CREDIT RISK & ACQUISITION

AGENDA

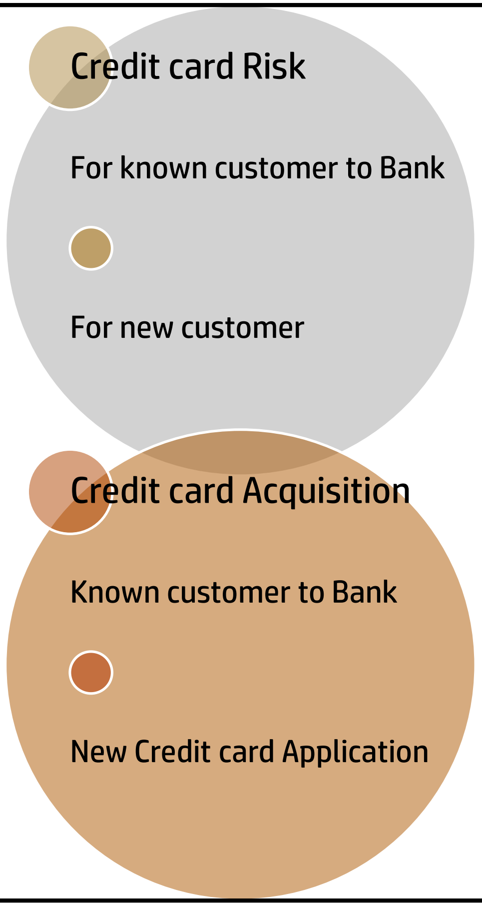
- BUSINESS PROBLEM & OBJECTIVES
- STRATEGIC APPROACH TO THE PROBLEM
- PLANNING & EXECUTION STEPS
- EXPLORATORY DATA ANALYSIS
- MODEL BUILDING, EVALUATION, MODEL SELECTION, BENEFITS TO THE BANK
- CREDIT RISK & ACQUISITION – ACHIEVEMENTS
- CREDIT RISK & ACQUISITION – APPENDIX

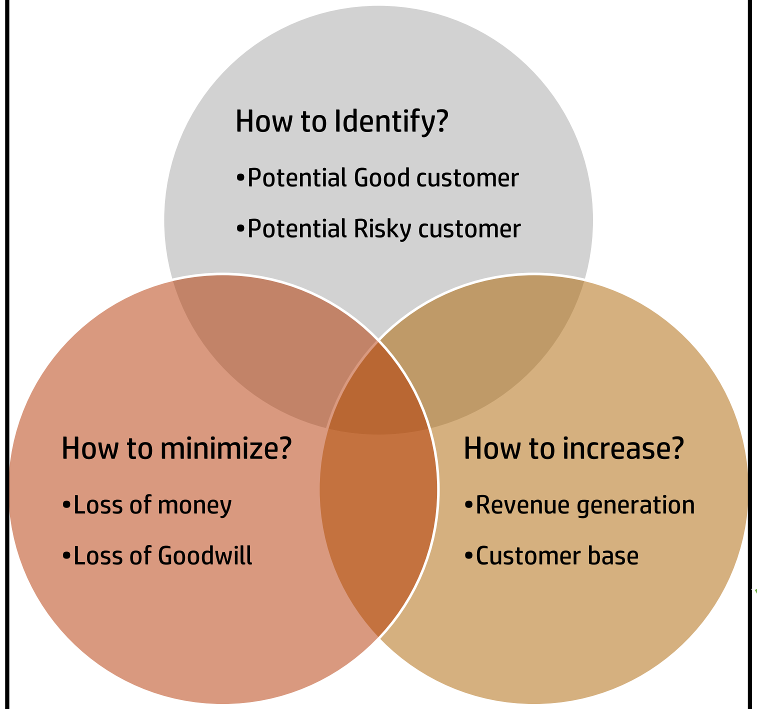
Presented By:

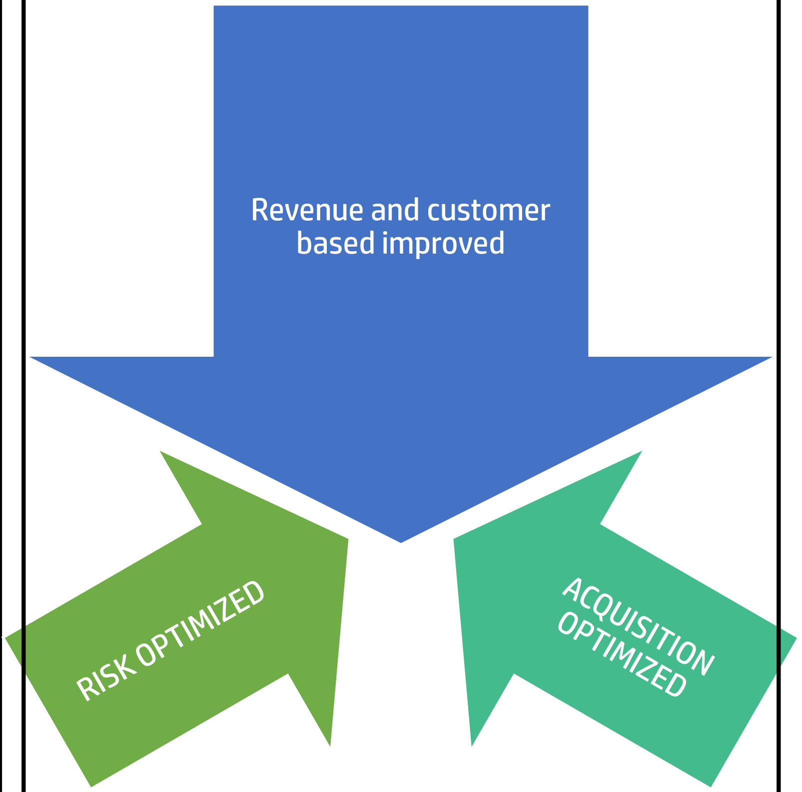
Subhra Sinha
Indira Kumari
Rohit Ayinaparthy
Amitava Mukherjee

