CreditScore", "NumOfProducts"), function(col)
  find_outliers(bank_churn, col))
print(outlier_ct)

##           Age   CreditScore NumOfProducts
##           359            15            60

#check percentage of rows to be removed
total_rows <- nrow(bank_churn)
percentage_removed <- sum(outlier_ct) / total_rows * 100
```

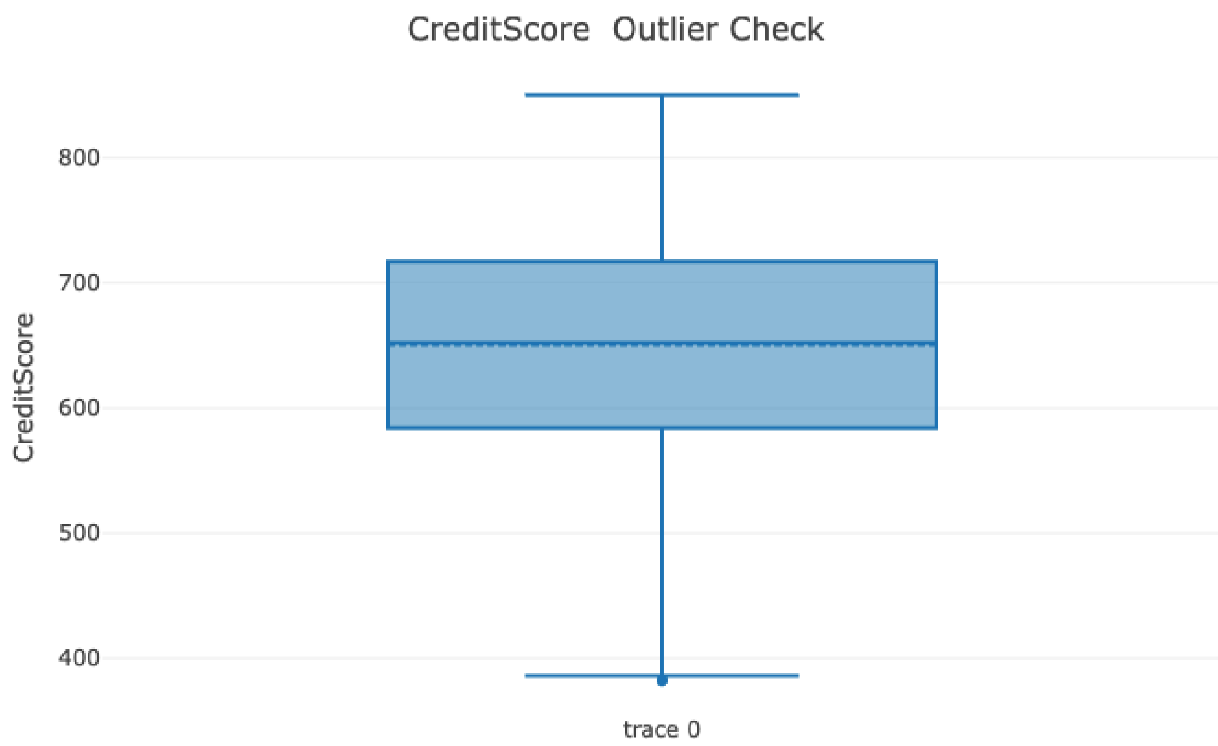
```r
print(paste(
  "Percentage of data to be removed:",
  round(percentage_removed, 2),
  "%"
))
```

```
## [1] "Percentage of data to be removed: 4.34 %"
```

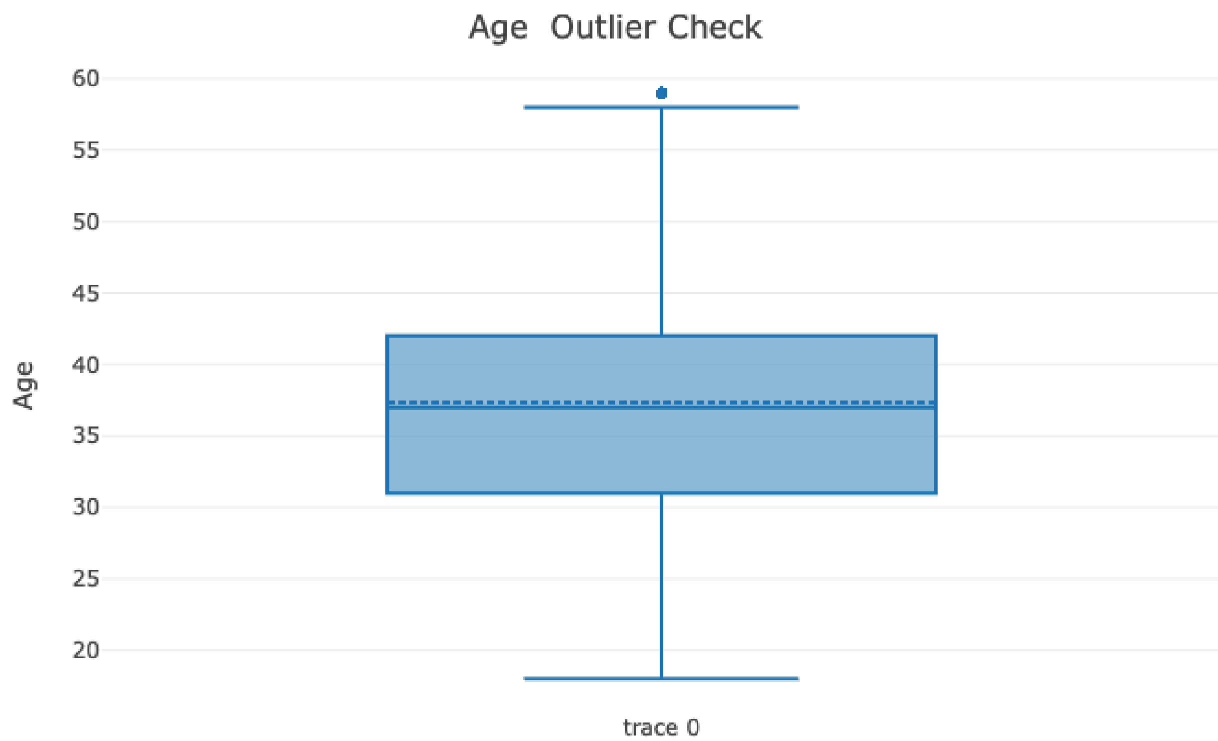
2.1.10 Define Method to Remove Outliers and View Features on Box plot after Outlier Removal

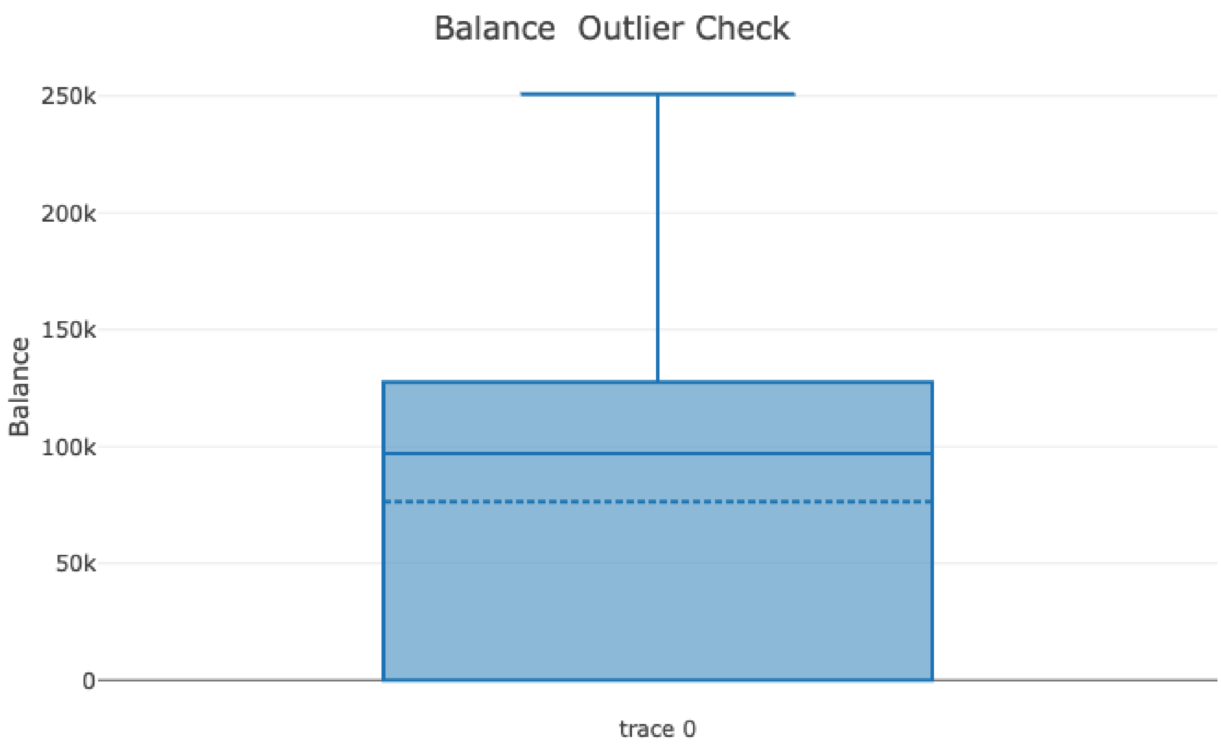
```
#Since the above result depicts that the data percentage that gets removed by
removing outliers is less than 5%( 4.34%), in the code below, the outlier of
those features is removed.

remove_outliers <- function(dataset, col){
  bound_values <- find_lower_upper_bound(dataset,col)
  print(paste(col,"Lower --->",bound_values$lower_bound,", Upper --->"
,bound_values$upper_bound))
   filtered_data <- dataset[dataset[[col]] >= bound_values$lower_bound &
dataset[[col]] <= bound_values$upper_bound, ]
  return(filtered_data)
}


 #remove outliers of data repetively until all outlier get removed
 ag_outlier_removed_data <- remove_outliers(bank_churn, "Age")

## [1] "Age Lower ---> 14 , Upper ---> 62"

 cr_outlier_removed_data<- remove_outliers(ag_outlier_removed_data,
"CreditScore")

## [1] "CreditScore Lower ---> 382 , Upper ---> 918"

 nm_outlier_removed_data <- remove_outliers(cr_outlier_removed_data,
"NumOfProducts")

## [1] "NumOfProducts Lower ---> -0.5 , Upper ---> 3.5"

 ag2outlier_removed_data <- remove_outliers(nm_outlier_removed_data, "Age")

## [1] "Age Lower ---> 15.5 , Upper ---> 59.5"

 ag3outlier_removed_data <- remove_outliers(ag2outlier_removed_data, "Age")

## [1] "Age Lower ---> 14.5 , Upper ---> 58.5"

 cr2_outlier_removed_data<- remove_outliers(ag3outlier_removed_data,
"CreditScore")

## [1] "CreditScore Lower ---> 384.5 , Upper ---> 916.5"
```

```r
 df_outlier_removed_data <- as.data.frame(ag2outlier_removed_data) # Convert
matrix to dataframe

#check all outliers got removed
for (fr in  numerical_fr){

  outlierremoved_plot <- plot_ly(
    data = df_outlier_removed_data,
    y = ~ df_outlier_removed_data[[fr]],
    type = 'box',
    boxmean = TRUE
  ) %>% layout(title = paste(fr," Outlier Check"),
               yaxis = list(title = fr))

  #view outlier removed  plot
  print(outlierremoved_plot)

  }

#check outlier count for each feature
outlier_ct01 <- sapply(c("Age", "CreditScore","NumOfProducts"), function(col)
find_outliers(cr2_outlier_removed_data, col))
print(outlier_ct01)

##           Age   CreditScore NumOfProducts
##             0             0             0
```

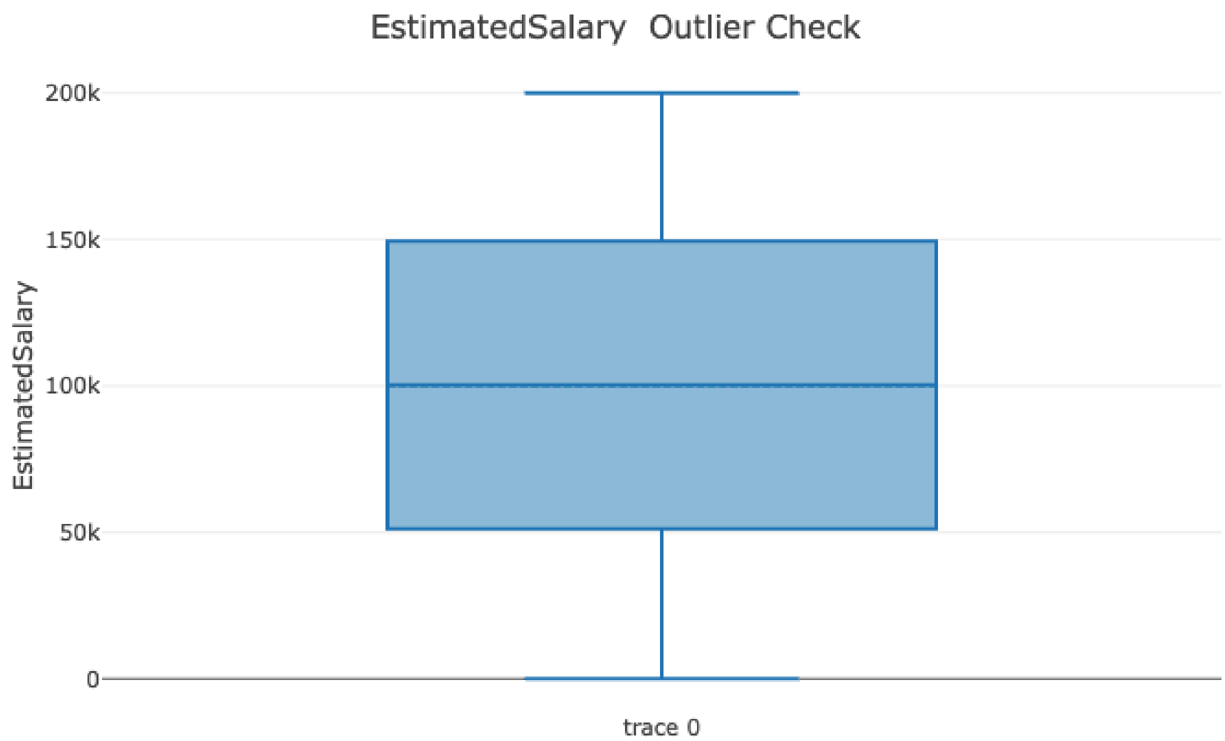


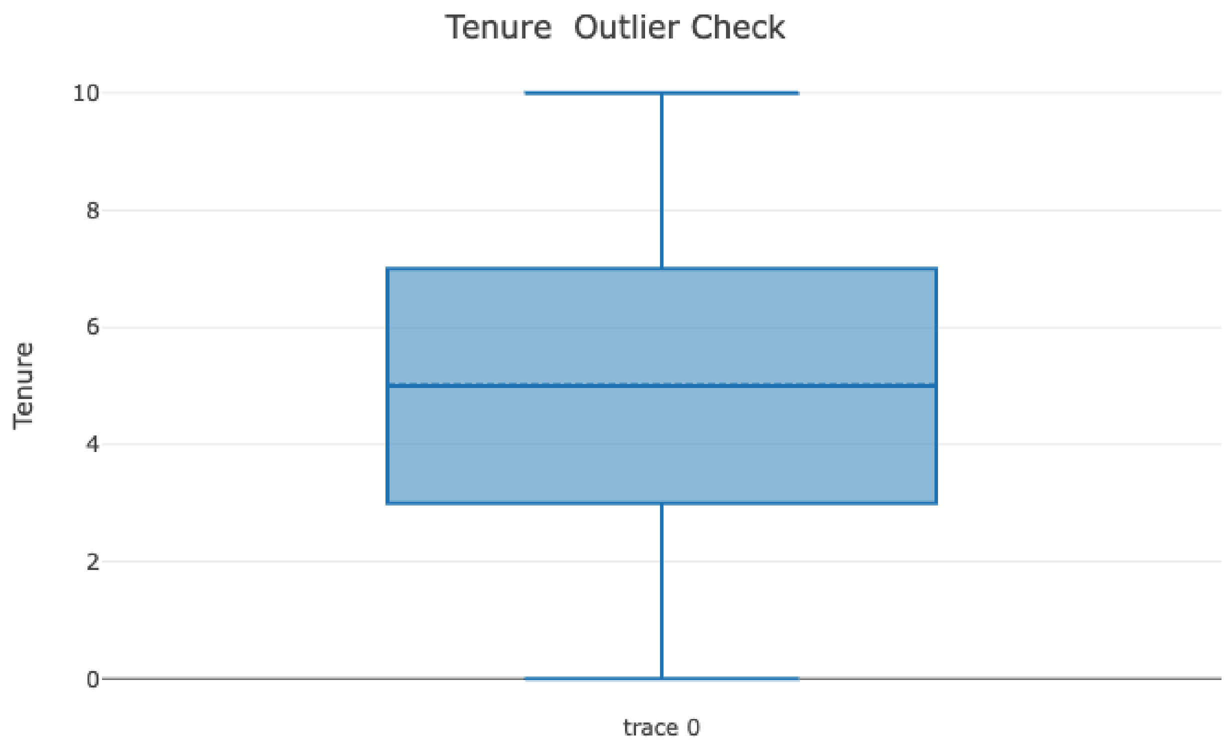
CreditScore Outlier Check
800
700
600
500
400
CreditScore
trace 0

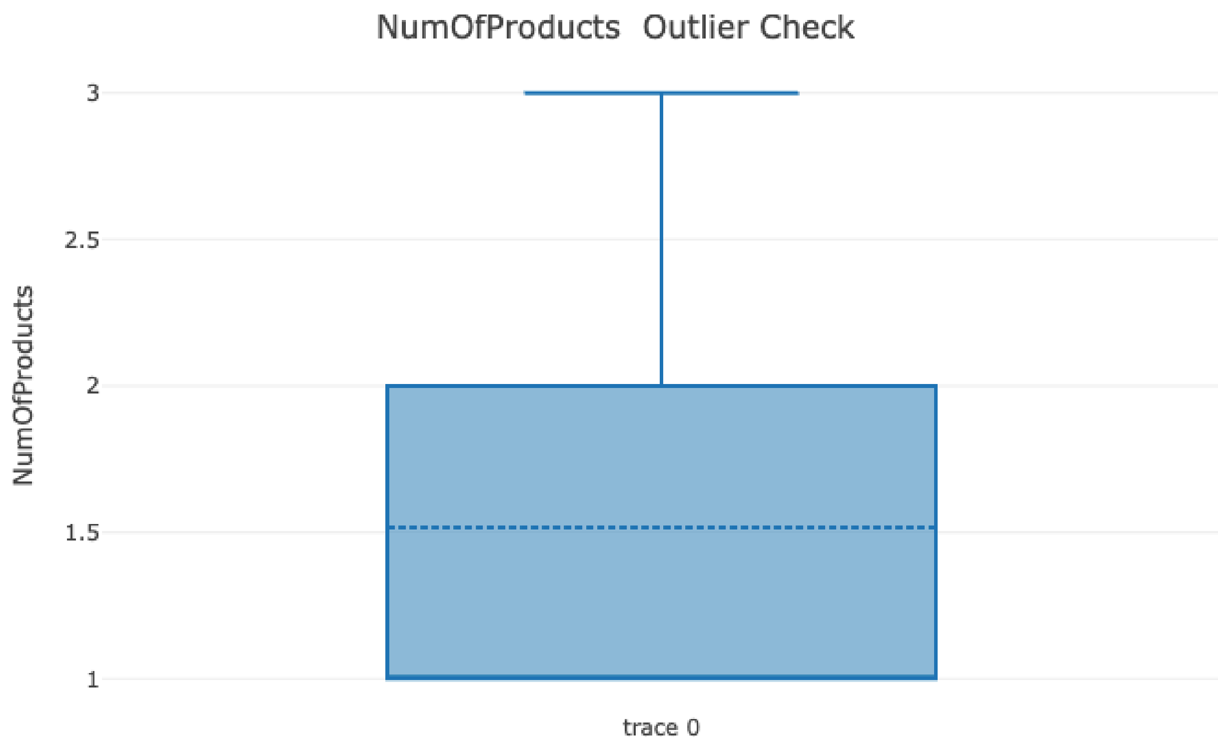

Age Outlier Check
60
55
50
45
40
35
30
25
20
Age
trace 0



Balance Outlier Check
250k
200k
150k
100k
50k
0
Balance
trace 0


EstimatedSalary Outlier Check
EstimatedSalary
200k
150k
100k
50k
0
trace 0


Tenure Outlier Check
Tenure
10
8
6
4
2
0
trace 0


NumOfProducts Outlier Check
3
2.5
2
1.5
1
NumOfProducts
trace 0

2.1.11 Convert Categorical Variables into Factors and Numerical data

```
#next need convert categorical data in to factors values in order to use in
regression model.


# Convert categorical variables to factors
df_outlier_removed_data$Geography <-
as.factor(df_outlier_removed_data$Geography)
df_outlier_removed_data$Gender <- as.factor(df_outlier_removed_data$Gender)
df_outlier_removed_data$Exited <-
factor(df_outlier_removed_data$Exited,levels = c(0, 1))
df_outlier_removed_data$HasCrCard <-
factor(df_outlier_removed_data$HasCrCard)
df_outlier_removed_data$IsActiveMember <-
factor(df_outlier_removed_data$IsActiveMember)

#since above is not working with smote and both traning data testing data
should have same format for each column. Next need to  convert categorcal
data to numerical vectors.

df_outlier_removed_data$Gender <-
factor(df_outlier_removed_data$Gender,labels = c(0, 1))

df_outlier_removed_data$Geography  <-
factor(df_outlier_removed_data$Geography)


# View structure after conversion
str(df_outlier_removed_data)

## 'data.frame':    9406 obs. of  11 variables:
##  $ Geography      : Factor w/ 3 levels "France","Germany",..: 1 3 1 1 3 3
1 1 1 1 ...
##  $ CreditScore    : int  619 608 502 699 850 645 822 501 684 528 ...
##  $ Gender         : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 2 2 2 ...
##  $ Age            : int  42 41 42 39 43 44 50 44 27 31 ...
##  $ Tenure         : int  2 1 8 1 2 8 7 4 2 6 ...
##  $ Balance        : num  0 83808 159661 0 125511 ...
##  $ NumOfProducts  : int  1 1 3 2 1 2 2 2 1 2 ...
```

```
##  $ HasCrCard      : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 2 1 2 1 ...
##  $ IsActiveMember : Factor w/ 2 levels "0","1": 2 2 1 1 2 1 2 2 2 1 ...
##  $ EstimatedSalary: num  101349 112543 113932 93827 79084 ...
##  $ Exited         : Factor w/ 2 levels "0","1": 2 1 2 1 1 2 1 1 1 1 ...
```

2.1.12 Split Data into Test and Train data

```
set.seed(123)

#splitting the data set considering target variable since we need a balance
exited = 1 and exited = 0 amout of data in both test and traning datasets.
train_index <- createDataPartition(df_outlier_removed_data$Exited,
                                   p = 0.8,
                                   list = FALSE)

#subset the data
train_data <- df_outlier_removed_data[train_index, ]  # Training set (80%)
test_data <- df_outlier_removed_data[-train_index, ]  # Testing set (20%)

#check dimensions
dim(train_data)

## [1] 7526   11

dim(test_data)

## [1] 1880   11
```

2.1.13 Split Further into X and Y data

```r
# seperate x and y data after data get spliited.

x_train_data <- train_data[, c(
  "Geography",
  "CreditScore",
  "Gender",
  "Age",
  "Tenure",
  "Balance",
  "NumOfProducts",
  "HasCrCard",
  "IsActiveMember",
  "EstimatedSalary"
)]

y_train_data <- train_data[, c("Exited")]

x_test_data <- test_data[, c(
  "Geography",
  "CreditScore",
  "Gender",
  "Age",
  "Tenure",
  "Balance",
  "NumOfProducts",
  "HasCrCard",
  "IsActiveMember",
  "EstimatedSalary"
)]

y_test_data <- test_data[, c("Exited")]
```

2.1.14 Feature Selection Using Correlation

Feature selection is need to be done after training and test data get split if test data also participated in feature selection accuracy of model will be get higher since model can learn indirectly from unseen data which may lead to overfitting.