CREDIT RISK & ACQUISITION – **BUSINESS PROBLEM & OBJECTIVES**

Situation to Handle | Challenges to Handle | Ideal Achievements







STRATEGIC APPROACH TO THE PROBLEM

Build Predictive Models

- *Focus:*
  - *Identify potential risky acquisition*
  - *Decide on how much risk can be taken*

Model Evaluations

- *Choose the best Model*
  - *Based on accuracy, sensitivity, specificity*
  - *Ensuring that Model is stable*
  - *Ensure that Model is not overfitting*

Minimize Revenue loss

- *Optimize acquisition*
- *Optimize risk*

Minimize Goodwill loss

- *Reduce operational cost*
- *Reduce verification process delay*

PLANNING & EXECUTION STEPS

- Understanding Data – Data Cleaning, Data Massaging
- EDA – Univariate, Bi-variate, Deriving calculated fields, WOE & IV Analysis
- Model Building & Evaluation – Logistic, Decision Tree and Random forest model building, Evaluating model, Identify alternate models
- Calculating scores for applications not known to the Model
- Calculating the loss of bank per month per 100 applications without having the Model
- Continuous improvement through Future cases and Feedbacks

PLANNING & EXECUTION STEPS

Data Quality Checks – Done
- See the Appendix

Data Massaging – Done
- See the Appendix

Analyzing Statistics - Done
- See the Appendix

Data Preparation for EDA – Done
- See the appendix

WOE and IV preparation and analysis- Done
- See the appendix

Data Balancing - Done
- See the Appendix

PLANNING & EXECUTION STEPS

Fine Tune Binning – Done

- **See the Appendix**

Model Buildings - Done

- **See the appendix**

Evaluate Models - Done

- **See the Appendix**

Unknown Application processing – Done

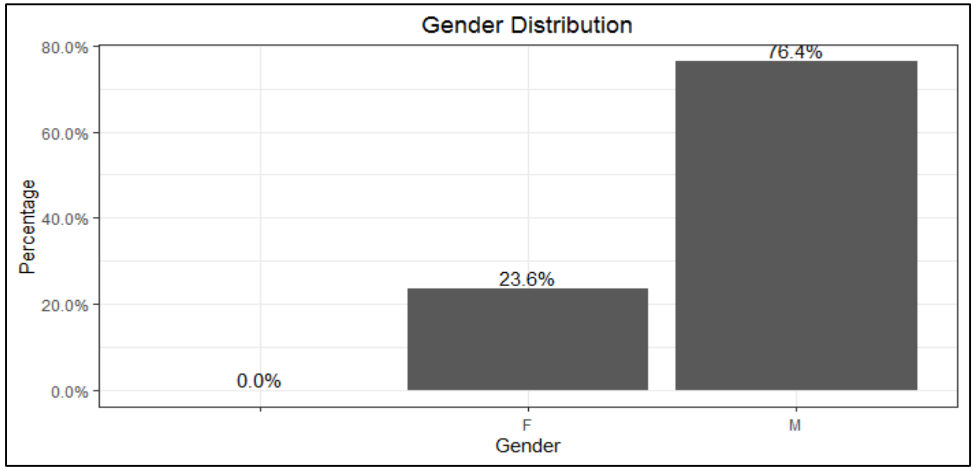
- **See the Appendix**

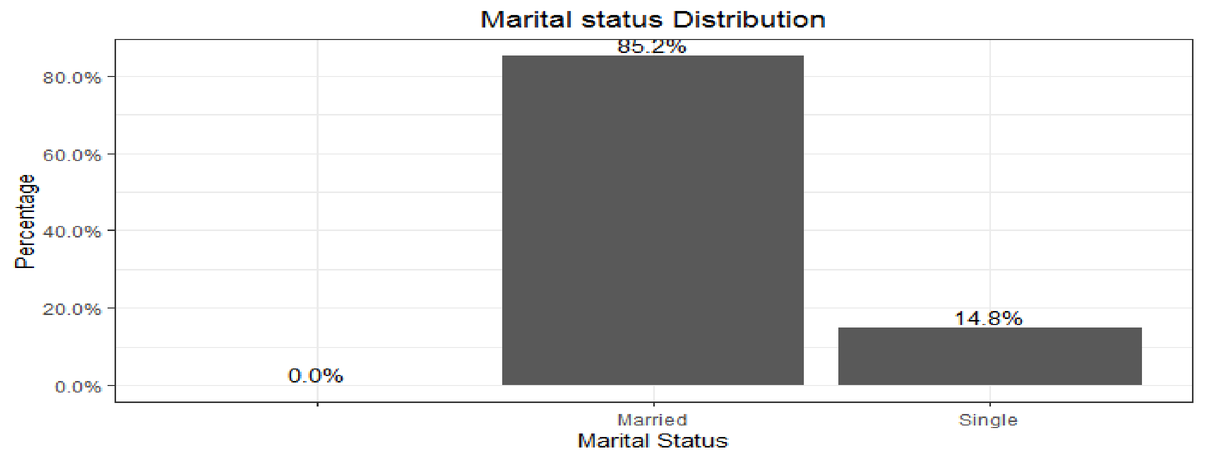
Decision Making – Done

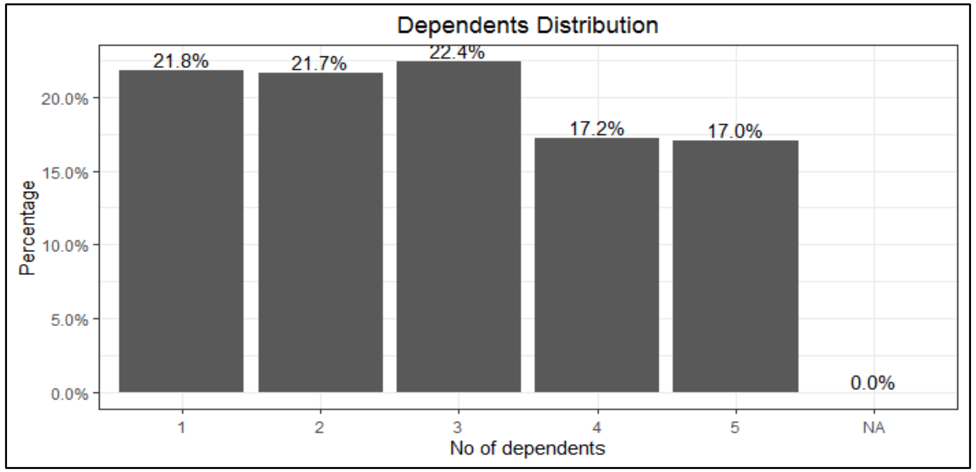
- **See the Appendix**

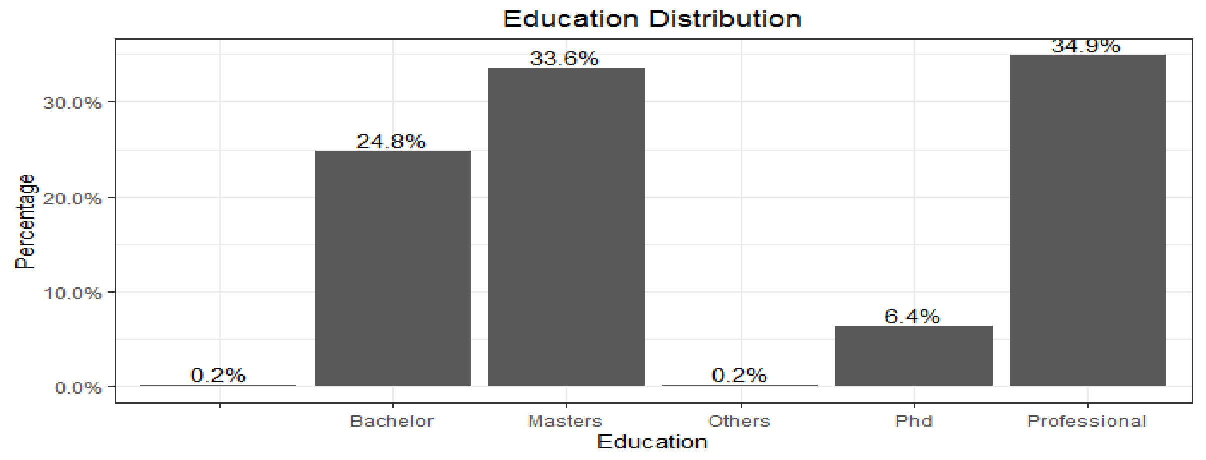
Future Planning

Exploratory Data Analysis – Univariate Analysis

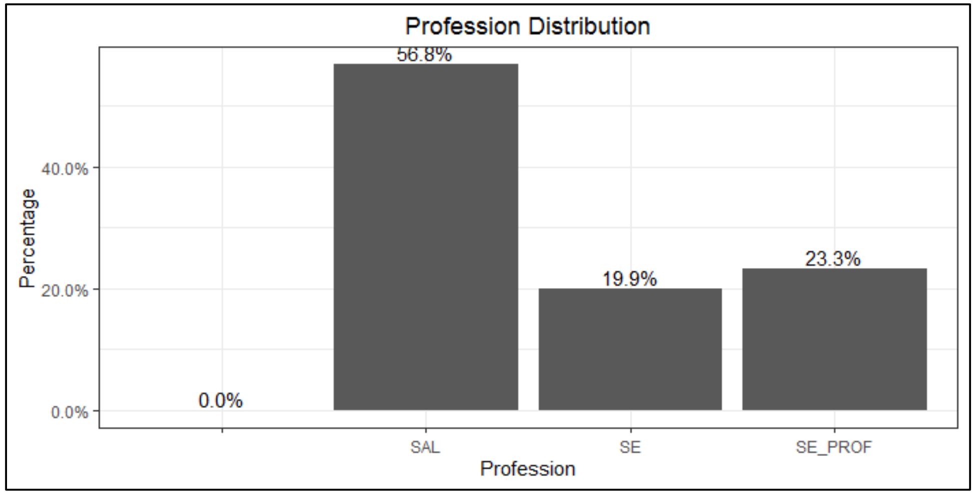


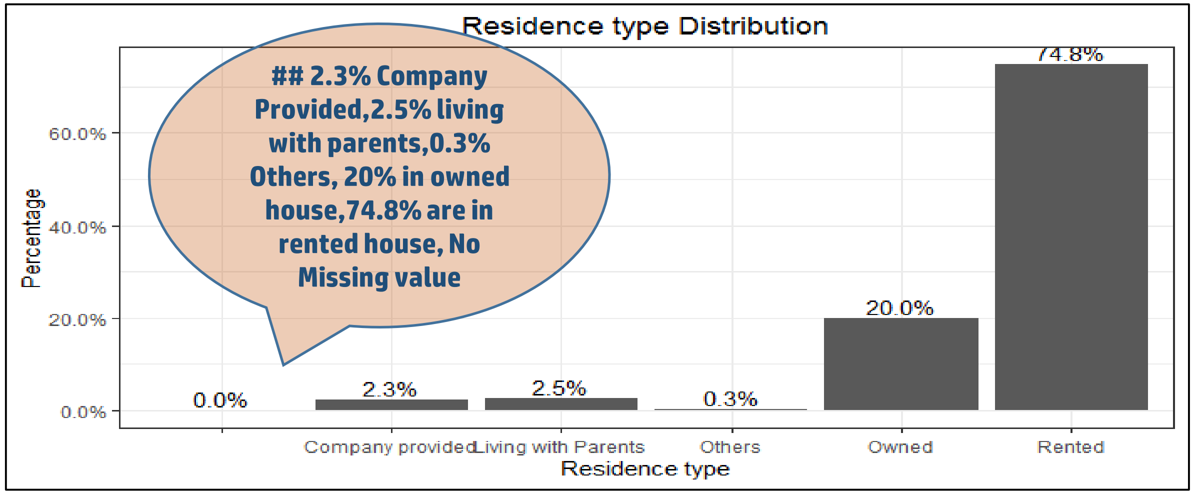


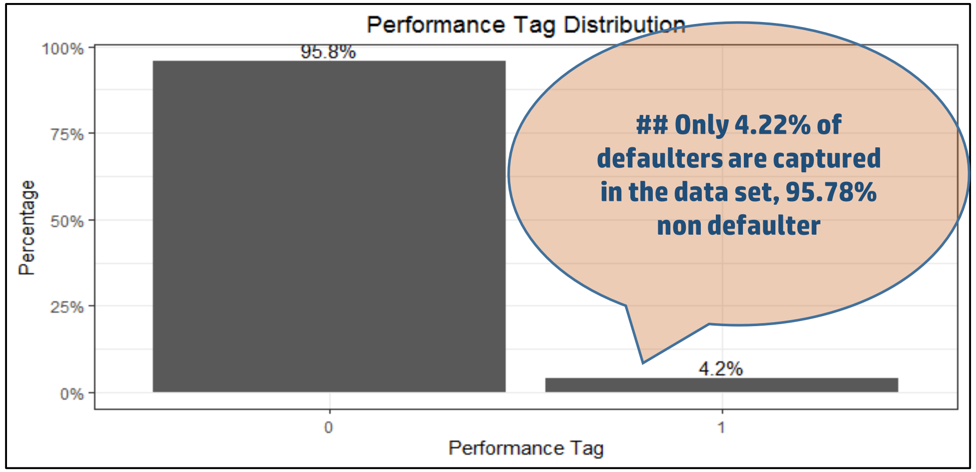


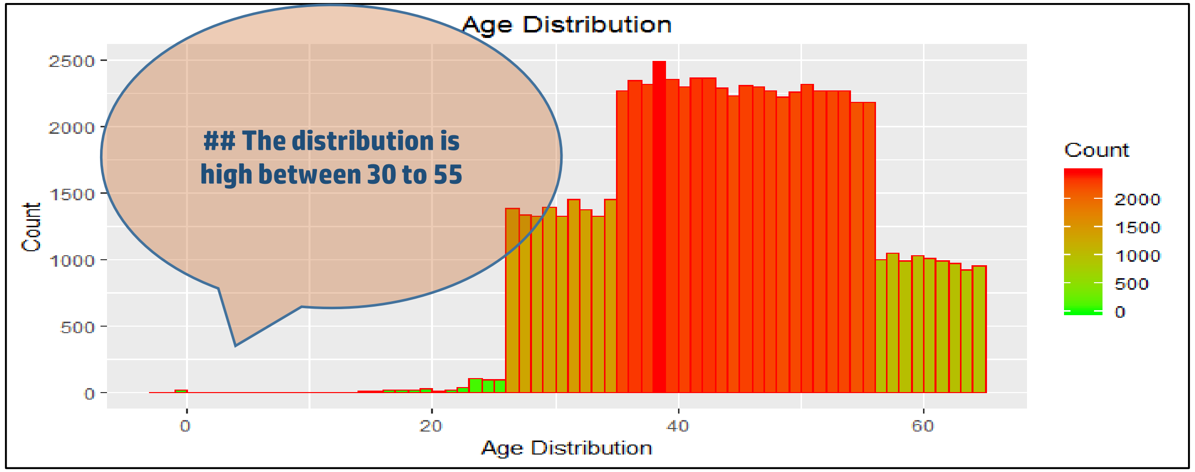


Exploratory Data Analysis – Univariate Analysis

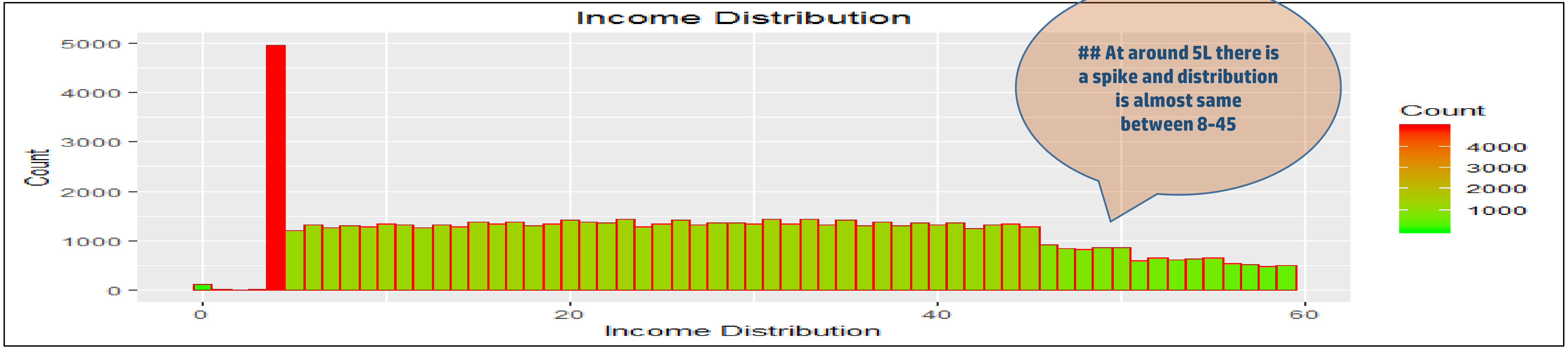


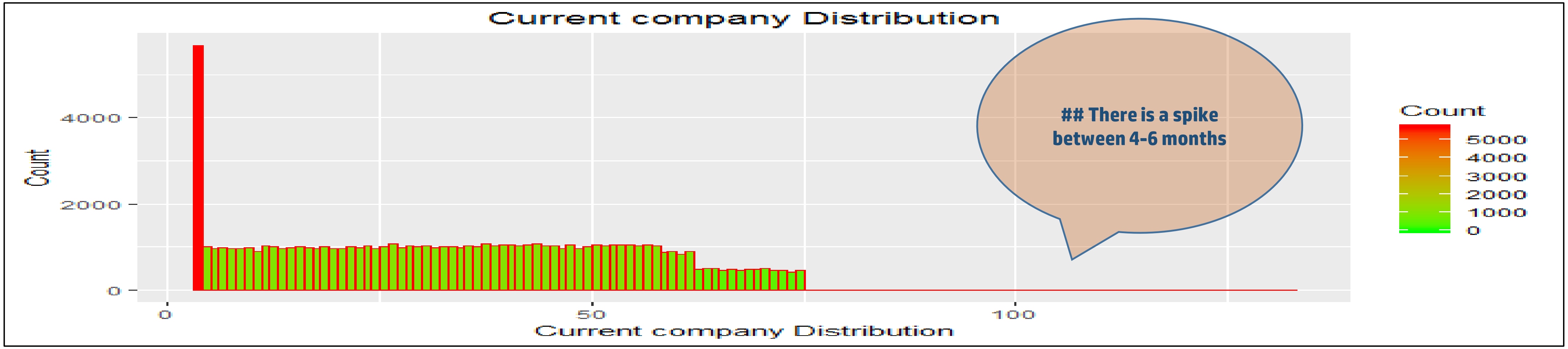




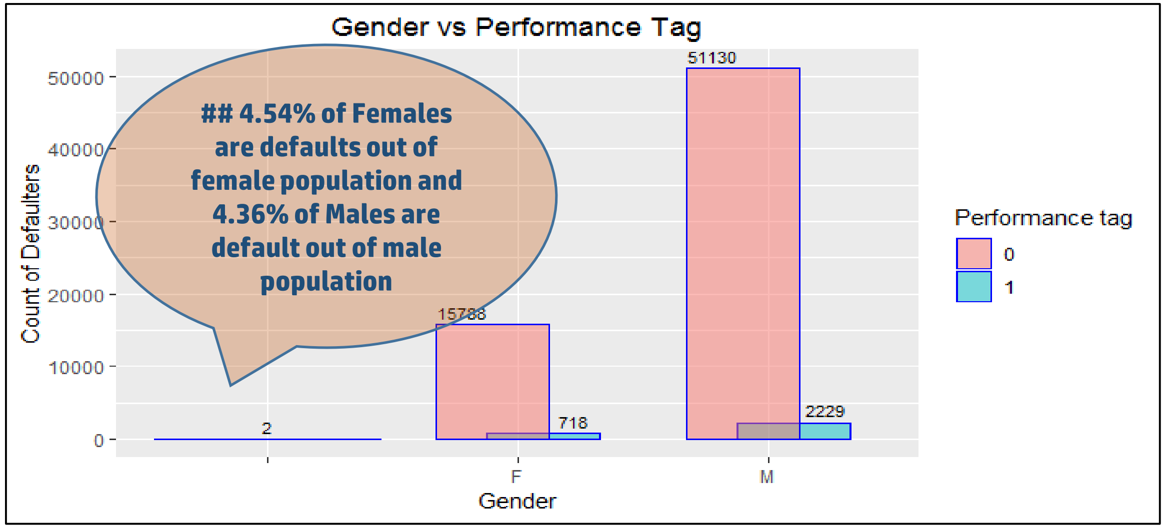


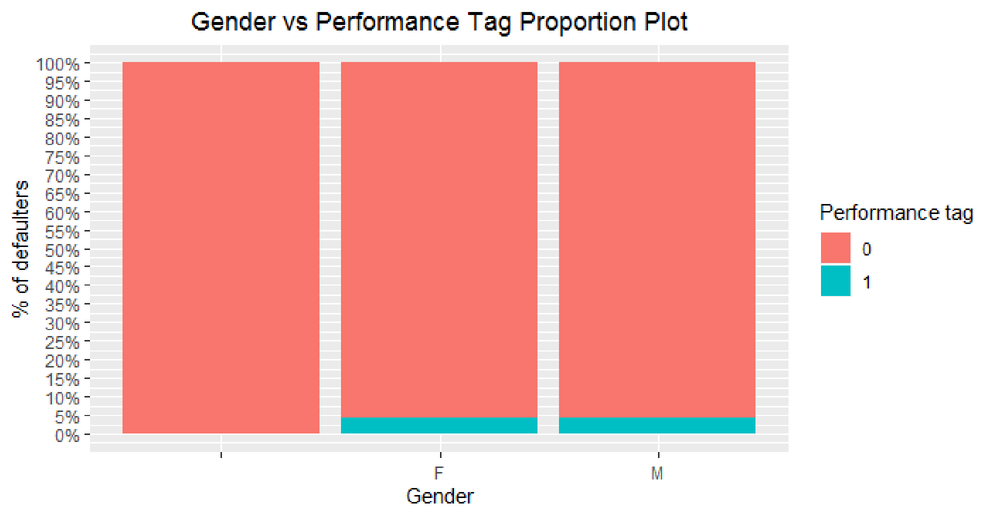
Exploratory Data Analysis – Univariate Analysis