```
#select important feature to traning the model using correlation

important_features <- names(which(abs(corelation_m["Exited", ]) > 0.1))

print(important_features)

## [1] "Age"            "Balance"        "IsActiveMember" "Exited"
```

2.1.15 Feature Selection Using Recursive Feature Elimination

But since we have both categorical and numerical features using correlation might mislead. because of above reason it is best to use “Recursive Feature Elimination”

```
  #define RFE control using cross-validation
  ctrl <- rfeControl(functions = rfFuncs,
                     method = "cv",
                     number = 5)

  #run RFE on training data
  rfe_result <- rfe(x_train_data,
                    #exclude target variable
                    y_train_data,
                    #target variable
                    sizes = c(1:5),
                    #number of features to select (1 to 5)
```

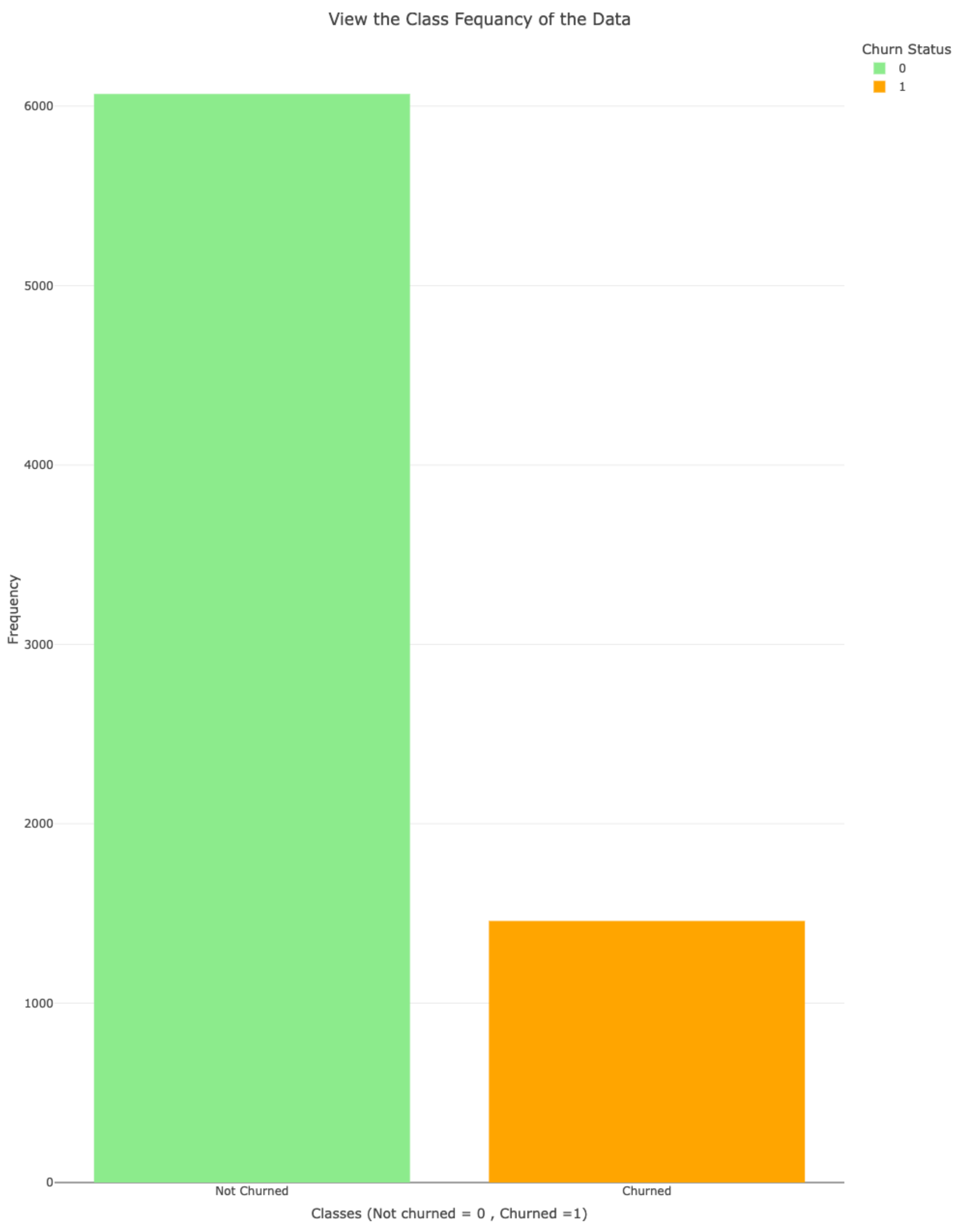
```
                    rfeControl = ctrl)

  #print the selected features
  print(rfe_result)

##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (5 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.8150 0.1906   0.010173 0.03275
##          2   0.8465 0.4350   0.010701 0.03638
##          3   0.8453 0.4137   0.008989 0.02894
##          4   0.8526 0.4388   0.008277 0.03367
##          5   0.8631 0.4743   0.011312 0.04619        *
##         10   0.8585 0.4617   0.008051 0.02849
##
## The top 5 variables (out of 5):
##    Age, NumOfProducts, Balance, Geography, IsActiveMember
```

2.1.16 View If Target Feature Data is Imbalanced

View if target feature data is imbalanced to prevent from model getting bias towards the majority class.

```
#view summary of data
summary(y_train_data)

##    0    1
## 6068 1458

#view if the target class data is imbalanced

ch_fr <- plot_ly(
  as.data.frame(table(y_train_data)),
  x = ~ y_train_data,
  y = ~ Freq,
  type = "bar",
  color = ~ factor(y_train_data),
  colors = c("1" = "orange", "0" = "lightgreen")
```

```
) %>%
  layout(
    title = "View the Class Fequancy of the Data",
    xaxis = list(title = "Classes (Not churned = 0 , Churned =1)",tickvals =
c(0, 1),
      ticktext = c("Not Churned", "Churned")),
    yaxis = list(title = "Frequency"),
    legend = list(title = list(text = 'Churn Status'))
  )

#view the plot
ch_fr

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca267673ee3/widgetca22b51de1e.html screenshot completed
```

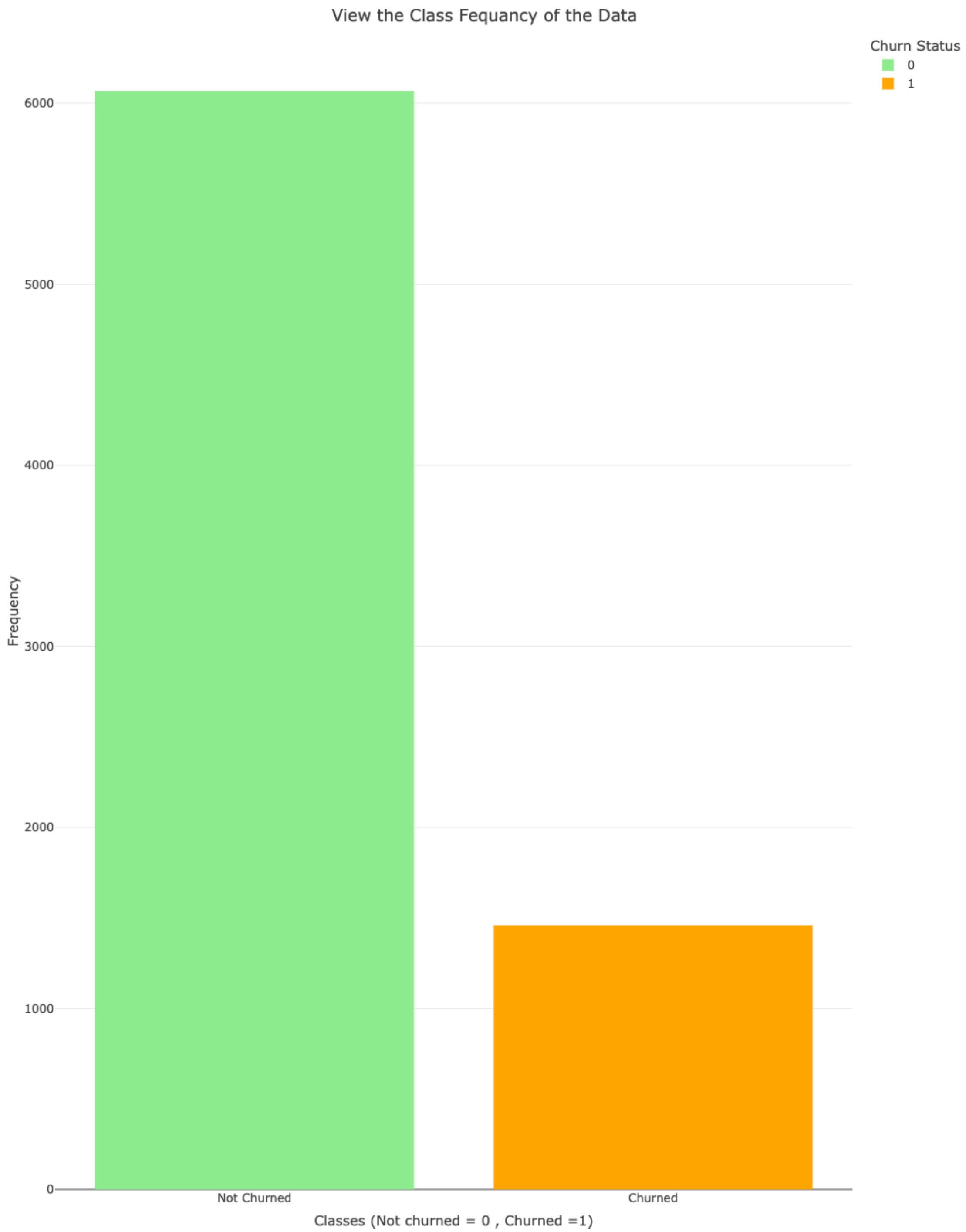


Since the above bar plot depict data of two classes in churn status is imbalanced. This need to be handled before training the model. Since the data set is not much large, SMOTE (Synthetic Minority Over-sampling Technique) will be used in next steps to balance the target data. This will improve the model and avoid model prediction bias towards the majority class.

2.1.17 Use SMOTE to Generate More Data Points For minority Class

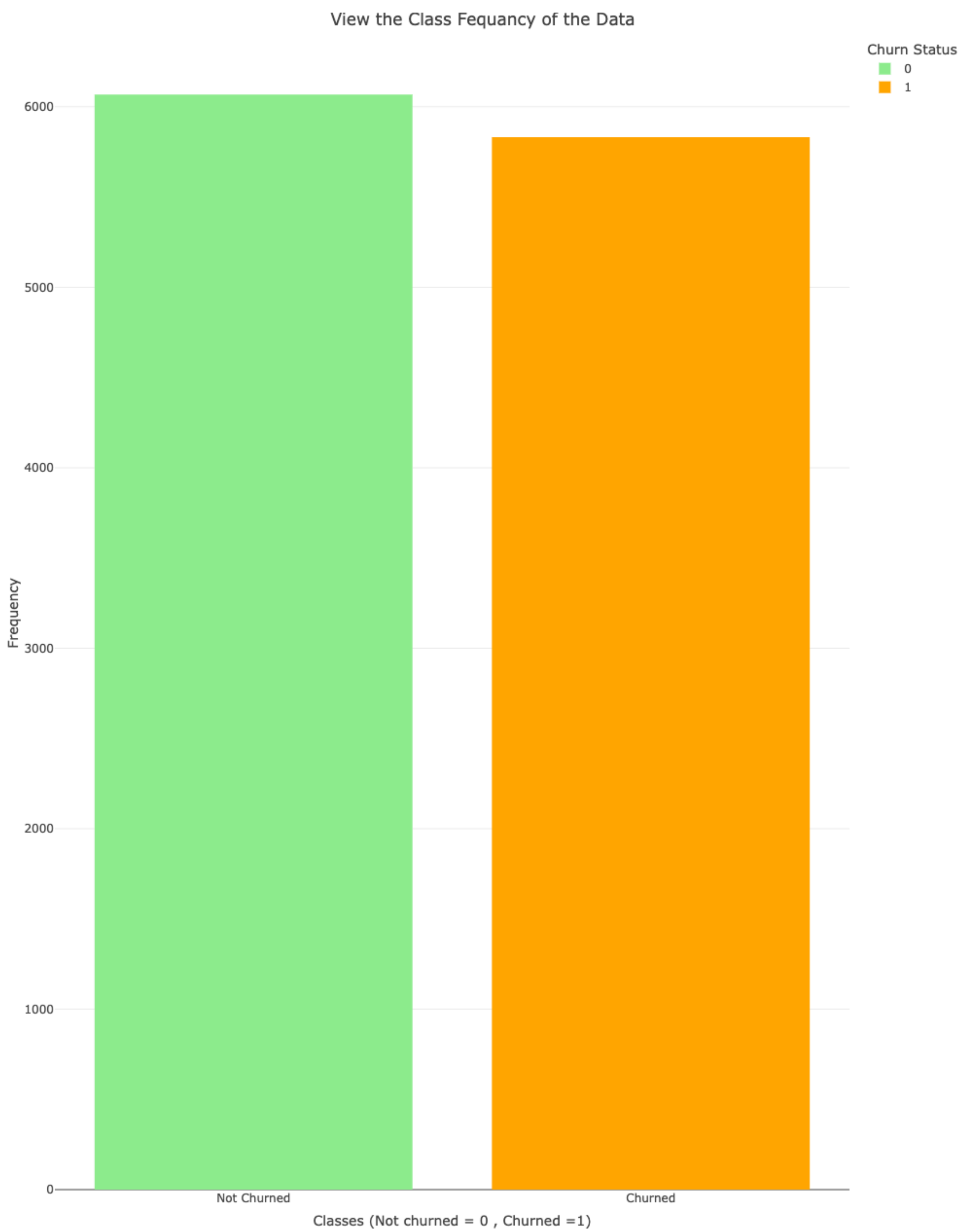
```
#apply one-hot encoding (convert factors to dummy variables)
predictor_vars <- x_train_data %>%
  mutate(across(where(is.factor), as.numeric))

#check the structure
str(predictor_vars)

## 'data.frame':    7526 obs. of  10 variables:
##  $ Geography      : num  3 1 1 3 3 1 1 1 3 1 ...
##  $ CreditScore    : int  608 502 699 850 645 501 684 528 497 476 ...
##  $ Gender         : num  1 1 1 1 2 2 2 2 2 1 ...
##  $ Age            : int  41 42 39 43 44 44 27 31 24 34 ...
##  $ Tenure         : int  1 8 1 2 8 4 2 6 3 10 ...
##  $ Balance        : num  83808 159661 0 125511 113756 ...
##  $ NumOfProducts  : int  1 3 2 1 2 2 1 2 2 2 ...
##  $ HasCrCard      : num  1 2 1 2 2 1 2 1 2 2 ...
##  $ IsActiveMember : num  2 1 1 2 1 2 2 1 1 1 ...
##  $ EstimatedSalary: num  112543 113932 93827 79084 149757 ...

#apply SMOTE
smote_result <- SMOTE(
  X = predictor_vars,
  target = y_train_data,
  K = 2,
  #number of nearest neighbors
  dup_size = 3
)          #oversampling rate

#check class distribution after SMOTE
table(smote_result$data$class)

##
##    0    1
## 6068 5832
```

```r
#check new class distribution

class_counts <- table(smote_result$data$class)
as.data.frame(class_counts)

##   Var1 Freq
## 1    0 6068
## 2    1 5832

x_train_smote <- smote_result$data[, !names(smote_result$data) %in% "class"]
y_train_smote <- smote_result$data$class



#view smote genrated data on barplot
ch_fr_after_smt <- plot_ly(
  as.data.frame(class_counts),
  x = ~ Var1,
  y = ~ Freq,
  type = "bar",
  color = ~ factor(Var1),
  colors = c("1" = "orange", "0" = "lightgreen")
) %>%
  layout(
    title = "View the Class Fequancy of the Data",
    xaxis = list(
      title = "Classes (Not churned = 0 , Churned =1)",
      tickvals = c(0, 1),
      ticktext = c("Not Churned", "Churned")
    ),
    yaxis = list(title = "Frequency"),
    legend = list(title = list(text = 'Churn Status'))
  )

#view the plot
ch_fr

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca2182c8eae/widgetca2154b4de8.html screenshot completed
```




```
ch_fr_after_smt
```

```
## file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/fileca23e47f4d9/widgetca2661bf280.html screenshot completed
```


View the Class Fequancy of the Data
Churn Status
0
1
6000
5000
4000
3000
2000
1000
0
Frequency
Not Churned
Churned
Classes (Not churned = 0 , Churned =1)

Task 2.2

2.2.1 Train Logistic Regression Model and Random Forest Model to Predict Churn Status

While using glm with the smote, It took a lot of time to produce the results used weights in glm to give more weight to 1 class(Exited).

```
#use selected features from RFE
selected_features_rf <- rfe_result$optVariables

model_data_glm = data.frame(y_train_data,x_train_data)



log_model_rw <- glm(y_train_data~ ., data = model_data_glm, family =
binomial,weights = ifelse(model_data_glm$y_train_data == 1, 2, 1))

#train Logistic Regression with selected features
log_model <- glm(y_train_data~ ., data = model_data_glm, family =
binomial,weights = ifelse(model_data_glm$y_train_data == 1, 2, 1))

stepwise_log_model <- step(log_model, direction = "both", trace = 0)



#train Random Forest with selected features
rf_model <- randomForest(
  y = y_train_data,
  x = x_train_data[,c(selected_features_rf)],
  ntree = 500,
  mtry = 2,
  sampsize = c(1000, 1000)
)
```

Task 2.3

2.3.1 Make Predictions using Above Trained Models

```
#make Predictions on Test Data
log_predictions <- predict(stepwise_log_model, x_test_data, type =
"response")
log_pred_class <- ifelse(log_predictions > 0.5, 1, 0)
log_pred_class <- as.factor(log_pred_class)

log_pred_rw <- predict(log_model_rw, x_test_data, type = "response")
log_pred_class_rw <- ifelse(log_pred_rw > 0.5, 1, 0)
log_pred_class_rw  <- as.factor(log_pred_class_rw)

rf_predictions<- predict(rf_model, x_test_data[, c(selected_features_rf)],
type = "prob")[, 2]
rf_pred_class <- ifelse(rf_predictions > 0.5, 1, 0)
rf_pred_class <- factor(rf_pred_class)

#evaluate Performance
log_cm <- confusionMatrix(log_pred_class, y_test_data, positive = "1")
log_cm_rw <- confusionMatrix(log_pred_class_rw, y_test_data, positive = "1")
rf_cm <- confusionMatrix(rf_pred_class, y_test_data, positive = "1")



print("Logistic Regression performance without step wise and feature
selection:")

## [1] "Logistic Regression performance without step wise and feature
selection:"

print(log_cm_rw)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1345  195
##          1  171  169
##
##                Accuracy : 0.8053
##                  95% CI : (0.7867, 0.823)
##     No Information Rate : 0.8064
##     P-Value [Acc > NIR] : 0.5603
##
##                   Kappa : 0.3605
##
##  Mcnemar's Test P-Value : 0.2293
##
##             Sensitivity : 0.46429
##             Specificity : 0.88720
##          Pos Pred Value : 0.49706
##          Neg Pred Value : 0.87338
##              Prevalence : 0.19362
##          Detection Rate : 0.08989
##    Detection Prevalence : 0.18085
##       Balanced Accuracy : 0.67574
##
##        'Positive' Class : 1
##
```

```
print("Logistic Regression performance with step wise and feature
selection:")
```

```
## [1] "Logistic Regression performance with step wise and feature
selection:"
```

```
print(log_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1346  192
##          1  170  172
##
##                Accuracy : 0.8074
```

```
##                  95% CI : (0.7889, 0.825)
##     No Information Rate : 0.8064
##     P-Value [Acc > NIR] : 0.4675
##
##                   Kappa : 0.3689
##
##  Mcnemar's Test P-Value : 0.2697
##
##             Sensitivity : 0.47253
##             Specificity : 0.88786
##          Pos Pred Value : 0.50292
##          Neg Pred Value : 0.87516
##              Prevalence : 0.19362
##          Detection Rate : 0.09149
##    Detection Prevalence : 0.18191
##       Balanced Accuracy : 0.68020
##
##        'Positive' Class : 1
##
```

```r
print("Random Forest Performance:")
```

```
## [1] "Random Forest Performance:"
```

```r
print(rf_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1178   90
##          1  338  274
##
##                Accuracy : 0.7723
##                  95% CI : (0.7527, 0.7911)
##     No Information Rate : 0.8064
##     P-Value [Acc > NIR] : 0.9999
##
##                   Kappa : 0.4208
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7527
##             Specificity : 0.7770
```

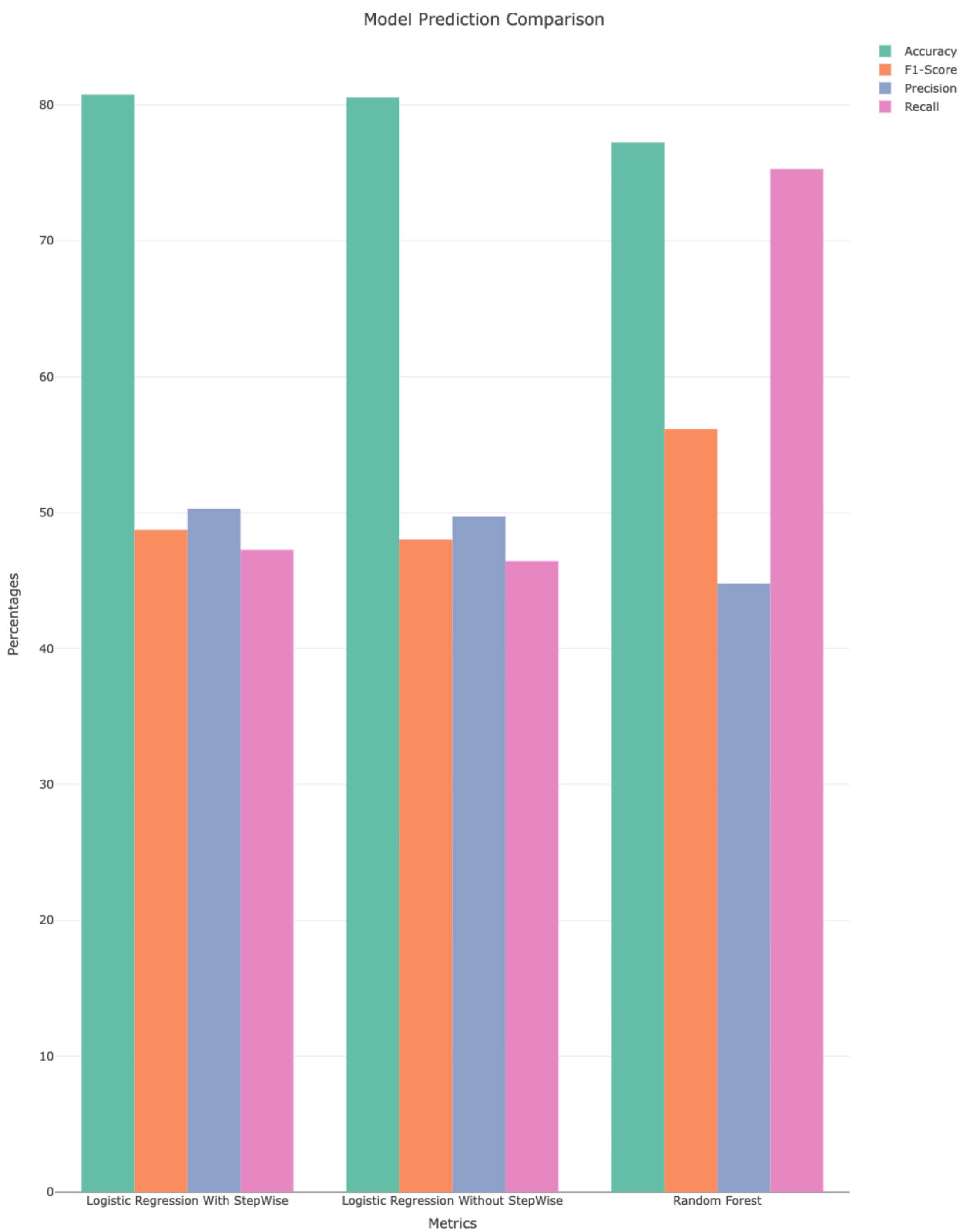
```
##          Pos Pred Value : 0.4477
##          Neg Pred Value : 0.9290
##              Prevalence : 0.1936
##          Detection Rate : 0.1457
##    Detection Prevalence : 0.3255
##       Balanced Accuracy : 0.7649
##
##        'Positive' Class : 1
##

#get Recall, Precision, and F1-Score from Confusion Matrix
accuracy <- log_cm$overall["Accuracy"]
log_recall <- log_cm$byClass["Sensitivity"]
log_precision <- log_cm$byClass["Precision"]
log_f1 <- log_cm$byClass["F1"]


accuracy_rw <- log_cm_rw$overall["Accuracy"]
log_recall_rw <- log_cm_rw$byClass["Sensitivity"]
log_precision_rw <- log_cm_rw$byClass["Precision"]
log_f1_rw <- log_cm_rw$byClass["F1"]

rf_acccuracy <- rf_cm$overall["Accuracy"]
rf_recall <- rf_cm$byClass["Sensitivity"]
rf_precision <- rf_cm$byClass["Precision"]
rf_f1 <- rf_cm$byClass["F1"]

#print Values
cat(
  "Logistic Regression with step wise - Recall:",
  log_recall,
  "Precision:",
  log_precision,
  "F1-Score:",
  log_f1,
  "\n"
)

## Logistic Regression with step wise - Recall: 0.4725275 Precision: 0.502924
F1-Score: 0.4872521

cat(
  "Logistic Regression without step wise - Recall:",
  log_recall_rw,
```

```r
  "Precision:",
  log_precision_rw,
  "F1-Score:",
  log_f1_rw,
  "\n"
)

## Logistic Regression without step wise - Recall: 0.4642857 Precision:
0.4970588 F1-Score: 0.4801136

cat(
  "Random Forest - Recall:",
  rf_recall,
  "Precision:",
  rf_precision,
  "F1-Score:",
  rf_f1,
  "\n"
)

## Random Forest - Recall: 0.7527473 Precision: 0.4477124 F1-Score: 0.5614754
```

2.3.2 View Predicted Data in Plot

```r
model_redictions <- data.frame(
  Model = rep(c('Logistic Regression With StepWise','Logistic Regression
Without StepWise','Random Forest'), each = 4),
  Metric = rep(c(
    "Accuracy", "Recall", "Precision", "F1-Score"
  ), times = 3),
  Prediction = c(
    accuracy,
    log_recall,
    log_precision,
    log_f1,
    accuracy_rw,
    log_recall_rw,
    log_precision_rw,
    log_f1_rw,
    rf_acccuracy,
    rf_recall,
    rf_precision,
```

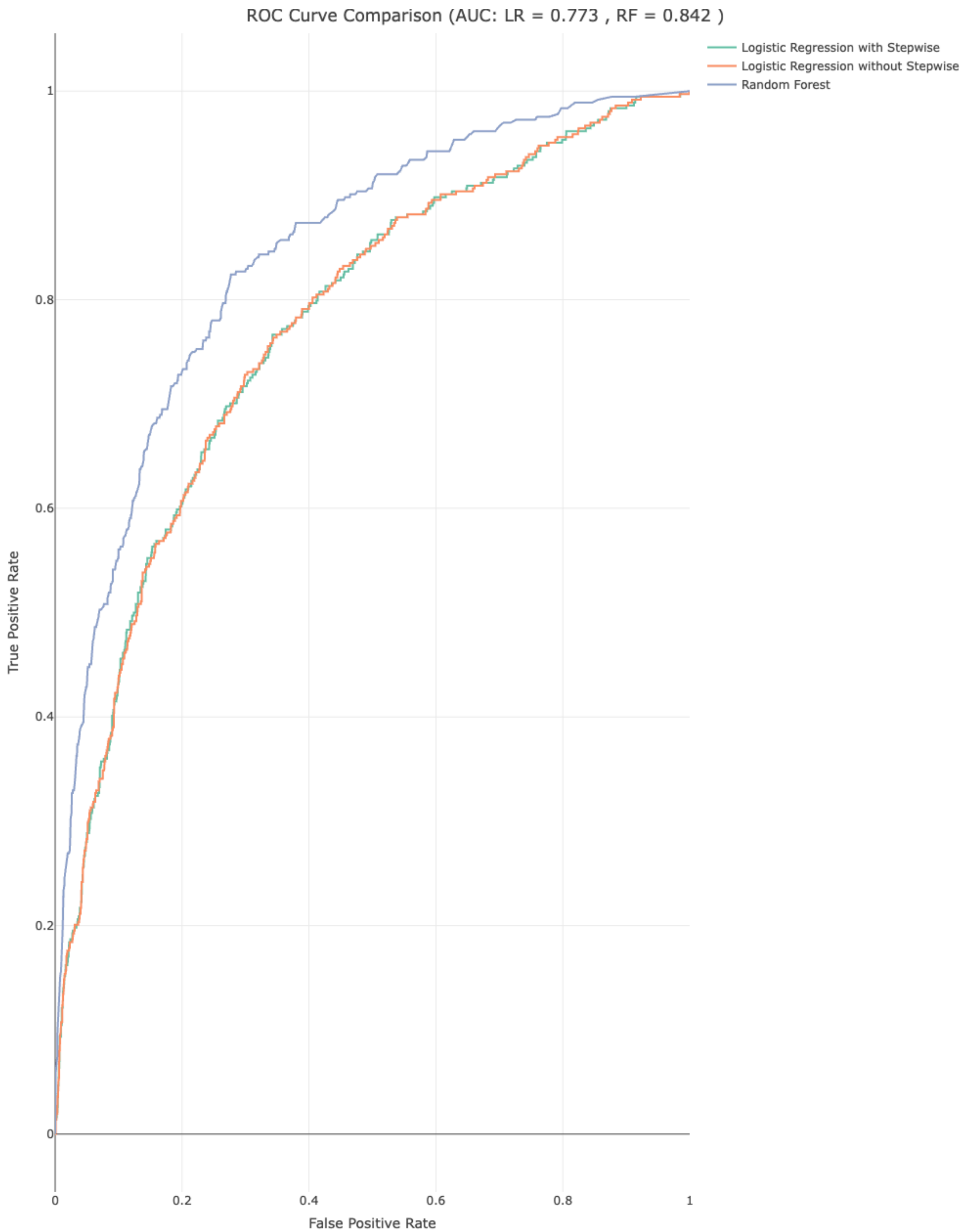
```
    rf_f1
  ) * 100
)

model_metrics_prediction <- plot_ly(
  model_redictions,
  x = ~ Model,
  y = ~ Prediction,
  type = 'bar',
  color = ~ Metric
) %>%
  layout(
    title = "Model Prediction Comparison",
    xaxis = list(title = "Metrics"),
    yaxis = list(title = "Percentages"),
    barmode = 'group',
    showlegend = TRUE
  )

#view the plot
model_metrics_prediction

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca24fb2bd30/widgetca25fa2c330.html screenshot completed
```


Model Prediction Comparison
Accuracy
F1-Score
Precision
Recall
Percentages
0
10
20
30
40
50
60
70
80
Logistic Regression With StepWise
Logistic Regression Without StepWise
Random Forest
Metrics

2.3.3 View ROC Curve

```
# Load pROC for ROC analysis
library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

# Compute ROC Curve
log_roc <- roc(test_data$Exited, log_predictions)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

log_roc_rw <- roc(test_data$Exited, log_pred_rw)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

rf_roc <- roc(test_data$Exited, rf_predictions)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

# Compute AUC
log_auc <- auc(log_roc)
log_auc_rw <- auc(log_roc_rw)
rf_auc <- auc(rf_roc)


roc_data <- data.frame(
  Model = rep(c('Logistic Regression with Stepwise','Logistic Regression
without Stepwise', 'Random Forest'),times =
c(length(log_roc$specificities),length(log_roc_rw$specificities),
length(rf_roc$specificities))),
  FPR = c(1 - log_roc$specificities,1 - log_roc_rw$specificities,1-
rf_roc$specificities),  # False Positive Rate
  TPR = c(log_roc$sensitivities
,log_roc_rw$sensitivities,rf_roc$sensitivities)       # True Positive Rate
```

```
)

plot_ly(roc_data, x = ~FPR, y = ~TPR, type = 'scatter', mode = 'lines',
color = ~Model
) %>%
  layout(
    title = paste("ROC Curve Comparison (AUC: LR =", round(log_auc, 3), ", RF
=", round(rf_auc, 3), ")"),
    xaxis = list(title = 'False Positive Rate'),
    yaxis = list(title = 'True Positive Rate')
  )

##
file:////private/var/folders/f7/y7dy19712nvdtgwj64c2bpl00000gn/T/Rtmp8976bi/f
ileca239520b7c/widgetca23970174a.html screenshot completed
```


ROC Curve Comparison (AUC: LR = 0.773 , RF = 0.842)
Logistic Regression with Stepwise
Logistic Regression without Stepwise
Random Forest
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1

Comparing the above result, we can understand that the random forest model is performing well with the data set. From statistical two models, the model used step wise to select the features has an increased accuracy, recall, f1-score precision. Since this is an imbalanced data set we need to compare all the metrics of the model to identify the best model. In the statistical models, the mode which uses stepwise has outperformed the model which did not use the stepwise method to select features since the all metrics of that model is higher that the other logistic regression model.

Task 2.4

2.4.1 Get the splitted data and separate x and y (label) data

```
set.seed(123)

#splitting the data set considering target variable since we need a balance
exited = 1 and exited = 0 amout of data in both test and traning datasets.
train_index_tn <- createDataPartition(df_outlier_removed_data$Tenure,
                                      p = 0.8,
                                      list = FALSE)

#subset the data
train_data_tn <- df_outlier_removed_data[train_index_tn, ]  # Training set
(80%)
test_data_tn <- df_outlier_removed_data[-train_index_tn, ]  # Testing set
(20%)

#check dimensions
dim(train_data_tn)

## [1] 7527   11

dim(test_data_tn)

## [1] 1879   11

#removed customerId and Surname since they not giving much important details
for training the model
x_train_data_tn <- train_data_tn[, c(
  "Geography",
  "CreditScore",
  "Gender",
```

```r
  "Age",
  "Balance",
  "NumOfProducts",
  "HasCrCard",
  "IsActiveMember",
  "EstimatedSalary",
  "Exited"
)]

y_train_data_tn <- train_data_tn$Tenure

x_test_data_tn <- test_data_tn[, c(
  "Geography",
  "CreditScore",
  "Gender",
  "Age",
  "Balance",
  "NumOfProducts",
  "HasCrCard",
  "IsActiveMember",
  "EstimatedSalary",
  "Exited"
)]

y_test_data_tn <- test_data_tn$Tenure
```

2.4.2 Select Features using RFE

```r
  #define RFE control using cross-validation
  ctrl_tn <- rfeControl(functions = rfFuncs,
                   method = "cv",
                   number = 4)

  #run RFE on training data
  rfe_result_tn <- rfe(x_train_data_tn,
                  #exclude target variable
                  y_train_data_tn,
                  #target variable
                  sizes = c(1:10),
                  #number of features to select (1 to 10)
                  rfeControl = ctrl)
```

```r
#print the selected features
print(rfe_result_tn)
```

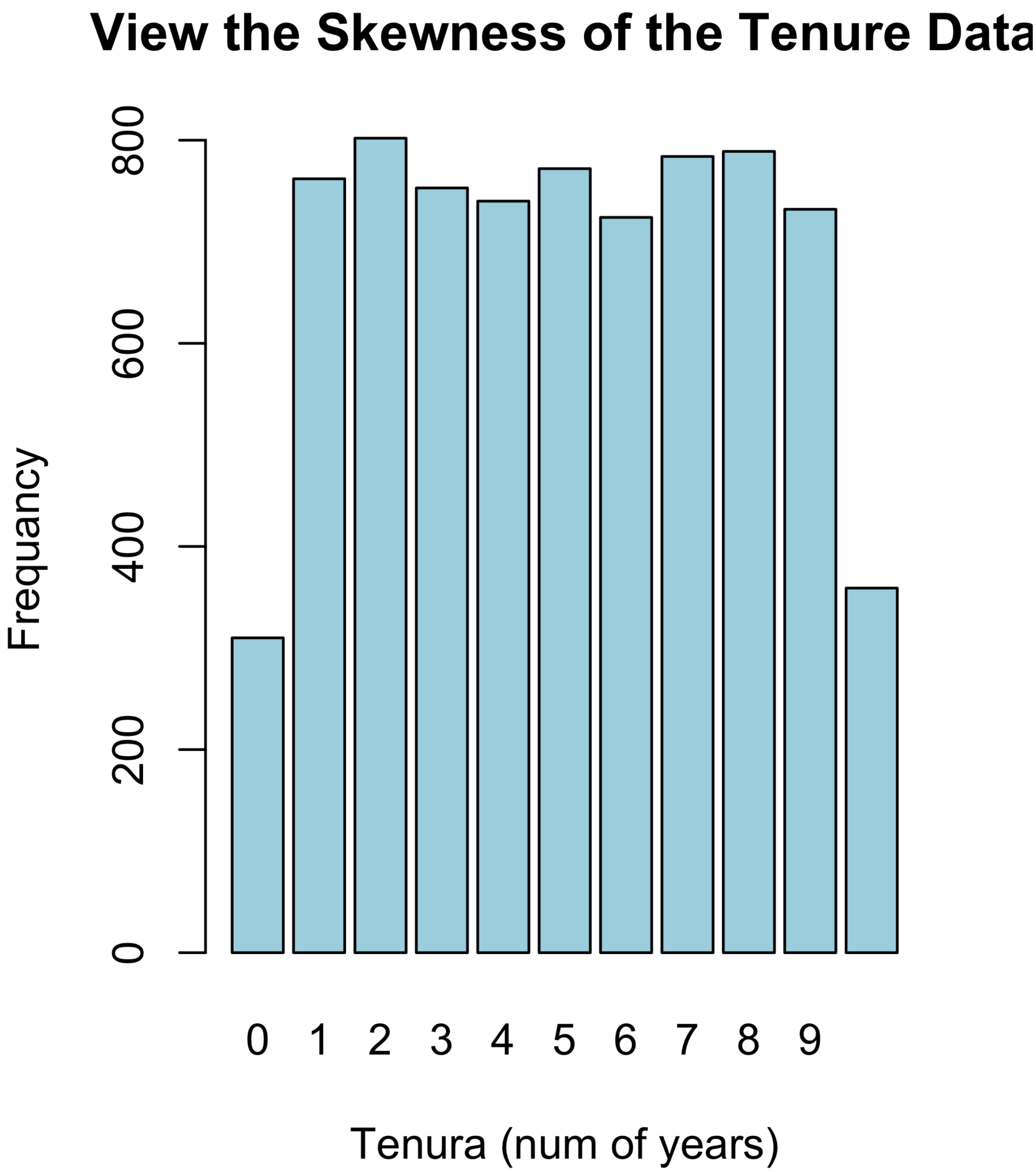
```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (5 fold)
##
## Resampling performance over subset size:
##
##  Variables  RMSE  Rsquared   MAE   RMSESD RsquaredSD    MAESD Selected
##          1 2.946 0.0006969 2.524 0.157288  0.0004784 0.106837
##          2 2.882 0.0006275 2.480 0.017736  0.0005363 0.010401
##          3 2.882 0.0011584 2.481 0.016870  0.0014473 0.009900        *
##          4 2.883 0.0007499 2.482 0.017413  0.0008283 0.010236
##          5 2.884 0.0004539 2.485 0.015663  0.0004625 0.008170
##          6 2.908 0.0010652 2.505 0.017153  0.0015470 0.008670
##          7 2.916 0.0003408 2.511 0.009877  0.0002376 0.006315
##          8 2.912 0.0001664 2.510 0.013359  0.0001505 0.008297
##          9 2.933 0.0003677 2.519 0.009988  0.0004550 0.007215
##         10 2.923 0.0005688 2.511 0.009312  0.0009470 0.007452
##
## The top 3 variables (out of 3):
##    Exited, NumOfProducts, Balance
```

2.4.3 View Skewness of Target Variable

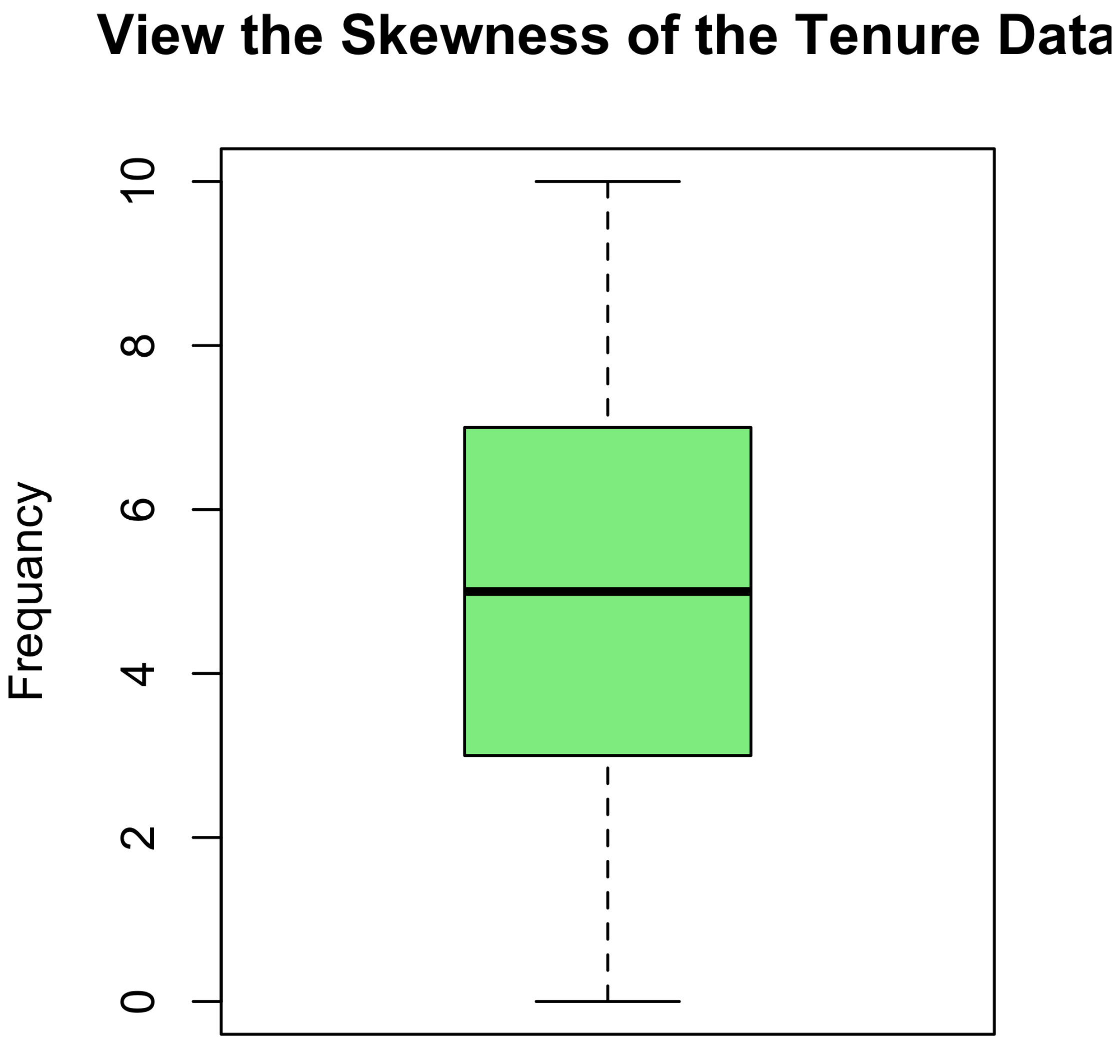
```r
# view skewness of target variable to prevent from model getting bias towards
the majority class

#view summary of data
summary(y_train_data_tn)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.000   5.000   5.018   7.000  10.000
```

```r
#View the Skewness of the Tenure Data
before_balance <- barplot(table(y_train_data_tn),main = "View the Skewness of
the Tenure Data",xlab = "Tenura (num of years)",ylab = "Frequancy",col =
"lightblue")
```




```
boxplot(y_train_data_tn, main = "View the Skewness of the Tenure Data", ylab
= "Frequancy", col = "lightgreen")
```



Since above box plot depict that sample mean X is slightly greater than M median which means this data is right skewed.

2.4.4 Handle Skewed Target Variable Data

```
y_train_data_tn <- log(y_train_data_tn + 1)  # Avoid log(0) by adding 1
```

2.4.5 Train Model for Tenure prediction

```
library(e1071)

#get selected features
selected_features_tn <- rfe_result_tn$optVariables

tn_model_data_train_rf =
data.frame(y_train_data_tn,x_train_data_tn[,c(selected_features_tn)])
tn_model_data_train_lm = data.frame(y_train_data_tn,x_train_data_tn)



model_lm_rw <- lm(y_train_data_tn ~., data = tn_model_data_train_lm)

model_lm <- lm(y_train_data_tn ~., data = tn_model_data_train_lm)
stepwise_lm_model <- step(model_lm, direction = "both", trace = 0)

num_features <- ncol(x_train_data_tn[,c(selected_features_tn)])

tuneGrid <- expand.grid(.mtry = 1:num_features)
control <- trainControl(method = "cv", number = 5)

# Train Random Forest with selected features
random_frst_tn <- train(
  y_train_data_tn ~.,
  data = tn_model_data_train_rf,
  method = "rf",
  trControl = control,
  tuneGrid = tuneGrid,
  ntree = 300
)



#view the summary of the model
print("Regression Model Summary")
```

```
## [1] "Regression Model Summary"
```

```r
summary(stepwise_lm_model)
```

```
##
## Call:
## lm(formula = y_train_data_tn ~ Balance + HasCrCard + IsActiveMember +
##     Exited, data = tn_model_data_train_lm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6921 -0.3058  0.1537  0.5052  0.8594
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.662e+00  1.758e-02  94.542  < 2e-16 ***
## Balance         -1.774e-07  1.137e-07  -1.561  0.11867
## HasCrCard1       3.001e-02  1.540e-02   1.949  0.05137 .
## IsActiveMember1 -4.055e-02  1.422e-02  -2.853  0.00435 **
## Exited1         -4.745e-02  1.813e-02  -2.617  0.00889 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6104 on 7522 degrees of freedom
## Multiple R-squared:  0.002724,   Adjusted R-squared:  0.002194
## F-statistic: 5.137 on 4 and 7522 DF,  p-value: 0.0003937
```

```r
print("Random Forest Model Summary")
```

```
## [1] "Random Forest Model Summary"
```

```r
summary(random_frst_tn)
```

```
##                 Length Class      Mode
## call               5   -none-     call
## type               1   -none-     character
## predicted       7527   -none-     numeric
## mse              300   -none-     numeric
## rsq              300   -none-     numeric
## oob.times       7527   -none-     numeric
## importance         3   -none-     numeric
## importanceSD       0   -none-     NULL
## localImportance    0   -none-     NULL
## proximity          0   -none-     NULL
## ntree              1   -none-     numeric
```

```
## mtry                 1   -none-     numeric
## forest              11   -none-     list
## coefs                0   -none-     NULL
## y                 7527   -none-     numeric
## test                 0   -none-     NULL
## inbag                0   -none-     NULL
## xNames               3   -none-     character
## problemType          1   -none-     character
## tuneValue            1   data.frame list
## obsLevels            1   -none-     logical
## param                1   -none-     list
```

2.4.6 Model Evaluation for Tenure

```
# predict the target variable using the trained model
predictions <- predict(stepwise_lm_model, newdata = x_test_data_tn)

predictions_rw <- predict(model_lm_rw, newdata = x_test_data_tn)

predictions_r <- predict(random_frst_tn, newdata =
x_test_data_tn[,c(selected_features_tn)])

# cctual values from the test data
actual <- y_test_data_tn

# calculate mean absolute error (MAE)
mae <- mean(abs(predictions - actual))
cat("MAE LN with stepwise:", mae, "\n")

## MAE LN with stepwise: 3.642581

mae_rw <- mean(abs(predictions_rw - actual))
cat("MAE LN without stepwise:", mae_rw, "\n")

## MAE LN without stepwise: 3.642039

mae_r <- mean(abs(predictions_r - actual))
cat("MAE of RF:", mae_r, "\n")

## MAE of RF: 3.641594
```

```r
# calculate mean squared error (MSE)
mse <- mean((predictions - actual)^2)
cat("MSE LN with stepwise:", mse, "\n")

## MSE LN with stepwise: 19.78904

mse_rw <- mean((predictions_rw - actual)^2)
cat("MSE LN without stepwise:", mse_rw, "\n")