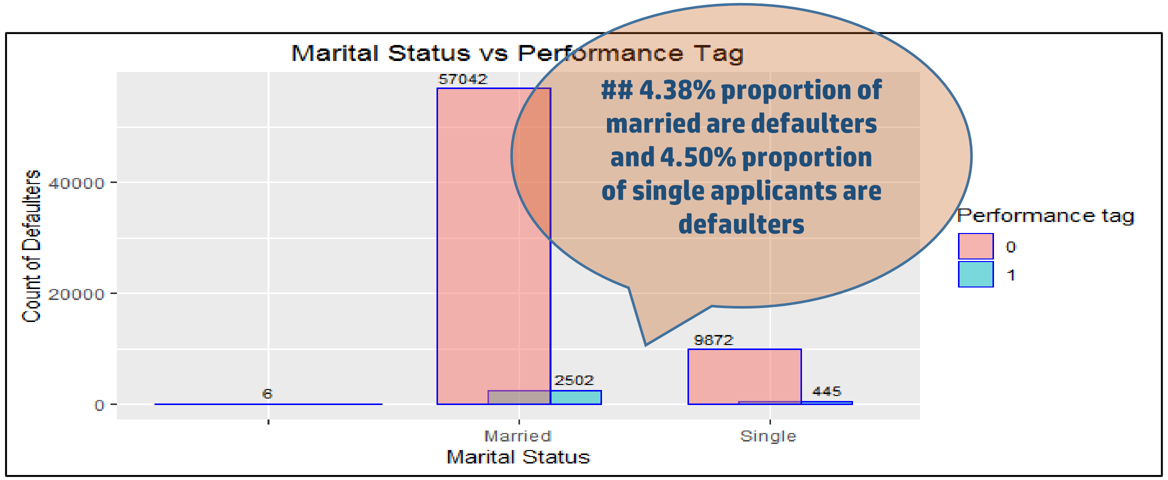


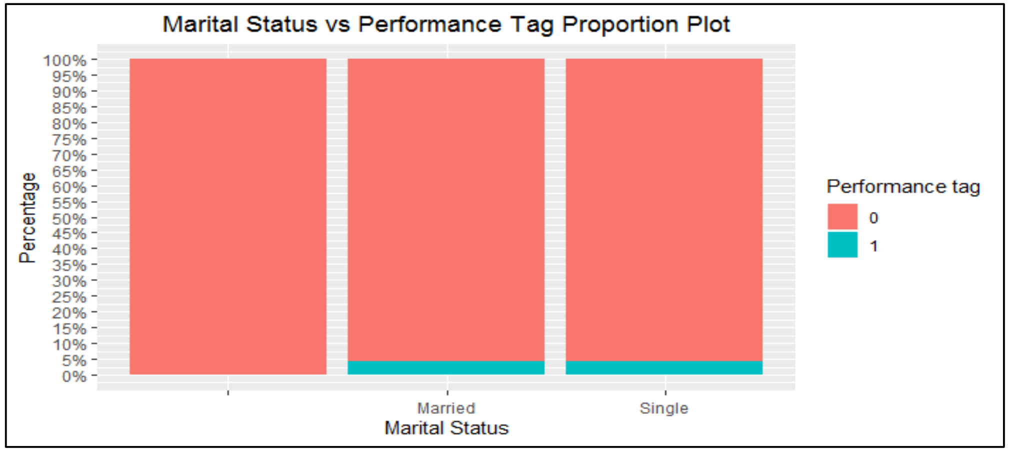


Exploratory Data Analysis – Bivariate Analysis

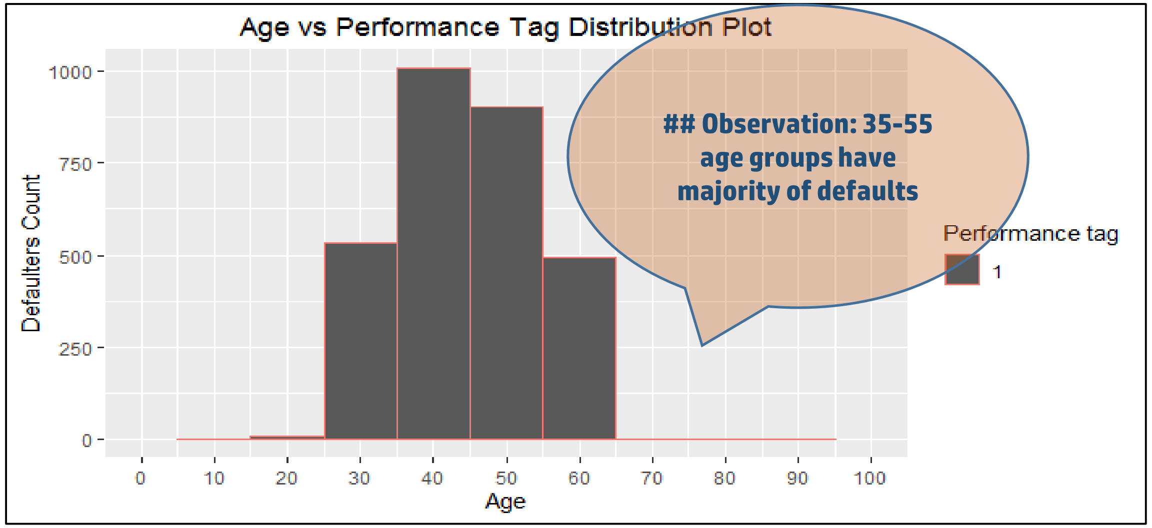