## MSE LN without stepwise: 19.78716

mse_r <- mean((predictions_r - actual)^2)
cat("MSE RF:", mse_r, "\n")

## MSE RF: 19.78402

# calculate root mean squared error (RMSE)
rmse <- sqrt(mse)
cat("RMSE LN with stepwise:", rmse, "\n")

## RMSE LN with stepwise: 4.448488

rmse_rw <- sqrt(mse_rw)
cat("RMSE LN without stepwise:", rmse_rw, "\n")

## RMSE LN without stepwise: 4.448276

rmse_r <- sqrt(mse_r)
cat("RMSE RF:", rmse_r, "\n")

## RMSE RF: 4.447923

# calculate r-squared (R²)

rf_r2 <- cor(predictions_r, y_test_data_tn)^2
rf_r2_rw <- cor(predictions_rw, y_test_data_tn)^2
rf_r2_lm <- cor(predictions, y_test_data_tn)^2
cat("RF rsq",rf_r2 ,"\n")

## RF rsq 0.0001068555

cat("LN rsq with stepwise:",rf_r2_lm,"\n" )

## LN rsq with stepwise: 0.0009068435

cat("LN rsq without stepwise:",rf_r2_rw,"\n" )
```

```
## LN rsq without stepwise: 0.0008981329
```

2.4.6 Model Performance Explanation for Tenure Prediction

The data set was initially gathered not to predict customer tenure but to predict churn status, which is a yes/no result. However, we used two models; a Random Forest (RF) regression model and two Linear Regression (LM) models to try and predict tenure, which varies from one to ten years.

1. **Mean Absolute Error (MAE):**

   Mean Absolute Error is the average absolute difference between the predicted tenure and the actual tenure. The model evaluation results depict it as approximately 3.64 years. Since the tenure ranges from 1 to 10 years, an average error of about 3.64 years is very large. This error represents over 35% of the total range, meaning the model's predictions are quite inaccurate.

2. **Root Mean Squared Error (RMSE):**

   The square root of the average squared differences between expected and actual values is identified as the root mean squared error. It gives more weight to larger errors. Model evaluation results depict it as approximately 4.45 years. Since RMSE value of 4.45 years shows that prediction errors are nearly half the range of tenure. This again indicates significant prediction errors.

3. **R-squared (R^2):**

   R-squared means the proportion of the variation in tenure that the model can explain. According to the evaluation result of the model two models, it is approximately 0.0001 for the Random Forest and 0.0009 for the Linear Models. These values are nearly zero, meaning that the models do not explain any of the variability in tenure. In other words, the predictors used in the models do not have a meaningful relationship with tenure.

The error values, (MAE of 3.64 and RMSE of 4.45) which are high indicate that the predictions are wrong by a large range compared to the small range of tenure (1–10 years). Additionally, the very low R^2 values show that the models are unable to capture the factors that influence tenure. This poor performance is likely because the data set designed for churn prediction, and the available features do not provide useful information for predicting how long a customer stays with the bank. For these reasons, both the Linear Model and the Random Forest model are not suitable for predicting tenure with this data set. The information would be more suitable for churn status forecasting since the characteristics are more relevant and probably will produce better predictive performance.

TASK 03

Task 3.1

3.1.1 Load Data set

```
load_dataset <- function(filepath, sep) {
  return(read.table(filepath, sep = sep, header = TRUE, quote = "\"",
stringsAsFactors = FALSE, na.strings = c("", "NA")))
}
```

3.1.2 Implement Methods to Identify Qualitative and Quantitative Variables in the Data set

```
identify_quantitative_qualitative <- function (data,highest_num_cat){
  feature_names <- names(data)
  qualitative <- c()
  quantitative <- c()

  for (fr in feature_names) {
      if(check_quantitative_qualitative(fr,data,highest_num_cat) ==
"quantitative"){
        quantitative<- cbind(quantitative,c(fr))
      } else if(check_quantitative_qualitative(fr,data,highest_num_cat) ==
"qualitative"){
        qualitative <- cbind(qualitative,c(fr))
      }
    }

    return(list(quantitative= quantitative ,qualitative=qualitative))

}
```

```
check_quantitative_qualitative <- function(feature, data,highest_num_cat){
  feature_data <- data[[feature]]
  is_number <-  all((is.numeric(feature_data)|| is.double(feature_data)) &&
(length(unique(feature_data)) != length(feature_data) ) )
  is_categorical <- ((length(unique(feature_data))) <= highest_num_cat)

  if (is_categorical){
    return("qualitative")

  } else {
     if(is_number) {
        return("quantitative")
     } else {
        return("not-both")
     }

  }

}
```

3.1.3 Execution of Functions

```
#/Users/naduniweerasinghe/CMM-703/candy-data.csv
#/Users/naduniweerasinghe/CMM-703/Bank_Churn.csv
#/Users/naduniweerasinghe/CMM-703/iris.csv"
#/Users/naduniweerasinghe/CMM-703/mt-cars.csv"
test_data_set <-
load_dataset("/Users/naduniweerasinghe/CMM-703/Bank_Churn.csv", "," )

res <- identify_quantitative_qualitative(test_data_set,3)
res

## $quantitative
##      [,1]          [,2]  [,3]     [,4]      [,5]            [,6]
## [1,] "CreditScore" "Age" "Tenure" "Balance" "NumOfProducts"
"EstimatedSalary"
##
## $qualitative
##      [,1]        [,2]     [,3]        [,4]             [,5]
## [1,] "Geography" "Gender" "HasCrCard" "IsActiveMember" "Exited"
```

Task 3.2

3.2.1 Method Implementation

3.2.1.1 Check if missing value exit in the feature

```
#check if missing value exit in the feature
check_for_missing_values <- function(feature){
  return(sum(is.na(feature)) > 0)
}
```

3.2.1.2 Imputation method implementation for numerical data & categorical data

```
#imputation method for numerical data
imputation_numeric <- function(feature,data){
  num_mean <- mean(data[[feature]],na.rm = TRUE)
  #cat(num_mean,"num_mean","\n")
  check_num_na <- any(is.na(data[[feature]]))
  #cat(feature,"feature","\n")

  data[[feature]] <- ifelse(any(is.na(data[[feature]])), num_mean,
data[[feature]])

  data[[feature]][any(is.na(data[[feature]]))] <- num_mean

  return(data)
}

#imputation method for categorical data
imputation_categorical <- function(feature,data){
  cat_mode <- names(which.max(table(data[[feature]])))
  #cat(feature,"feature","\n")

  check_cat_na <- any(is.na(data[[feature]]))
   if(length(cat_mode) > 0){
    data[[feature]][is.na(data[[feature]])] <- cat_mode[1]
   }else{
    data[[feature]][is.na(data[[feature]])] <- cat_mode
   }
```

```
    return(data)
  }
```

3.2.1.3 Imputation method implementation

```
impute_missing_values <- function(data,highest_num_cat){
  feature_names <- names(data)

    for (fr in feature_names) {

       feature_data <- data[[fr]]

        if(check_for_missing_values(feature_data)){

          if(check_quantitative_qualitative(fr,data,highest_num_cat) ==
"quantitative"){
            data <- imputation_numeric(fr,data)

          }else if(check_quantitative_qualitative(fr,data,highest_num_cat) ==
"qualitative"){
            data <- imputation_categorical(fr,data)
          }

        } else {
          next
        }
    }

    return(data)

  }
```

3.2.1.4 Method execution

```
cat("\n\n","BEFORE MISSING VALUE IMPUTATION","\n")

##
##
##  BEFORE MISSING VALUE IMPUTATION

colSums(is.na(test_data_set))
```

```
##      CustomerId         Surname     CreditScore       Geography
Gender
##               0               0               0               0
0
##             Age          Tenure         Balance   NumOfProducts
HasCrCard
##               0               0               0               0
0
##  IsActiveMember EstimatedSalary          Exited
##               0               0               0
```

```r
head(test_data_set)
```

```
##   CustomerId  Surname CreditScore Geography Gender Age Tenure   Balance
## 1   15634602 Hargrave         619    France Female  42      2      0.00
## 2   15647311     Hill         608     Spain Female  41      1  83807.86
## 3   15619304     Onio         502    France Female  42      8 159660.80
## 4   15701354     Boni         699    France Female  39      1      0.00
## 5   15737888 Mitchell         850     Spain Female  43      2 125510.82
## 6   15574012      Chu         645     Spain   Male  44      8 113755.78
##   NumOfProducts HasCrCard IsActiveMember EstimatedSalary Exited
## 1             1         1              1       101348.88      1
## 2             1         0              1       112542.58      0
## 3             3         1              0       113931.57      1
## 4             2         0              0        93826.63      0
## 5             1         1              1        79084.10      0
## 6             2         1              0       149756.71      1
```

```r
#executing missing value imputation method
new_dataset <- impute_missing_values(test_data_set,10)

cat("\n \n","AFTER MISSING VALUE IMPUTATION","\n")
```

```
## 
## 
##  AFTER MISSING VALUE IMPUTATION
```

```r
colSums(is.na(new_dataset))
```

```
##      CustomerId         Surname     CreditScore       Geography
Gender
##               0               0               0               0
0
##             Age          Tenure         Balance   NumOfProducts
HasCrCard
```

```
##               0               0               0               0
0
##  IsActiveMember EstimatedSalary          Exited
##               0               0               0
```

```
head(new_dataset)
```

```
##   CustomerId  Surname CreditScore Geography Gender Age Tenure   Balance
## 1   15634602 Hargrave         619    France Female  42      2      0.00
## 2   15647311     Hill         608     Spain Female  41      1  83807.86
## 3   15619304     Onio         502    France Female  42      8 159660.80
## 4   15701354     Boni         699    France Female  39      1      0.00
## 5   15737888 Mitchell         850     Spain Female  43      2 125510.82
## 6   15574012      Chu         645     Spain   Male  44      8 113755.78
##   NumOfProducts HasCrCard IsActiveMember EstimatedSalary Exited
## 1             1         1              1       101348.88      1
## 2             1         0              1       112542.58      0
## 3             3         1              0       113931.57      1
## 4             2         0              0        93826.63      0
## 5             1         1              1        79084.10      0
## 6             2         1              0       149756.71      1
```

Task 3.3

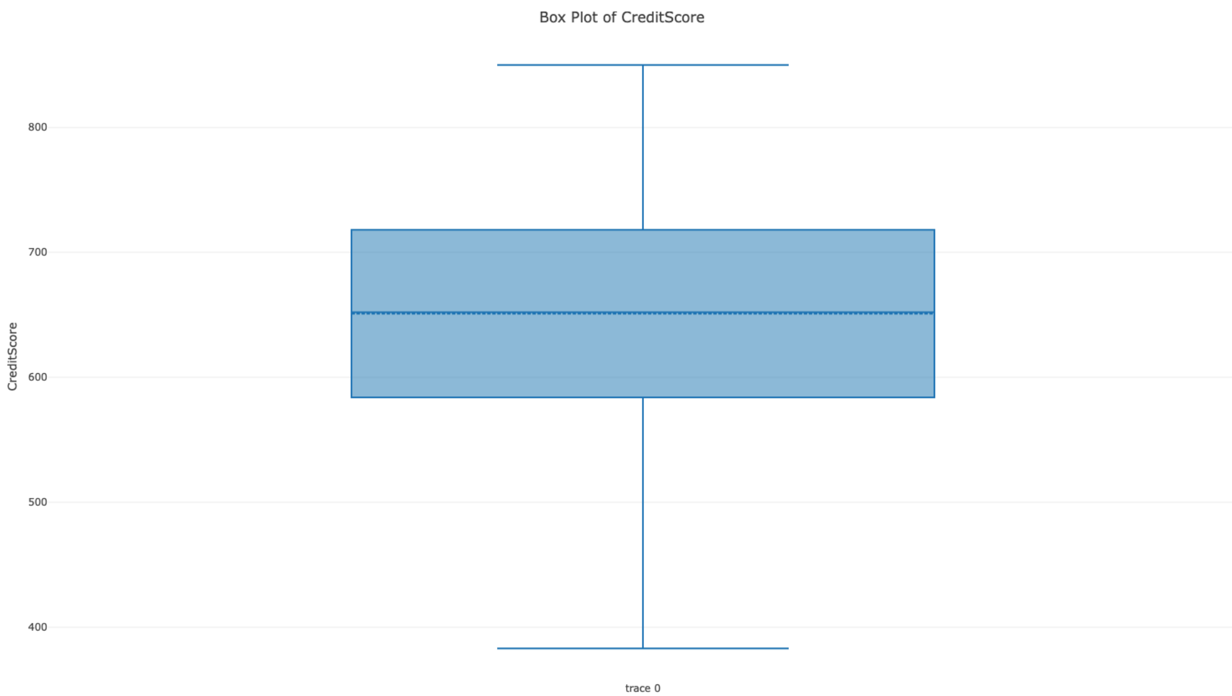
3.3.1 Method implementation for outlier removal

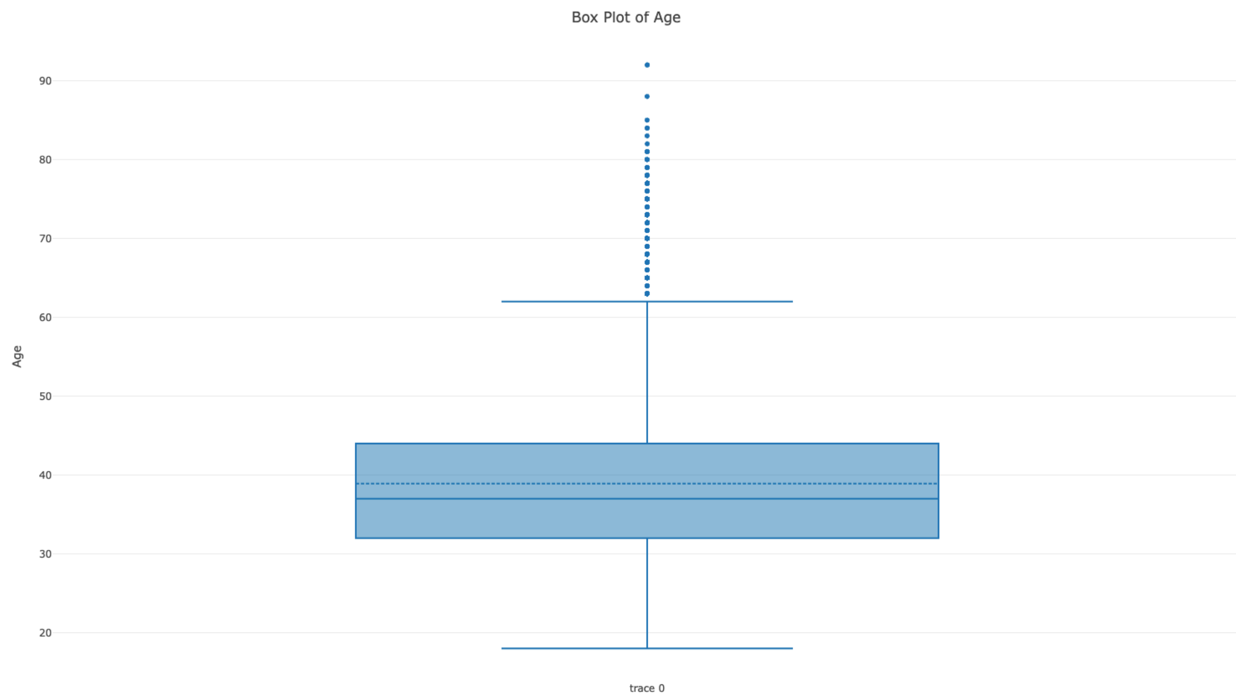
```
iqr_method <- function(data,feature){

  Q1_value <- quantile(data[[feature]], 0.25, na.rm = TRUE)
  Q3_value <- quantile(data[[feature]], 0.75, na.rm = TRUE)
  IQR_ <- Q3_value - Q1_value

  # find the lower bound and the upper bound
  lower_bound_ <- Q1_value - 1.5 * IQR_
  upper_bound_ <- Q3_value + 1.5 * IQR_

  filtered_data_ <- data[data[[feature]] >= lower_bound_ & data[[feature]] <=
upper_bound_, ]
  return(filtered_data_)

}
```

```r
zcore_method <- function(data,feature){
  mean_ <- mean(data[[feature]], na.rm = TRUE)
  standard_d <- sd(data[[feature]], na.rm = TRUE)

  z_scores <- (data[[feature]] - mean_) / standard_d

  data <- data[which(abs(z_scores) > 3), ]

  return(data)
}

outlier_remove <- function(highest_num_cat,data,outlier_method = "IQR"){
   feature_names <- names(data)

    for (fr in feature_names) {
     if(check_quantitative_qualitative(fr,data,highest_num_cat) ==
"quantitative"){
      if(tolower(outlier_method) == "zcore"){
        data <- zcore_method(data,fr)
      }else {
        data <- iqr_method(data,fr)
      }
     } else {
       next
     }
      return(data)
    }
}
```

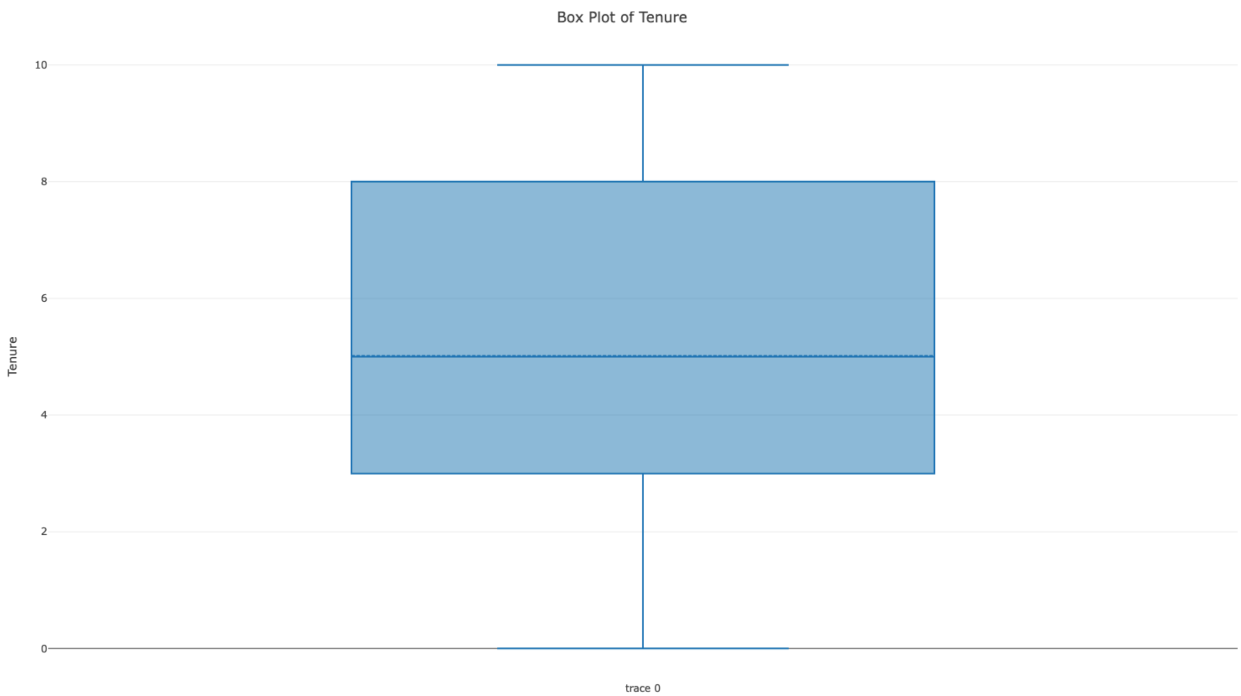
3.3.2 Method Execution

```
outlier_removed_new_dataset <- outlier_remove(10,new_dataset)

for( fr in as.vector(res$quantitative)) {
  outlier_removed_bx <-plot_ly(
              data = outlier_removed_new_dataset,
              y = ~outlier_removed_new_dataset[[fr]],
              type = 'box',
              boxmean = TRUE  # Optionally show the mean inside the box plot
            ) %>% layout(
              title = paste("Box Plot of", fr),
              yaxis = list(title = fr)
            )

  print(outlier_removed_bx)
}
```

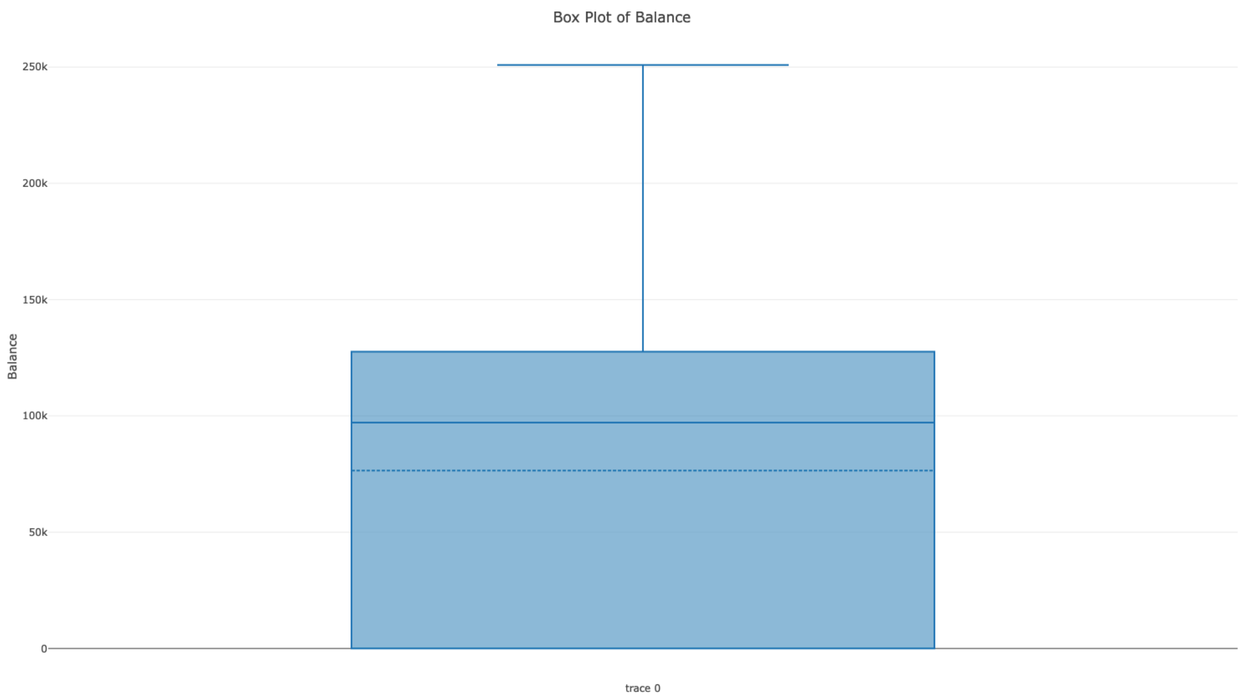

Box Plot of CreditScore
800
700
600
500
400
CreditScore
trace 0

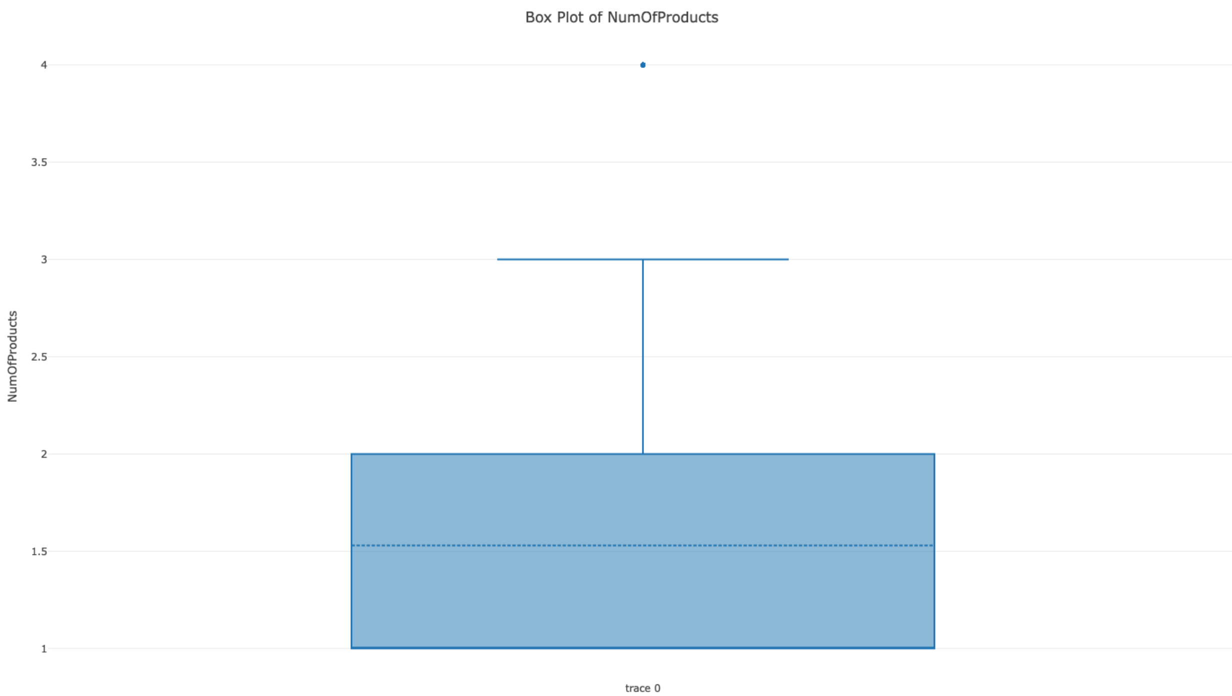


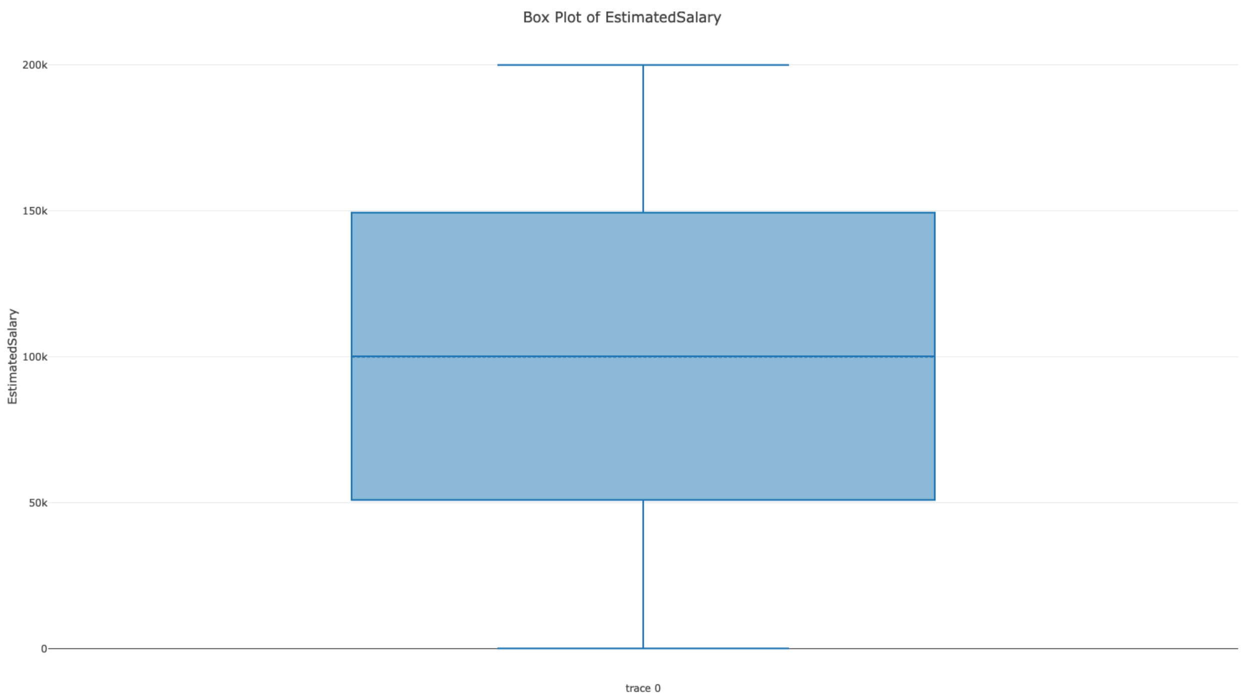
Box Plot of Age
Age
90
80
70
60
50
40
30
20
trace 0



Box Plot of Tenure
Tenure
10
8
6
4
2
0
trace 0


Box Plot of Balance
250k
200k
150k
100k
50k
0
Balance
trace 0


Box Plot of NumOfProducts
NumOfProducts
4
3.5
3
2.5
2
1.5
1
trace 0


Box Plot of EstimatedSalary
200k
150k
100k
50k
0
EstimatedSalary
trace 0

Task 3.4

3.4.1 Implement Methods to View Data in Relevant Plots

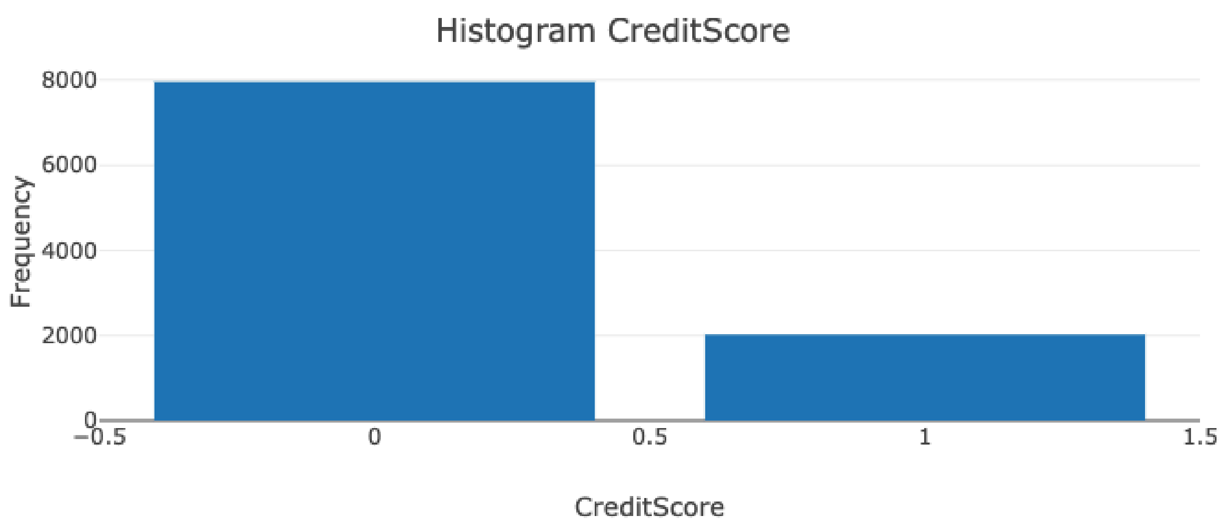
```
view_data_in_plot <- function(data, highest_num_cat) {
  feature_names <- names(data)
  quant_plots <- list()
  qulitat_plot <- list()

  for (fr in feature_names) {
    summary_data <- as.data.frame(table(data[[fr]]))
    colnames(summary_data) <- c("Category", "Count")

    if (check_quantitative_qualitative(fr, data, highest_num_cat) ==
"quantitative") {
      quantitative <- plot_ly(
        data = data,
        y = ~ data[[fr]],
        type = "box",
        boxmean = TRUE
      ) %>% layout(
        title = paste("Box Plot of", fr),
        yaxis = list(title = fr)
      )

      quantitative1 <- plot_ly(
        data = data,
        x = ~ data[[fr]],
        type = "histogram"
      ) %>% layout(
        title = paste("Histogram", fr),
        xaxis = list(title = fr),
        yaxis = list(title = "Frequency"),
        margin = list(b = 200),
        bargap = 0.2
      )

      quant_plots[[fr]] <-  quantitative1


    } else if (check_quantitative_qualitative(fr, data, highest_num_cat) ==
"qualitative") {
      qualitative <- plot_ly(
```

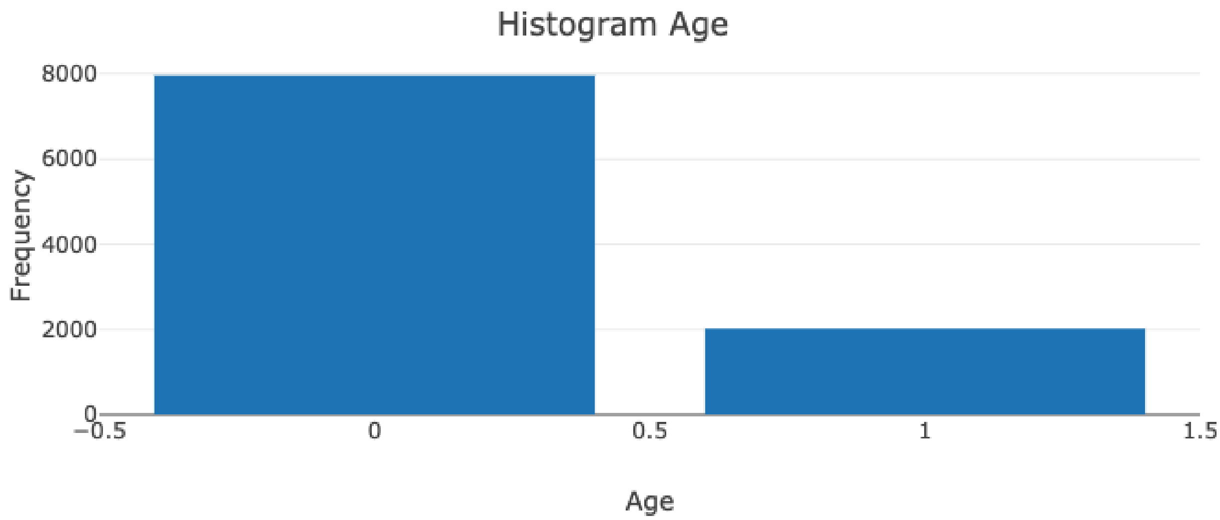
```
        data = summary_data,
        x = ~ Category,
        y = ~ Count,
        type = 'bar'
      ) %>% layout(
        title = paste("View Data of", fr),
        xaxis = list(title = fr, tickangle = -45),
        yaxis = list(title = "Count"),
        margin = list(b = 200)
      )

      qulitat_plot[[fr]] <- qualitative

    }
  }

  return(list(quant_plots = quant_plots, qulitat_plot = qulitat_plot))
}
```

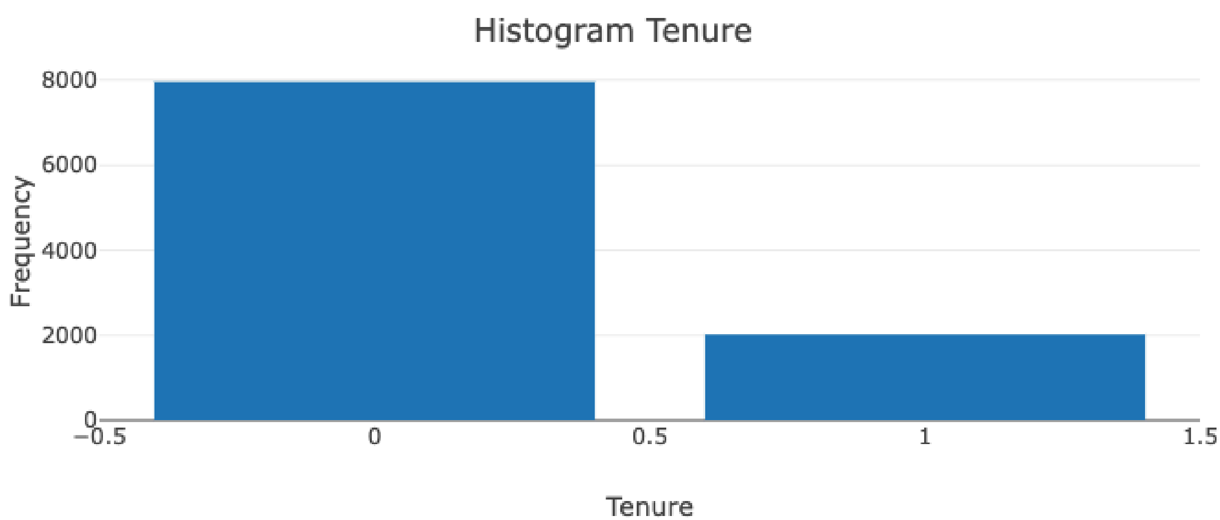
3.4.2 Method Execution

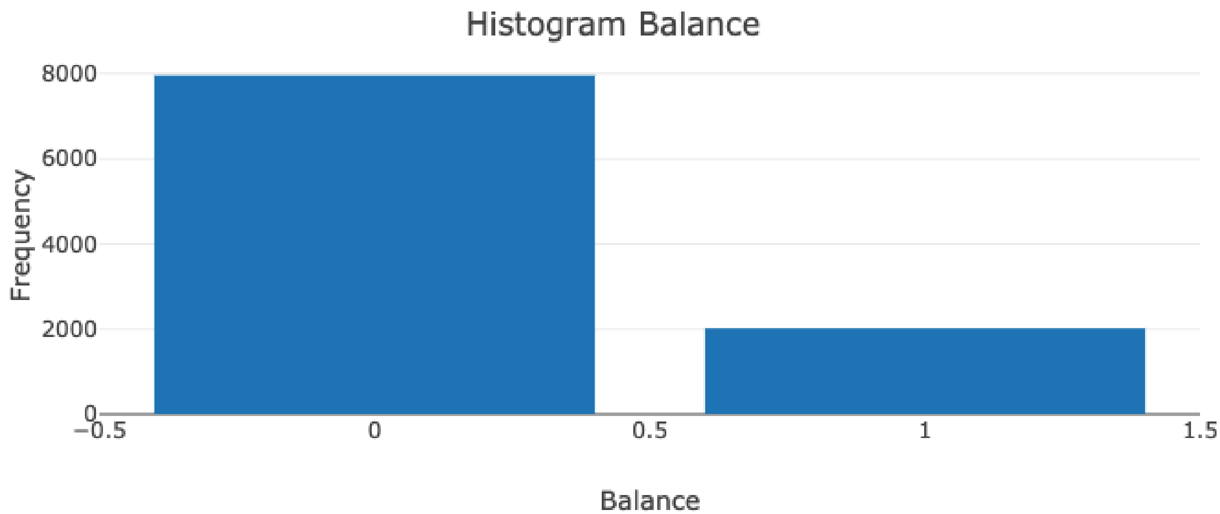
```
res_p <- view_data_in_plot(outlier_removed_new_dataset,10)


res_p

## $quant_plots
## $quant_plots$CreditScore
##
## $quant_plots$Age
##
## $quant_plots$Tenure
##
## $quant_plots$Balance
##
## $quant_plots$EstimatedSalary
##
##
## $qulitat_plot
## $qulitat_plot$Geography
##
```

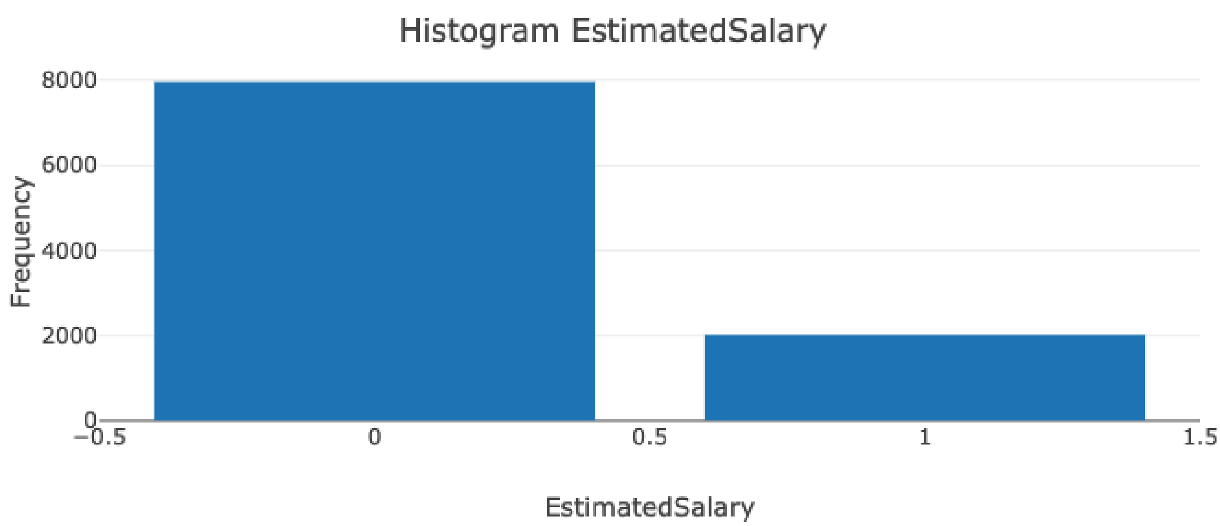
```
## $qulitat_plot$Gender
##
## $qulitat_plot$NumOfProducts
##
## $qulitat_plot$HasCrCard
##
## $qulitat_plot$IsActiveMember
##
## $qulitat_plot$Exited
```

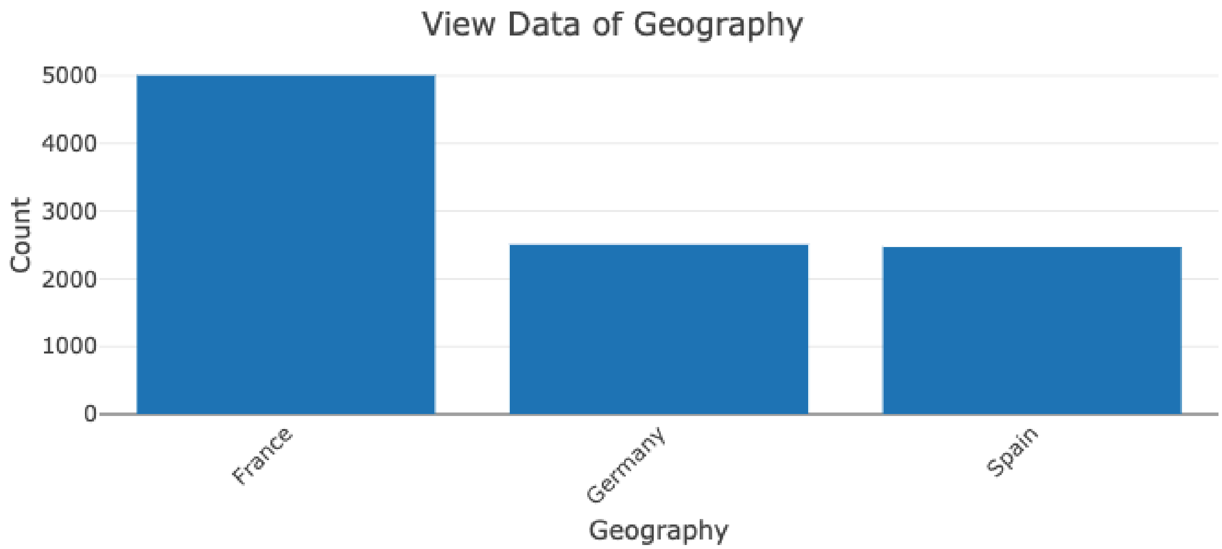


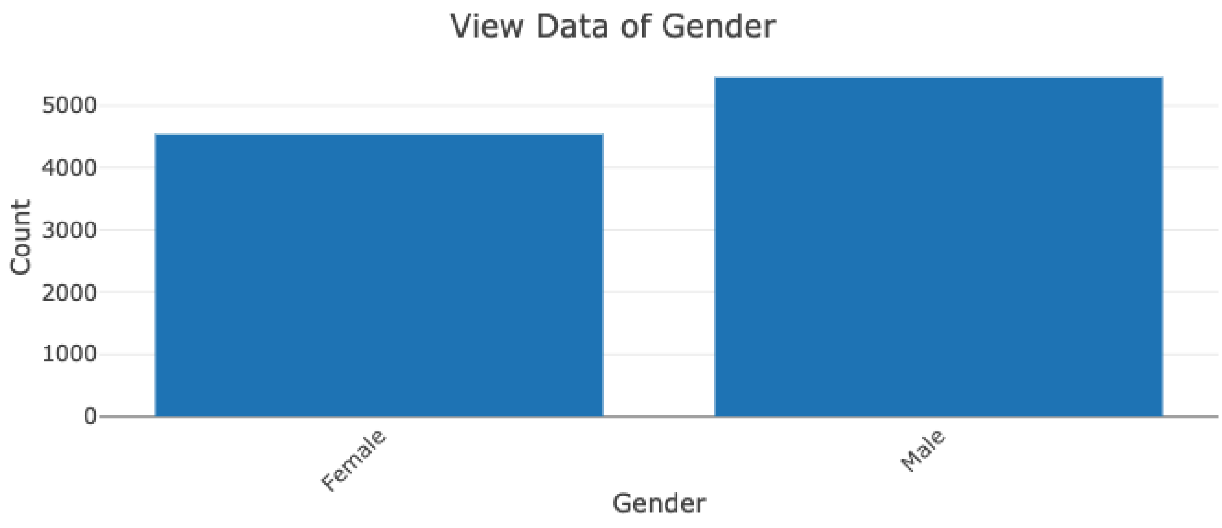

Histogram Age
8000
6000
4000
2000
0
Frequency
−0.5
0
0.5
1
1.5
Age

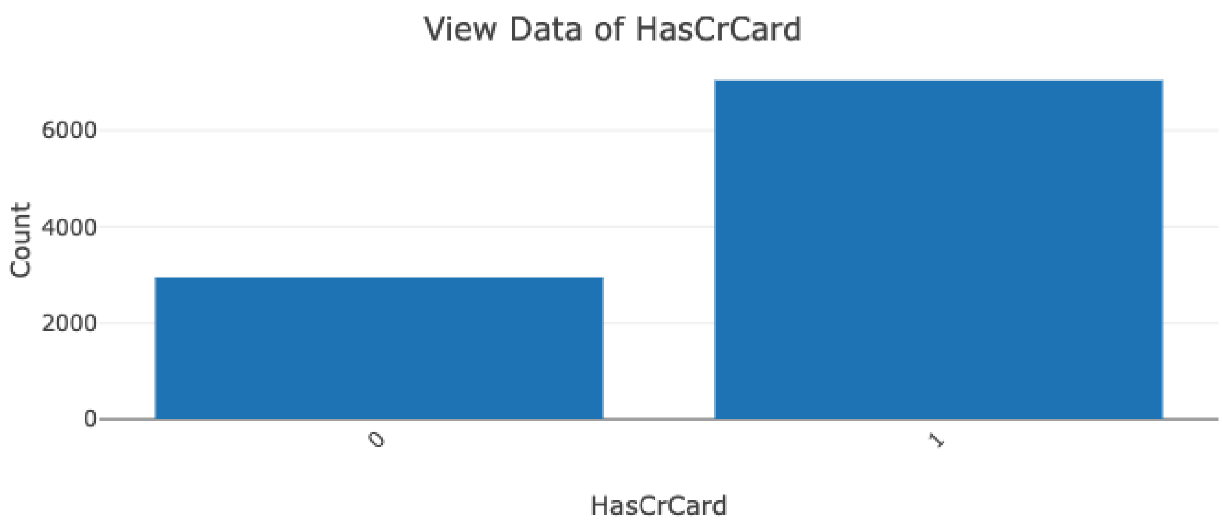

Histogram Tenure
8000
6000
4000
2000
0
Frequency
−0.5
0
0.5
1
1.5
Tenure

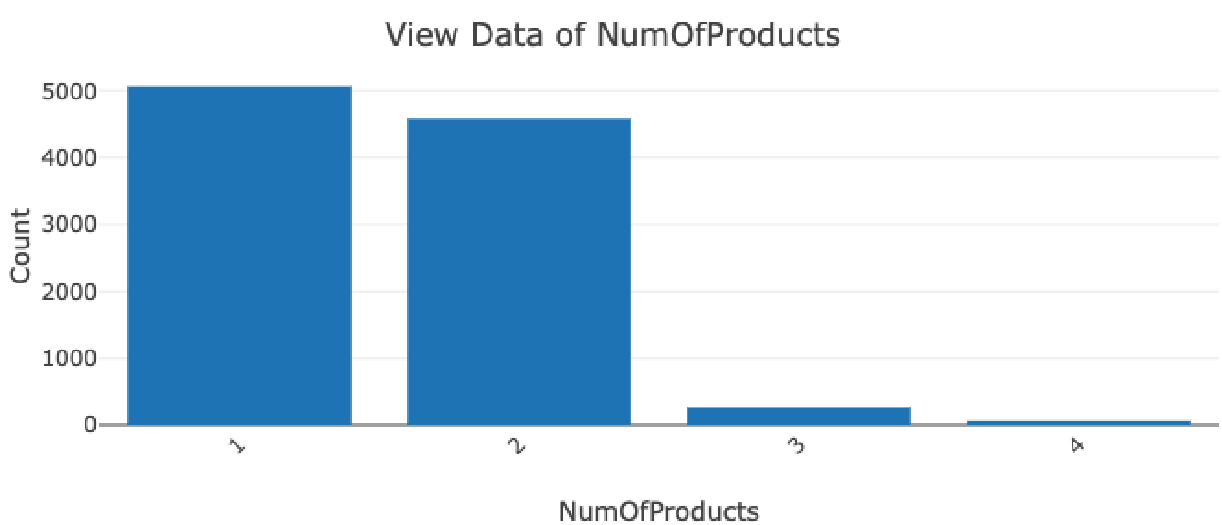

Histogram Balance
8000
6000
4000
2000
0
Frequency
−0.5
0
0.5
1
1.5
Balance

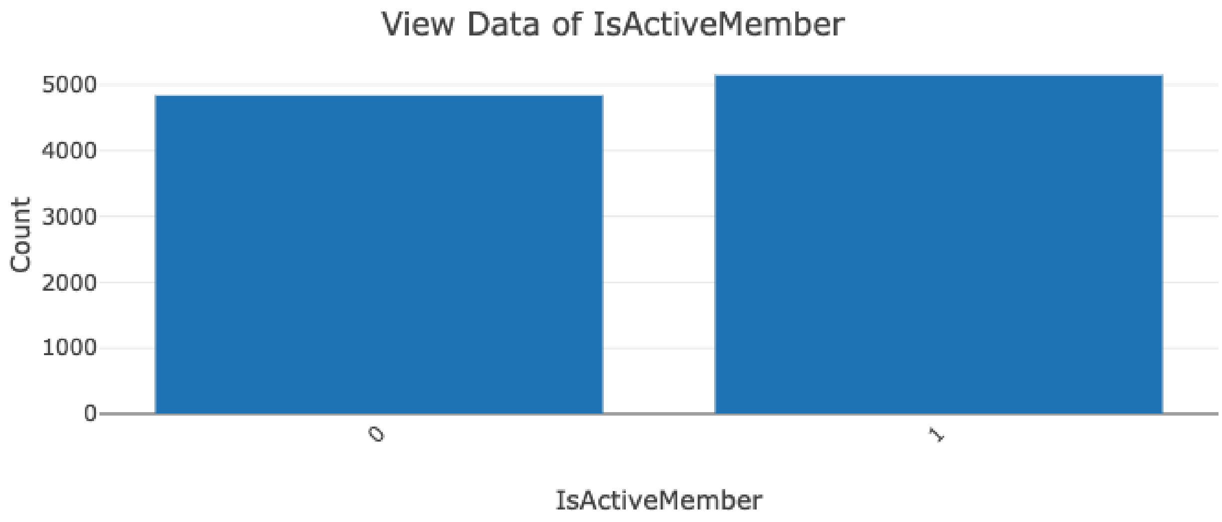

Histogram EstimatedSalary
8000
6000
4000
2000
0
Frequency
−0.5
0
0.5
1
1.5
EstimatedSalary

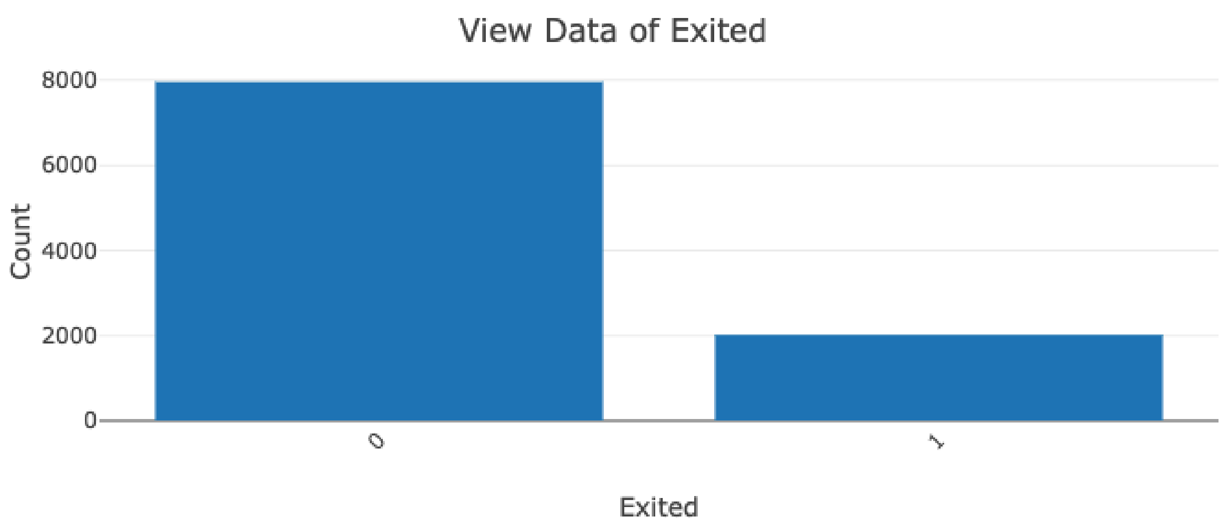

View Data of Geography
5000
4000
3000
2000
1000
0
Count
France
Germany
Spain
Geography


View Data of Gender
5000
4000
3000
2000
1000
0
Count
Female
Male
Gender


View Data of HasCrCard
Count
6000
4000
2000
0
0
1
HasCrCard


View Data of NumOfProducts
Count
5000
4000
3000
2000
1000
0
1
2
3
4
NumOfProducts


View Data of IsActiveMember
5000
4000
3000
2000
1000
0
Count
0
1
IsActiveMember


View Data of Exited
8000
6000
4000
2000
0
Count
0
1
Exited

Task 3.5

3.5.1 Implement Methods to Predict Data for Given Variable

3.5.1.1 Remove Unwanted Columns