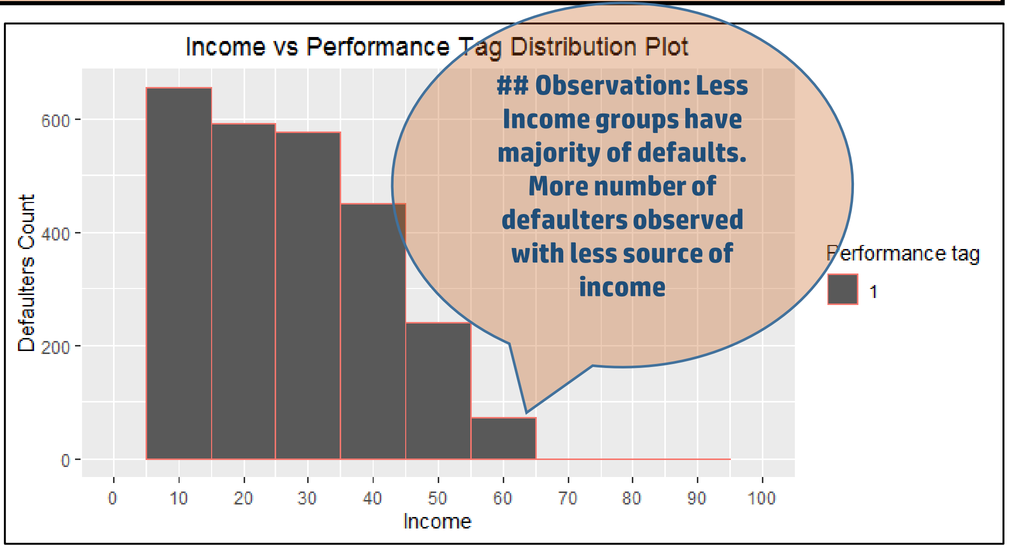


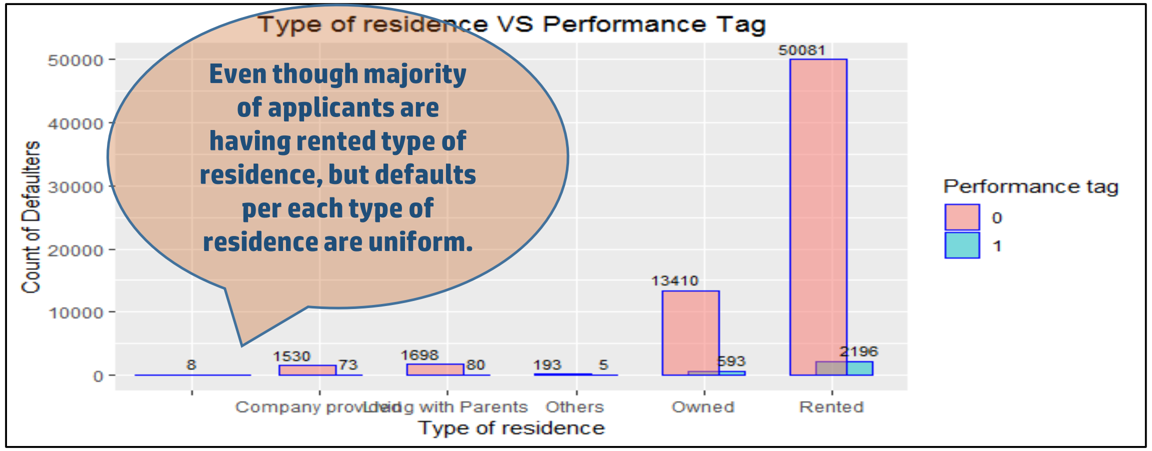


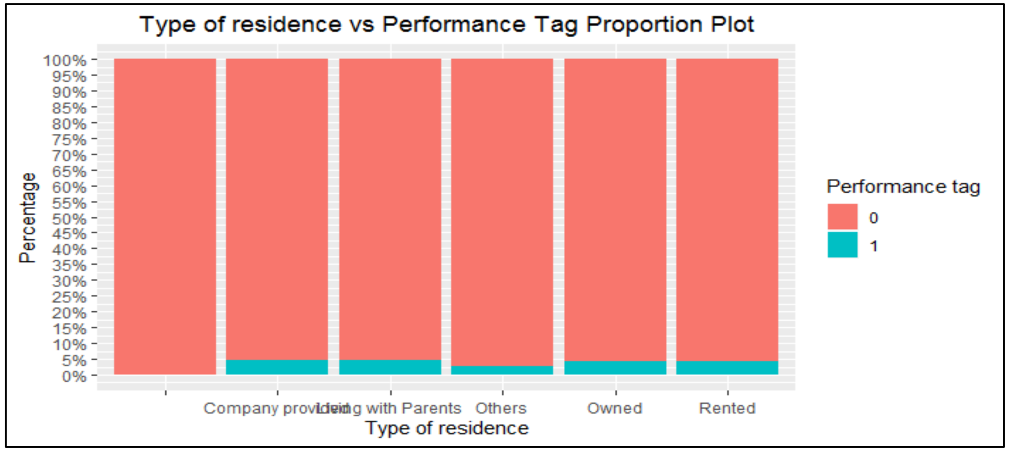