```
remove_unwanted_columns <- function(data,highest_cat_level){
 print(data)
 qt_qlt <-  identify_quantitative_qualitative(data,highest_cat_level)
 cbind_qt_qlt <- cbind(qt_qlt$quantitative,qt_qlt$qualitative)

 return (data[,cbind_qt_qlt])
}
```

3.5.1.2 convert qualitative data to factors

```
convert_qualitative_data_to_factors <- function(data, highest_cat_level){
  feature_names <- names(data)

    for(fr in feature_names){
      if (check_quantitative_qualitative(fr, data, highest_cat_level) ==
"quantitative") {
        next
      } else if(check_quantitative_qualitative(fr, data, highest_cat_level)
== "qualitative"){

        data[[fr]] <- factor(data[[fr]])
      }
    }
  return(data)
}
```

3.5.1.3 Data Prepossessing

```
data_preprocessing <- function(data, highest_cat_level) {

  plots <- view_data_in_plot(data, highest_cat_level)

  remove_unwanted_cols <- remove_unwanted_columns(data, highest_cat_level)

  df_without_missing_values <- impute_missing_values(remove_unwanted_cols,
highest_cat_level)

  df_without_outliers <- outlier_remove(highest_cat_level,
df_without_missing_values)

  factor_conversion <-
convert_qualitative_data_to_factors(df_without_outliers,highest_cat_level)
  return(list (df_without_outliers = factor_conversion, plots = plots))
}
```

3.5.1.4 Data Splitting

```
data_splitting <- function(data, target_variable) {
  set.seed(123)

  train_index_ <- createDataPartition(data[[target_variable]], p = 0.8, list
= FALSE)

  train_data <- data[train_index_, ]
  test_data <- data[-train_index_, ]
```

```
  x_train <- train_data[, !(colnames(train_data) == target_variable)]
  y_train <- train_data[, (colnames(train_data) == target_variable)]
  y_test <- test_data[, (colnames(test_data) == target_variable)]
  x_test <- test_data[, !(colnames(test_data) == target_variable)]

  return(list(
    x_train = x_train,
    y_train = y_train,
    y_test = y_test,
    x_test = x_test
  ))
}
```

3.5.1.5 Data Imbalance

```
fix_class_imbalance <- function(x_train, y_train) {

  cat(class(as.data.frame(x_train)))
  x_train_df <- as.data.frame(x_train)
  # Apply one-hot encoding (convert factors to dummy variables)
  predict_vars <- x_train_df %>%
    mutate(across(where(is.factor), as.numeric))

  print(dim(predict_vars))    # Should return rows and columns
  print(length(y_train))
  # Check the structure
  str(predict_vars)

  # Apply SMOTE
  smote_res <- SMOTE(
    X = predict_vars,
    target = y_train,
    K = 2,
    # Number of nearest neighbors
    dup_size = 6
  )          # Oversampling rate

  # Check class distribution after SMOTE
  table(smote_result$data$class)

  # Check new class distribution
```

```
  before_balance
  class_counts <- table(smote_result$data$class)

  return((smote_res))

}
```

3.5.1.6 Feature Selection

```
feature_selection_for_model <- function(x_train, y_train) {
  x_train_df <- as.data.frame(x_train)
  # Apply one-hot encoding (convert factors to dummy variables)
  x_train_df <- x_train_df %>%
    mutate(across(where(is.factor), as.numeric))
  # Define RFE control using cross-validation
  ctrl_ <- rfeControl(functions = rfFuncs,
                      method = "cv",
                      number = 5)


  # Run RFE on training data
  rfe_res <- rfe(x_train_df,
                 # Exclude target variable
                 y_train,
                 # Target variable
                 sizes = c(1:5),
                 # Number of features to select (1 to 5)
                 rfeControl = ctrl_)

  # Print the selected features
  print(rfe_res)
  return(as.vector(rfe_res$optVariables))
}
```

3.5.1.7 Run Best Model Method Implement

```
select_and_run_best_model <- function(target_variable, data, highest_num_cat)
{
  force(data)
  is_feature_exist <- target_variable %in% colnames(data)

  y_check <- data[[target_variable]]
  is_binary <- ((is.factor(y_check) && length(levels(y_check)) == 2) ||
              (is.numeric(y_check) && length(unique(y_check)) == 2))

  if (!any(is_feature_exist)) {
    print(paste(target_variable, " is not Found in the Data set"))
  }
  else {

    cleaned_data <- data_preprocessing(data, highest_num_cat)
    splited_d <- data_splitting(cleaned_data$df_without_outliers,
target_variable)
    fr_selected <- feature_selection_for_model(splited_d$x_train,
splited_d$y_train)


    x_train_fr_selected <- splited_d$x_train[, fr_selected]
    x_test_fr_selected <- splited_d$x_test[, fr_selected]

    train_model_data_rf <- data.frame(x_train_fr_selected, y_train =
splited_d$y_train)
    train_model_data_lm <- data.frame(splited_d$x_train, y_train =
splited_d$y_train)
```

```r
    if (check_quantitative_qualitative(target_variable, data,highest_num_cat)
== "quantitative") {

      ln_model <- lm(y_train ~., data = train_model_data_lm)
      stepwise_ln_new <- step(ln_model, direction = "both", trace = 0)

      num_fr <- ncol(x_train_fr_selected)

      tuneGrid <- expand.grid(.mtry = 1:num_fr)
      control <- trainControl(method = "cv", number = 5)

      random_fmodel <- train(
        y_train ~.,
        data = train_model_data_rf,
        method = "rf",
        trControl = control,
        tuneGrid = tuneGrid,
        ntree = 300
      )

      pred_rf <- predict(random_fmodel, newdata = x_test_fr_selected)
      pred_lm <- predict(stepwise_ln_new, newdata = splited_d$x_test)

      rmse_lm <- sqrt(mean((pred_lm - splited_d$y_test)^2))
      rmse_rf <- sqrt(mean((pred_rf - splited_d$y_test)^2))

      best_model_type <- names(which.min(c(LM = rmse_lm, RF = rmse_rf)))

      if (best_model_type == "LM") {
        chosen_model <- ln_model
        chosen_preds <- pred_lm
      } else {
        chosen_model <- random_fmodel
        chosen_preds <- pred_rf
      }

      model_summary <- capture.output(summary(chosen_model))

      performance <- paste("RMSE LM =", round(rmse_lm, 2),
                           "| RMSE RF =", round(rmse_rf, 2),
                           "| Selected:", best_model_type)
```

```
      pred_real <- function() {
        plot_ly(x = splited_d$y_test, y = chosen_preds, type = 'scatter',
mode = 'markers') %>%
        layout(
          title = "Observed vs. Predicted",
          xaxis = list(title = "Observed Values"),
          yaxis = list(title = "Predicted Values"),
          shapes = list(
            list(
              type = "line",
              x0 = min(splited_d$y_test), x1 = max(splited_d$y_test),
              y0 = min(splited_d$y_test), y1 = max(splited_d$y_test),
              line = list(dash = "dot", width = 2)
            )
          )
        )
      }

      return(list(
        response_type = "continuous",
        best_model = chosen_model,
        model_type = best_model_type,
        predictions = chosen_preds,
        performance = performance,
        model_summary = model_summary,
        plot_list = cleaned_data$plots,
        pred_vs_real = pred_real
      ))

    } else if ( is_binary && check_quantitative_qualitative(target_variable,
data,highest_num_cat ) == "qualitative") {

    glm_model <- glm(y_train ~ ., data = train_model_data_lm, family =
binomial)
    stepwise_glm_model <- step(glm_model, direction = "both", trace = 0)

    rf_bi_model <- randomForest(
      y = splited_d$y_train,
      x = x_train_fr_selected,
      ntree = 500,
      mtry = 2,
      sampsize = c(length(x_train_fr_selected)/2,
```

```
length(x_train_fr_selected)/2),
      replace = TRUE
    )

    glm_pred <- predict(stepwise_glm_model, splited_d$x_test, type =
"response")
    glm_pred_class <- ifelse(glm_pred > 0.5, 1, 0)
    glm_pred_class <- as.factor(glm_pred_class)

    rf_bi_pred <- predict(rf_bi_model, x_test_fr_selected, type = "prob")[,2]
    rf_bi_pred_class <- ifelse(rf_bi_pred > 0.5, 1, 0)
    rf_bi_pred_class <- factor(rf_bi_pred_class)

    # evaluate Performance
    glm_cm <- confusionMatrix(glm_pred_class, splited_d$y_test, positive =
"1")
    rf_bi_cm <- confusionMatrix(rf_bi_pred_class, splited_d$y_test, positive
= "1")

    rf_bi_roc <- roc(splited_d$y_test, rf_bi_pred)
    glm_roc <- roc(splited_d$y_test, glm_pred)

    glm_auc <- auc(glm_roc)
    rf_bi_auc <- auc(rf_bi_roc)

    best_model_type <- names(which.min(c(GLM = glm_auc, RF = rf_bi_auc)))

    if (best_model_type == "GLM") {
      chosen_model <- glm_model
      performance <- paste("AUC GLM =", round(glm_auc, 3))
      model_auc <- glm_auc
      model_roc <- glm_roc
      model_name <- "Logistic Regression"
      model_cm <- glm_cm
      chosen_pred <- glm_pred
    } else {
      chosen_model <- rf_bi_model
      performance <- paste("AUC RF =", round(rf_bi_auc, 3))
      model_auc <- rf_bi_auc
      model_roc <- rf_bi_roc
      model_name <- "Random Forest"
      model_cm <- rf_bi_cm
      chosen_pred <-rf_bi_pred
```

```
    }
    model_summary <- capture.output(summary(chosen_model))

    roc_data <- data.frame(
      Model = model_name,
      FPR = c(1 - model_roc$specificities),
      TPR = c(model_roc$sensitivities)
    )

    roc_plt <- function(){

      plot_ly(roc_data, x = ~FPR, y = ~TPR, type = "scatter", mode = "lines",
color = ~Model) %>%
      layout(
        title = paste("ROC Curve Comparison (AUC", model_name, ":",
round(model_auc, 3), ")"),
        xaxis = list(title = "False Positive Rate"),
        yaxis = list(title = "True Positive Rate")
      )
    }

    return(list(
      response_type = "binary",
      best_model = chosen_model,
      model_type = best_model_type,
      predictions = chosen_pred,
      performance = performance,
      confusion_matrix = model_cm,
      model_summary = model_summary,
      roc_plot = roc_plt,
      plot_list = cleaned_data$plots
    ))

    } else {
      print("This Method is design for Binary Classification")
    }
  }
}
```

3.5.1.8 Execute Method to Check

```
select_and_run_best_model("Exited",test_data_set,3)
```

```
##  [ reached 'max' / getOption("max.print") -- omitted 2308 rows ]
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (5 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.8205 0.1913   0.004213 0.02799
##          2   0.8390 0.4338   0.007389 0.03488
##          3   0.8505 0.4153   0.006414 0.02799
##          4   0.8539 0.4473   0.007311 0.03238
##          5   0.8553 0.4637   0.004412 0.02016
##         10   0.8568 0.4719   0.009048 0.03535        *
##
## The top 5 variables (out of 10):
##    NumOfProducts, Age, IsActiveMember, Balance, Geography
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

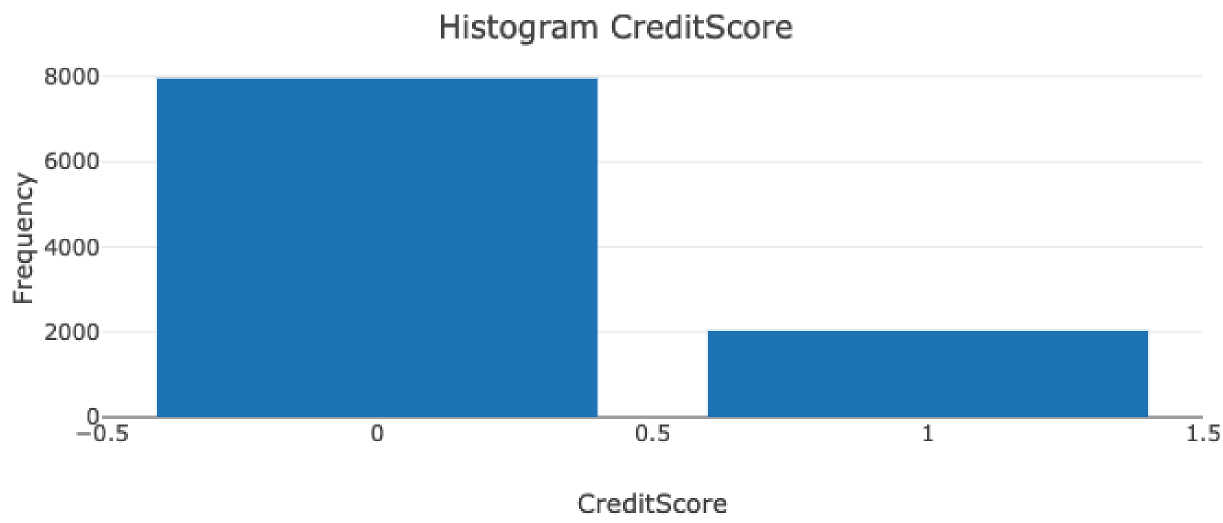
```
## $response_type
## [1] "binary"
##
## $best_model
##
```

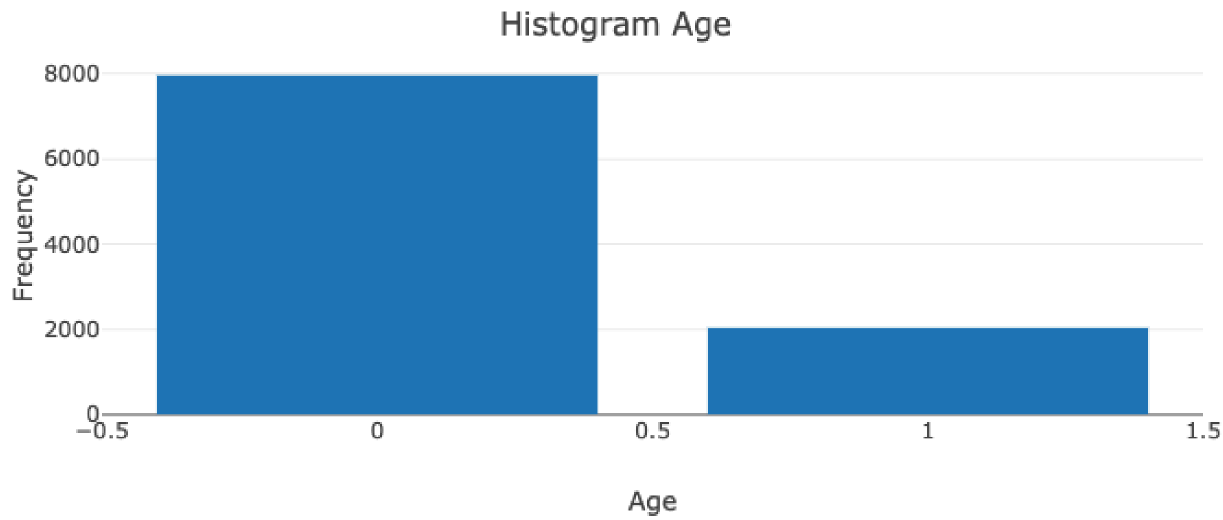
```
## Call:  glm(formula = y_train ~ ., family = binomial, data =
train_model_data_lm)
##
## Coefficients:
##      (Intercept)       CreditScore               Age            Tenure
##       -3.474e+00        -4.595e-04         7.043e-02        -1.562e-02
##          Balance     NumOfProducts   EstimatedSalary  GeographyGermany
##        2.503e-06        -1.012e-01         3.976e-07         7.938e-01
##   GeographySpain        GenderMale        HasCrCard1   IsActiveMember1
##        6.230e-02        -5.135e-01        -3.148e-03        -1.075e+00
##
## Degrees of Freedom: 7988 Total (i.e. Null);  7977 Residual
## Null Deviance:       8051
## Residual Deviance: 6862  AIC: 6886
##
## $model_type
## [1] "GLM"
##
## $predictions
##          1          5         11         14         16         19
20
## 0.12215416 0.16371864 0.11413410 0.09905349 0.22514703 0.23441603
0.03074925
##         25         30         31         37         41         49
52
## 0.08281248 0.03824668 0.22306284 0.05789558 0.18975611 0.15679423
0.30930501
##         57         61         62         63         65         68
86
## 0.28046000 0.29547445 0.26569590 0.13060333 0.07123034 0.16938885
0.54347533
##         88         94        100        107        108        109
117
## 0.07486772 0.02501275 0.11034591 0.10526096 0.19362511 0.16178917
0.47045130
##        122        124        132        138        141        147
```

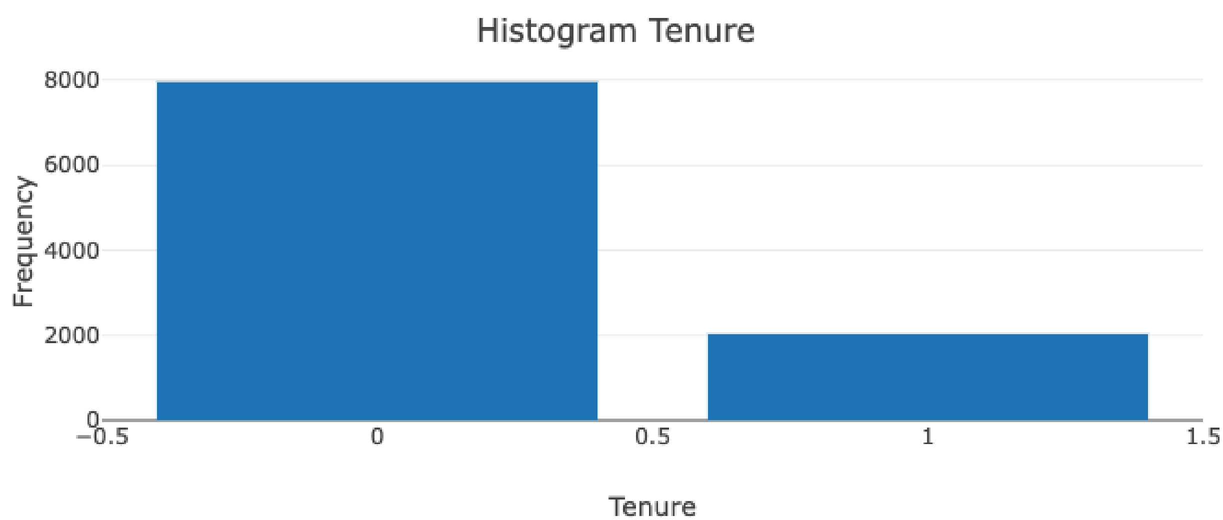
```
## $performance
## [1] "AUC GLM = 0.779"
##
## $confusion_matrix
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1549  317
##          1   43   87
##
##                Accuracy : 0.8196
##                  95% CI : (0.8021, 0.8363)
##     No Information Rate : 0.7976
##     P-Value [Acc > NIR] : 0.007107
##
##                   Kappa : 0.2521
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.21535
##             Specificity : 0.97299
##          Pos Pred Value : 0.66923
##          Neg Pred Value : 0.83012
##              Prevalence : 0.20240
##          Detection Rate : 0.04359
##    Detection Prevalence : 0.06513
##       Balanced Accuracy : 0.59417
##
##        'Positive' Class : 1
##
##
## $model_summary
##  [1] ""
##  [2] "Call:"
##  [3] "glm(formula = y_train ~ ., family = binomial, data =
train_model_data_lm)"
##  [4] ""
```

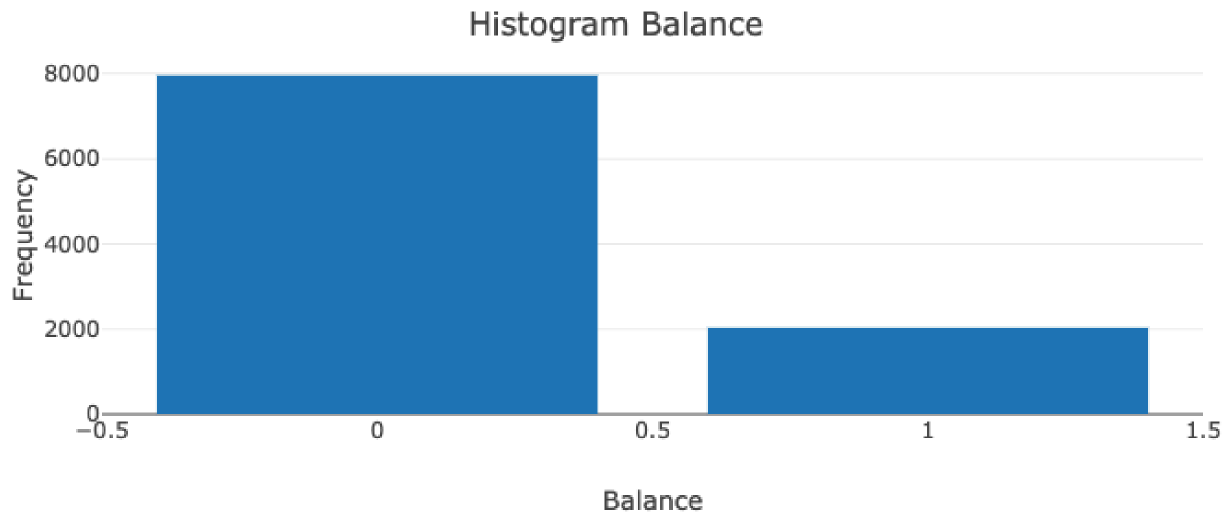
```
##  [5] "Coefficients:"
##  [6] "                    Estimate Std. Error z value Pr(>|z|)    "
##  [7] "(Intercept)      -3.474e+00  2.742e-01 -12.669  < 2e-16 ***"
##  [8] "CreditScore      -4.595e-04  3.158e-04  -1.455   0.1457    "
##  [9] "Age               7.043e-02  2.853e-03  24.691  < 2e-16 ***"
## [10] "Tenure           -1.562e-02  1.046e-02  -1.493   0.1353    "
## [11] "Balance           2.503e-06  5.765e-07   4.343 1.41e-05 ***"
## [12] "NumOfProducts    -1.012e-01  5.279e-02  -1.917   0.0553 .  "
## [13] "EstimatedSalary   3.976e-07  5.284e-07   0.752   0.4518    "
## [14] "GeographyGermany  7.938e-01  7.571e-02  10.485  < 2e-16 ***"
## [15] "GeographySpain    6.230e-02  7.907e-02   0.788   0.4308    "
## [16] "GenderMale       -5.135e-01  6.089e-02  -8.433  < 2e-16 ***"
## [17] "HasCrCard1       -3.148e-03  6.643e-02  -0.047   0.9622    "
## [18] "IsActiveMember1  -1.075e+00  6.450e-02 -16.668  < 2e-16 ***"
## [19] "---"
## [20] "Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1"
## [21] ""
## [22] "(Dispersion parameter for binomial family taken to be 1)"
## [23] ""
## [24] "    Null deviance: 8051.1  on 7988  degrees of freedom"
## [25] "Residual deviance: 6862.3  on 7977  degrees of freedom"
## [26] "AIC: 6886.3"
## [27] ""
## [28] "Number of Fisher Scoring iterations: 5"
## [29] ""
##
## $roc_plot
## function ()
## {
##     plot_ly(roc_data, x = ~FPR, y = ~TPR, type = "scatter", mode =
"lines",
##         color = ~Model) %>% layout(title = paste("ROC Curve Comparison
(AUC",
##         model_name, ":", round(model_auc, 3), ")"), xaxis = list(title =
"False Positive Rate"),
##         yaxis = list(title = "True Positive Rate"))
## }
## <environment: 0x1280bbc80>
```

```
##
## $plot_list
## $plot_list$quant_plots
## $plot_list$quant_plots$CreditScore
##
## $plot_list$quant_plots$Age
##
## $plot_list$quant_plots$Tenure
##
## $plot_list$quant_plots$Balance
##
## $plot_list$quant_plots$NumOfProducts
##
## $plot_list$quant_plots$EstimatedSalary
##
##
## $plot_list$qulitat_plot
## $plot_list$qulitat_plot$Geography
##
## $plot_list$qulitat_plot$Gender
##
## $plot_list$qulitat_plot$HasCrCard
##
## $plot_list$qulitat_plot$IsActiveMember
##
## $plot_list$qulitat_plot$Exited
```

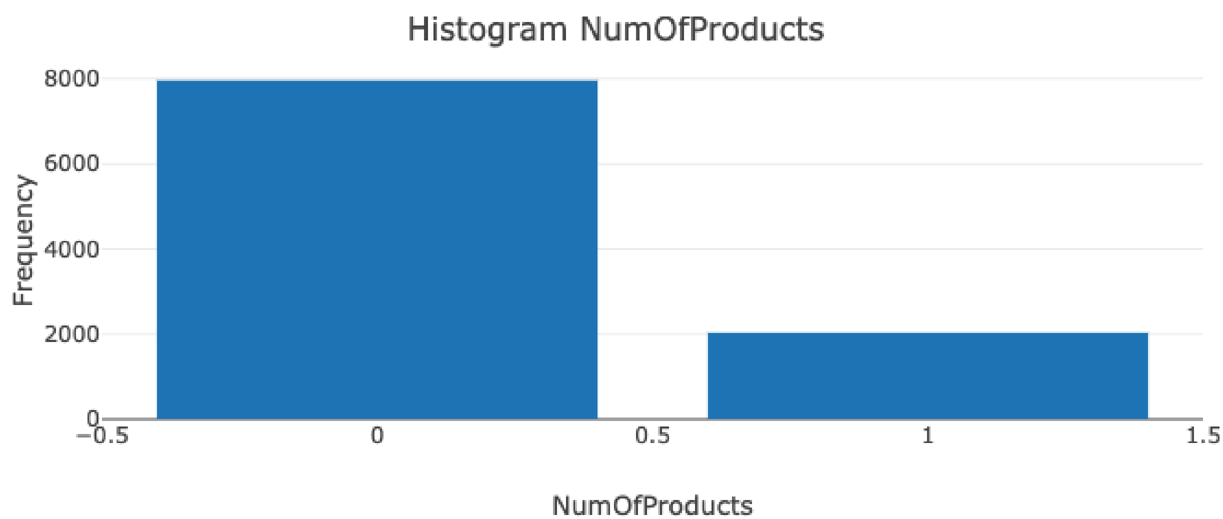

Histogram CreditScore
Frequency
8000
6000
4000
2000
0
−0.5
0
0.5
1
1.5
CreditScore


Histogram Age
Frequency
8000
6000
4000
2000
0
−0.5
0
0.5
1
1.5
Age

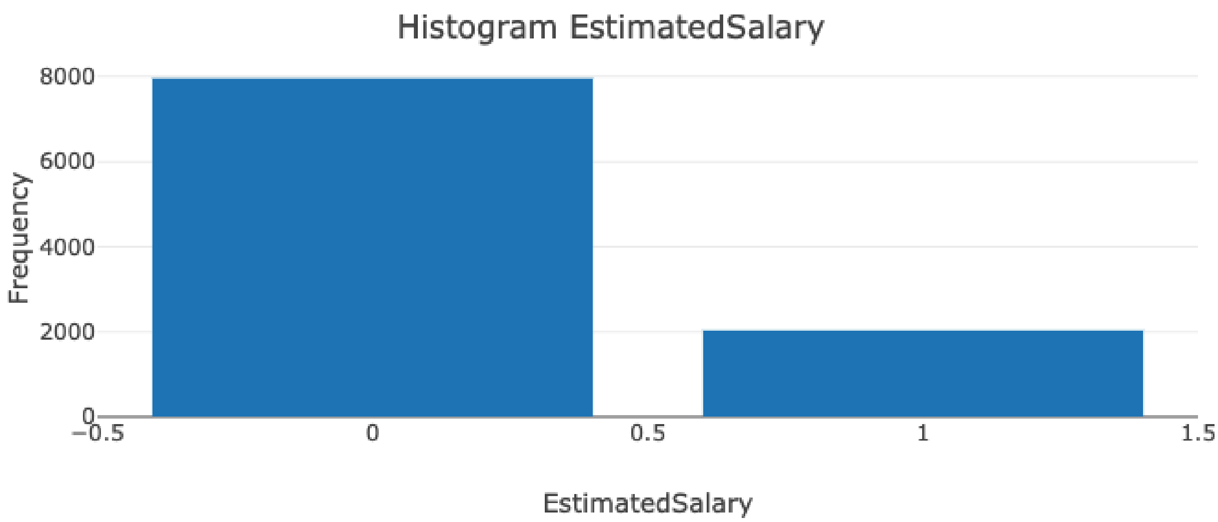

Histogram Tenure
Frequency
8000
6000
4000
2000
0
−0.5
0
0.5
1
1.5
Tenure



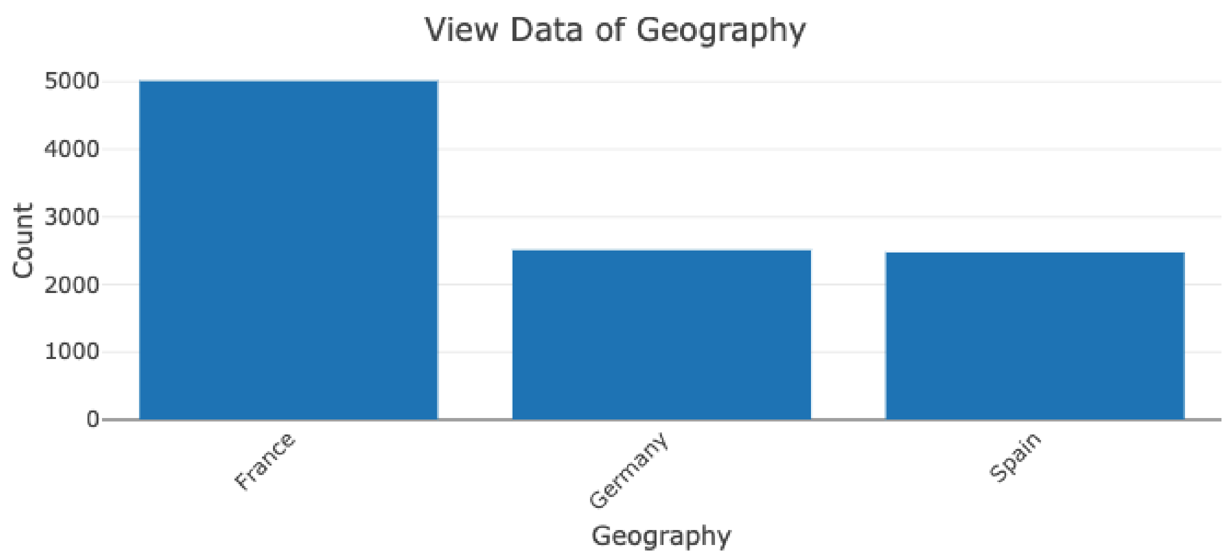
Histogram Balance
Frequency
8000
6000
4000
2000
0
−0.5
0
0.5
1
1.5
Balance

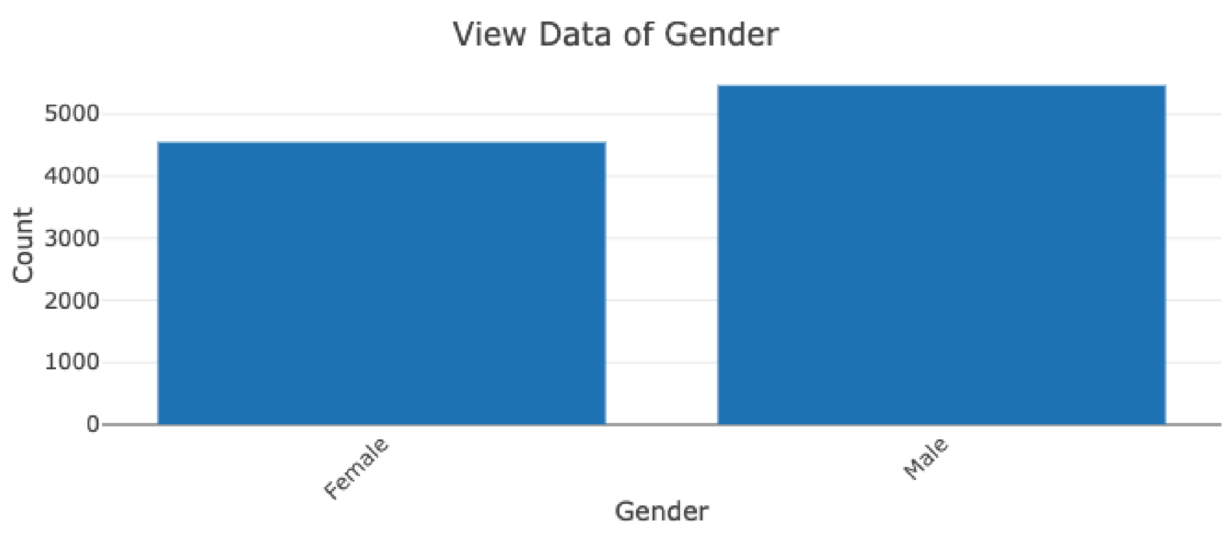


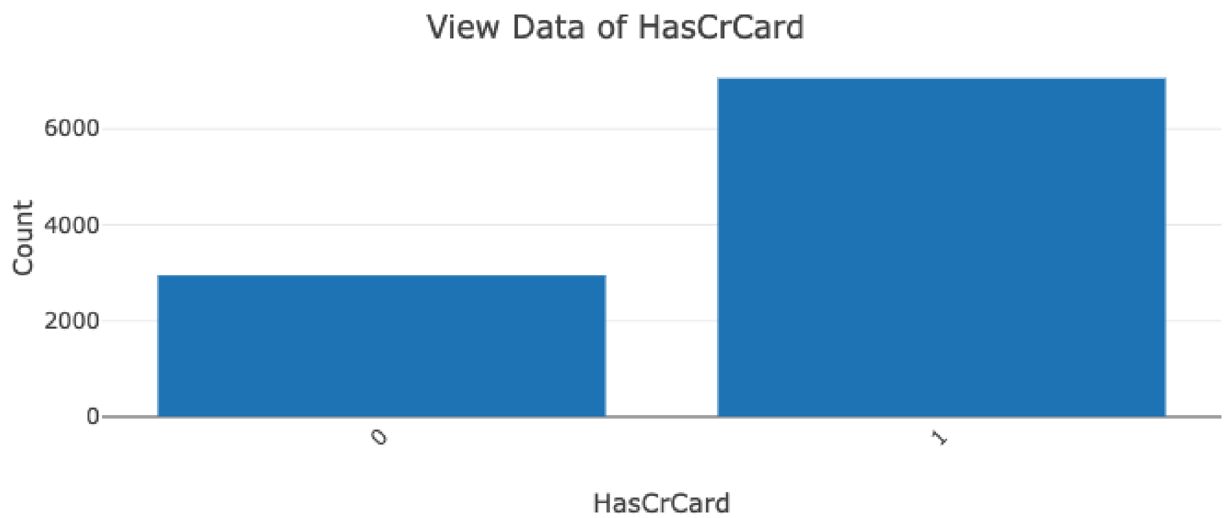
Histogram NumOfProducts
Frequency
8000
6000
4000
2000
0
−0.5
0
0.5
1
1.5
NumOfProducts

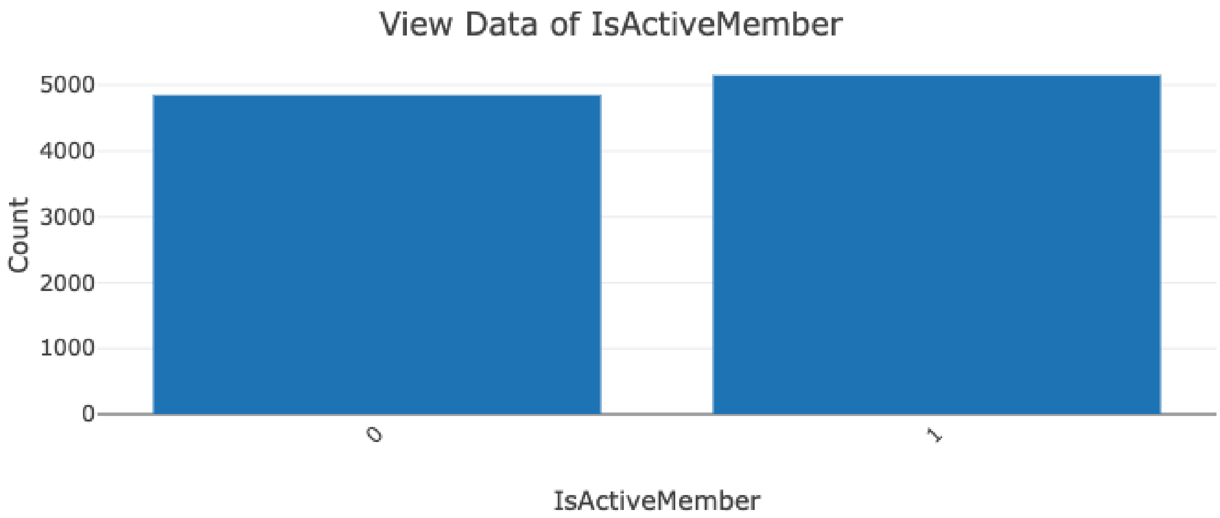


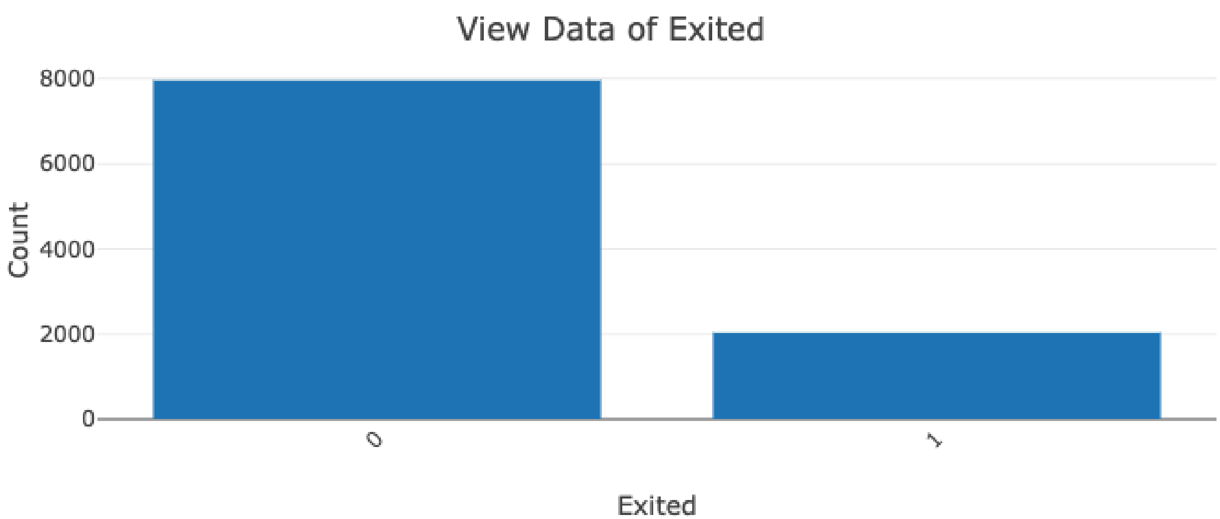
Histogram EstimatedSalary
8000
6000
4000
2000
0
Frequency
−0.5
0
0.5
1
1.5
EstimatedSalary


View Data of Geography
5000
4000
3000
2000
1000
0
Count
France
Germany
Spain
Geography


View Data of Gender
5000
4000
3000
2000
1000
0
Count
Female
Male
Gender


View Data of HasCrCard
Count
6000
4000
2000
0
0
1
HasCrCard


View Data of IsActiveMember
5000
4000
3000
2000
1000
0
Count
0
1
IsActiveMember


View Data of Exited
8000
6000
4000
2000
0
Count
0
1
Exited

Task 3.6

3.6.1 Shiny app implementation

```
ui <- fluidPage(
  titlePanel("Auto Model Selection Dashboard"),


  sidebarLayout(
    sidebarPanel(
      fileInput("datafile", "Upload CSV", accept = ".csv"),
      selectInput("target_var", "Select Response Variable", choices = NULL),
      numericInput("highest_category_count", "Max Category Count", value =
10, min = 2),
      actionButton("run_model", "Run Best Model")
    ),

    mainPanel(
      fluidRow(
        column(width = 12,
           wellPanel(
                 h3("Quantitative Features Plot List"),
                 uiOutput("quant_plots")
           )
        )
      ),
    fluidRow(
        column(width = 12,
           wellPanel(
                 h3("Qualitative Features Plot List"),
                 uiOutput("qulitat_plot")
           )
        )
      ),
      # First row: Model summary and model results side-by-side
      fluidRow(
        column(width = 6,
               wellPanel(
                 h3("Model Summary"),
```

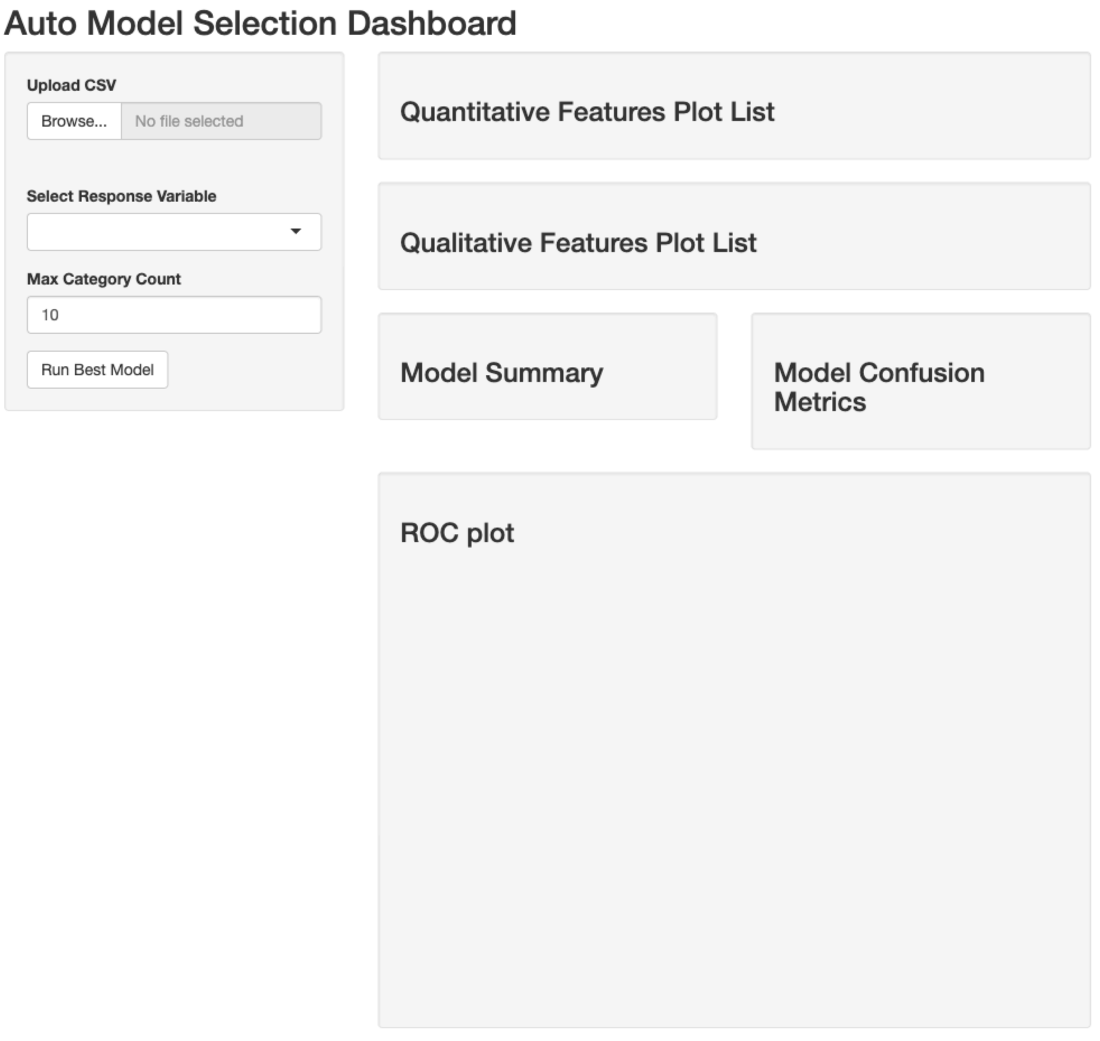
```
                 verbatimTextOutput("model_summary")
               )
        ),
        column(width = 6,
               wellPanel(
                 h3("Model Confusion Metrics"),
                 DTOutput("confMat")
               )
        )
      ),
      # Second row: Preprocessing plots full width
      fluidRow(
        column(width = 12,
               wellPanel(
                 h3("ROC/Observed vs. Predicted plot"),
                 plotlyOutput("rocPlot")
               )
        )
      )
    )
  )
)

server <- function(input, output, session) {
  # Reactive: Load data from uploaded CSV file
  upload_dataset <- reactive({
    req(input$datafile)
    read.table(input$datafile$datapath,
               sep = ",",
               header = TRUE,
               quote = "\"",
               stringsAsFactors = FALSE,
               na.strings = c("", "NA"))
  })

  # Update target variable choices once the dataset is loaded
  observe({
    req(upload_dataset())
    updateSelectInput(session, "target_var", choices =
names(upload_dataset()))
    updateSelectInput(session, "highest_category_count", choices =
names(upload_dataset()))
```

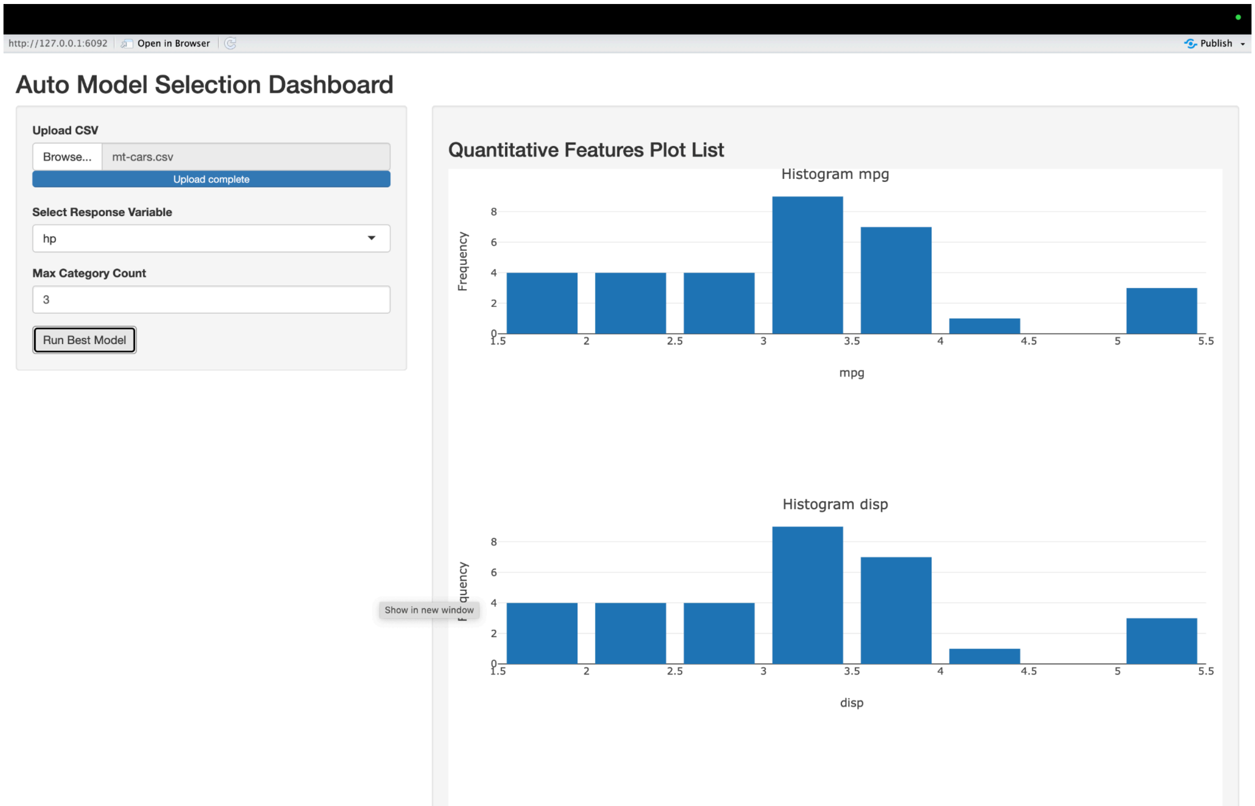
```
  })

  observe({
   req(results())
   qulitat_plots <- results()$plot_list$qulitat_plot
    lapply(1:length(qulitat_plots), function(i) {
      output[[ paste("qulitat_plot", i, sep = "") ]] <- renderPlotly({
        qulitat_plots[[i]]
      })
    })
  })

  observe({
   req(results())
   quant_plots <- results()$plot_list$quant_plots
    lapply(1:length(quant_plots), function(i) {
      output[[ paste("quant_plot_", i, sep = "") ]] <- renderPlotly({
        quant_plots[[i]]
      })
    })
  })

  # Run model when the "Run Best Model" button is clicked
  results <- eventReactive(input$run_model, {
    req(upload_dataset(), input$target_var)
    select_and_run_best_model(input$target_var, upload_dataset(),
input$highest_category_count)
  })

  # Display model summary output (best model info)
    output$model_summary <- renderPrint({
    req(results())
    list(
    Model_Type = results()$model_type,
    Performance = results()$performance,
    Model_Summary = results()$model_summary
    )
  })

  output$confMat <- renderDT({
    req(results())
    if (results()$response_type == "binary") {
      as.data.frame(results()$confusion_matrix$table)
    }
```

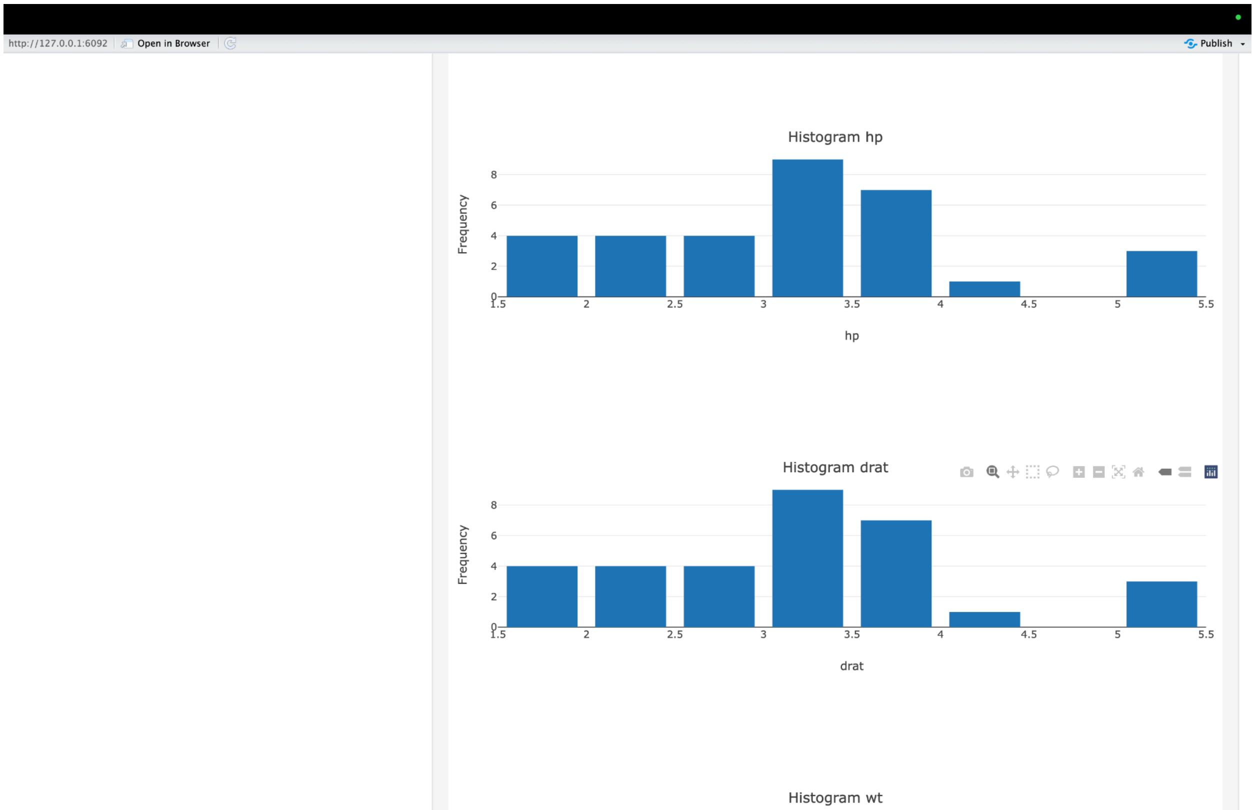
```
  })

  output$rocPlot <- renderPlotly({
    req(results())
    if (results()$response_type == "binary") {
      results()$roc_plot()
    } else {
      results()$pred_vs_real()
    }
  })

output$quant_plots <- renderUI({
  req(results())
  quant_plots <- results()$plot_list$quant_plots
  lapply(1:length(quant_plots), function(i) {
      plotlyOutput(outputId = paste("quant_plot_", i, sep = ""))
    })
})

output$qulitat_plot <- renderUI({

  req(results())
  qulitat_plots <- results()$plot_list$qulitat_plot
  lapply(1:length(qulitat_plots), function(i) {
      plotlyOutput(outputId = paste("qulitat_plot", i, sep = ""))
    })

})


}


shinyApp(ui, server)

##
## Listening on http://127.0.0.1:8624
```

3.6.2 Shiny app dashboard before dataset insert

3.6.3 Shiny app dashboard after mt-cars dataset upload & select numerical feature as response variable




http://127.0.0.1:6092
Open in Browser
Publish
Auto Model Selection Dashboard
Upload CSV
Browse...
mt-cars.csv
Upload complete
Select Response Variable
hp
Max Category Count
3
Run Best Model
Quantitative Features Plot List
Histogram mpg
Frequency
mpg
Histogram disp
Show in new window
disp


http://127.0.0.1:6092
Open in Browser
Publish
Histogram hp
Frequency
8
6
4
2
0
1.5
2
2.5
3
3.5
4
4.5
5
5.5
hp
Histogram drat
Frequency
8
6
4
2
0
1.5
2
2.5
3
3.5
4
4.5
5
5.5
drat
Histogram wt

http://127.0.0.1:6092
Open in Browser
Publish
Histogram wt
Frequency
wt
Qualitative Features Plot List
View Data of cyl
Count
cyl

[redacted]

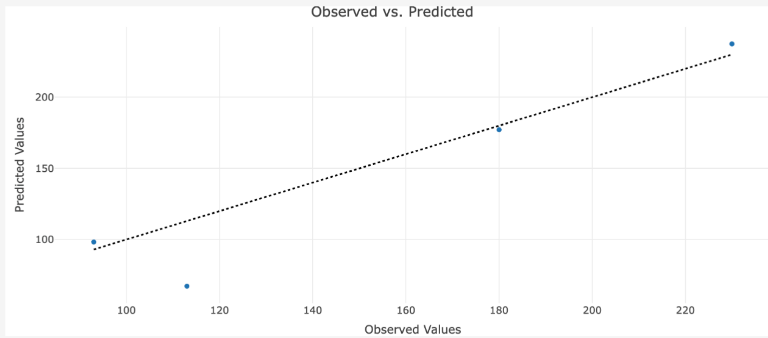
http://127.0.0.1:7935 | Open in Browser | Publish

Model Summary

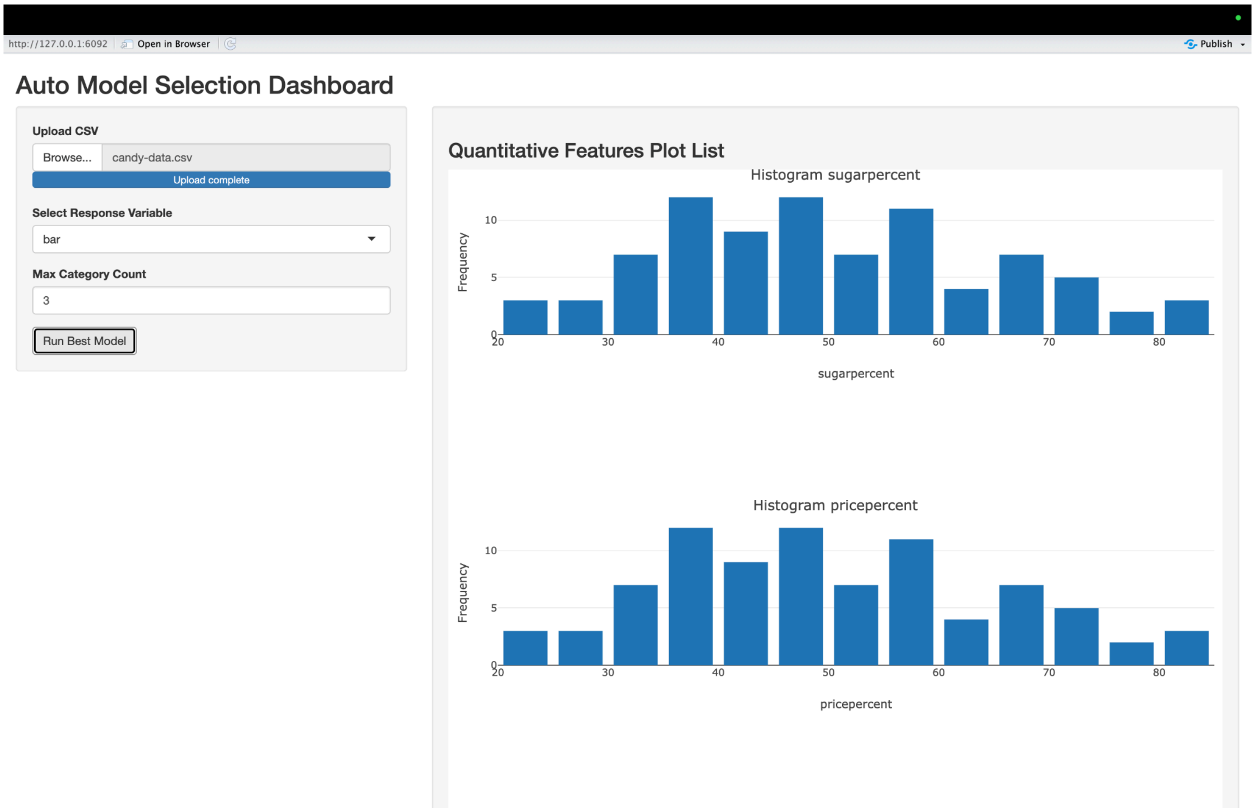
```
$Model_Type
[1] "RF"

$Performance
[1] "RMSE LM = 40.72 | RMSE RF = 23.39 | Selected: RF"

$Model_Summary
 [1] "                 Length Class      Mode     " "ca
 [4] "predicted        27    -none-     numeric  " "ms
 [7] "oob.times        27    -none-     numeric  " "im
[10] "localImportance   0    -none-     NULL     " "pr
[13] "mtry              1    -none-     numeric  " "fo
[16] "y                27    -none-     numeric  " "te
[19] "xNames            6    -none-     character" "pr
[22] "obsLevels         1    -none-     logical  " "pa
```

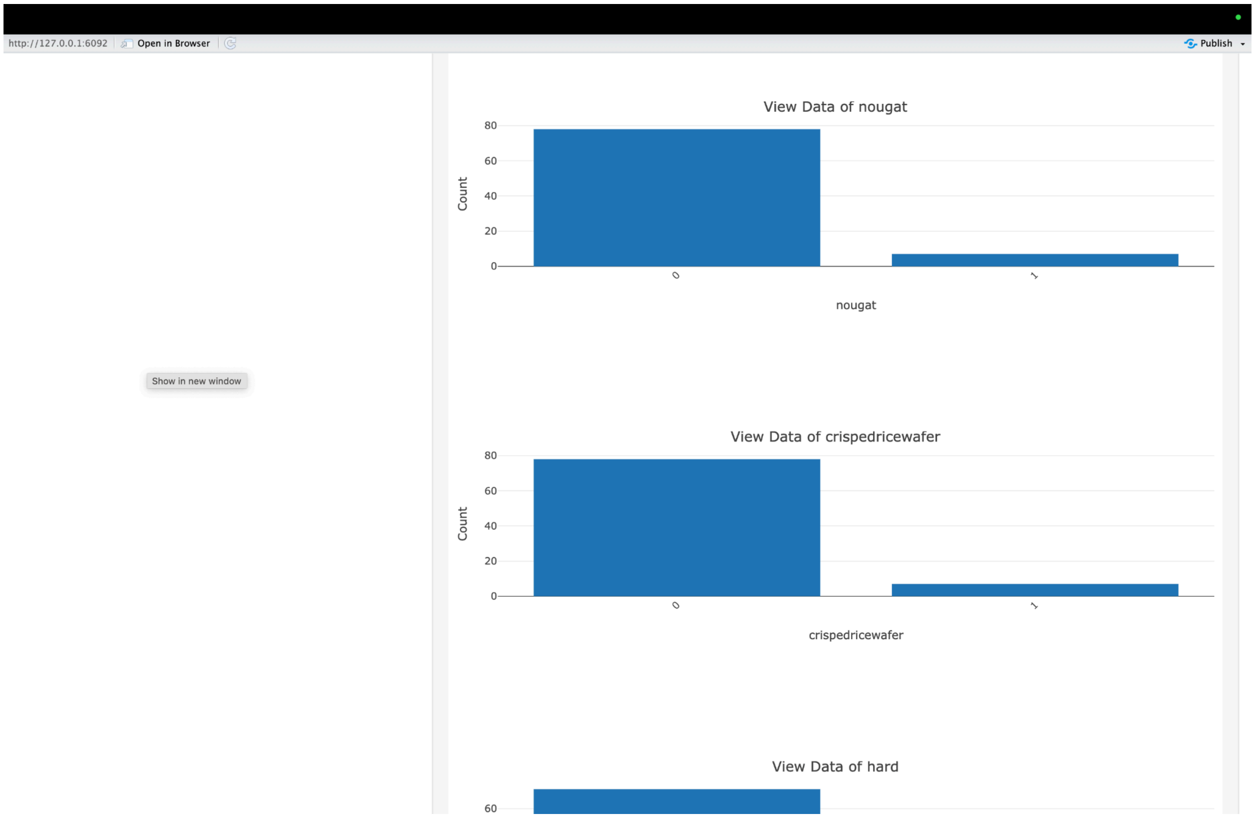
Model Confusion Metrics

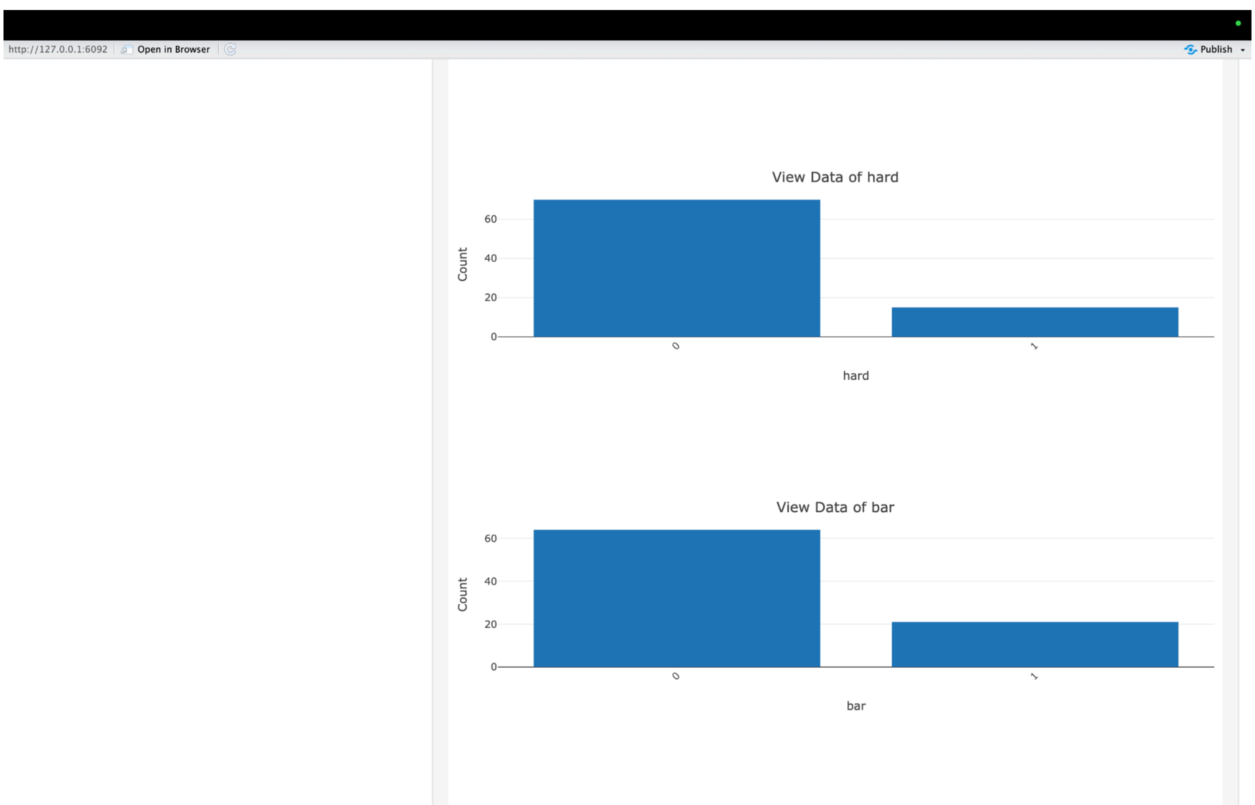
ROC/Observed vs. Predicted plot



3.6.4 Shiny app dashboard after candy data set dataset upload & select binary response variable




http://127.0.0.1:6092
Open in Browser
Publish
Show in new window
View Data of nougat
Count
80
60
40
20
0
0
1
nougat
View Data of crispedricewafer
Count
80
60
40
20
0
0
1
crispedricewafer
View Data of hard
60


http://127.0.0.1:6092
Open in Browser
Publish
View Data of hard
Count
60
40
20
0
0
1
hard
View Data of bar
Count
60
40
20
0
0
1
bar


http://127.0.0.1:6092
Open in Browser
Publish
View Data of bar
Count
60
40
20
0
0
1
bar
View Data of pluribus
Count
40
30
20
10
0
0
1
pluribus

http://127.0.0.1:6092 Open in Browser Publish

Model Summary

```
$Model_Type
[1] "RF"

$Performance
[1] "AUC RF = 0.979"

$Model_Summary
 [1] "                 Length Class  Mode     " "call
 [4] "predicted         69   factor numeric  " "err.ra
 [7] "votes            138   matrix numeric  " "oob.ti
[10] "importance        10   -none- numeric  " "import
[13] "proximity          0   -none- NULL     " "ntree
[16] "forest            14   -none- list     " "y
[19] "inbag              0   -none- NULL     "
```

Model Confusion Metrics

Show 10 entries Search:

	Prediction	Reference	Freq
1	0	0	11
2	1	0	1
3	0	1	0
4	1	1	4

Showing 1 to 4 of 4 entries Previous 1 Next

ROC plot