Exploratory Data Analysis – Bivariate Analysis

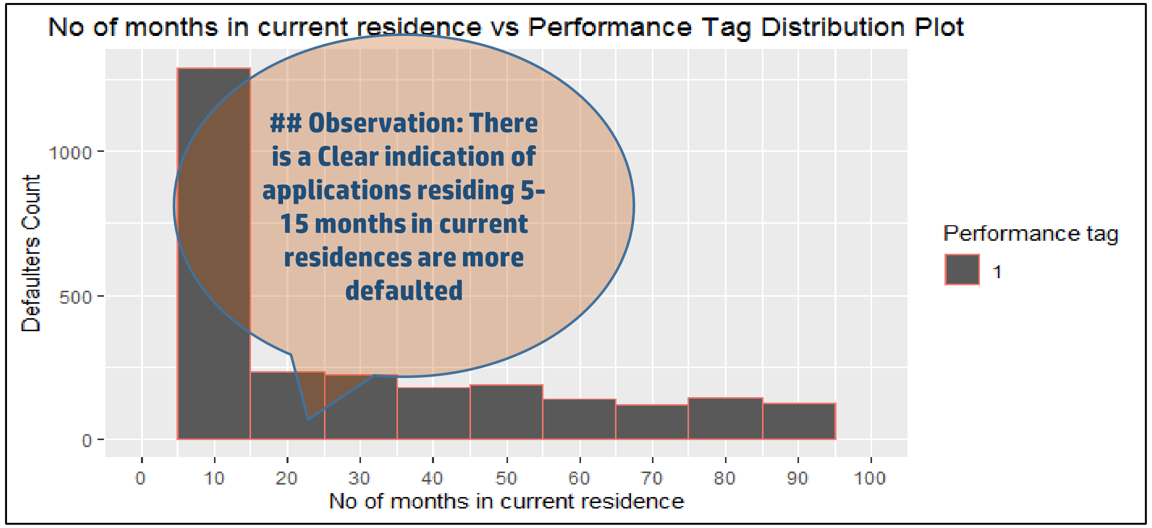


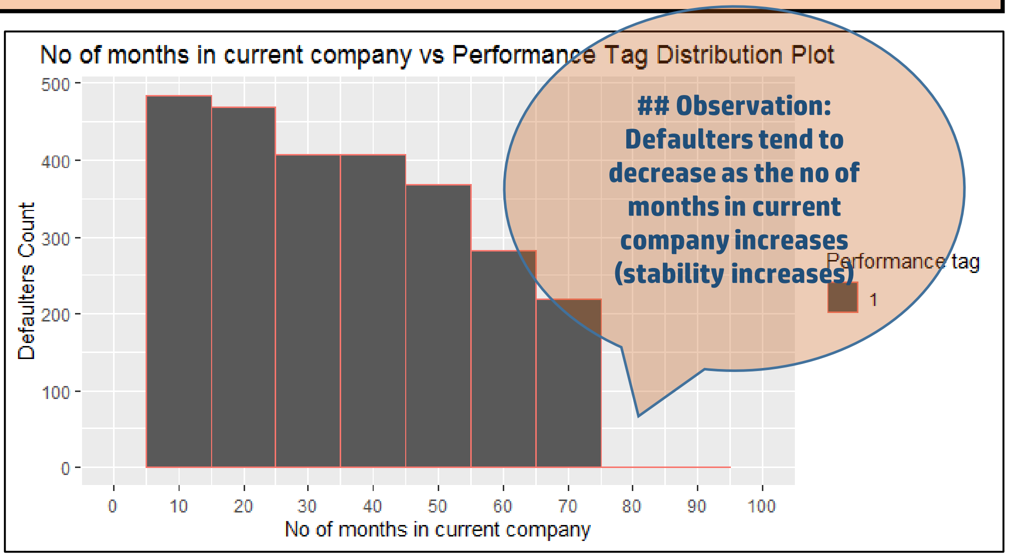


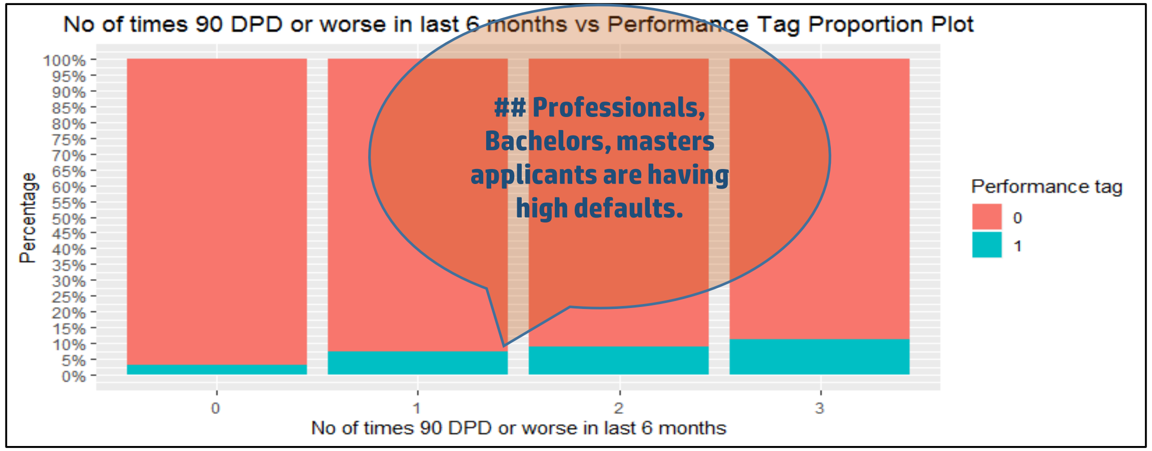


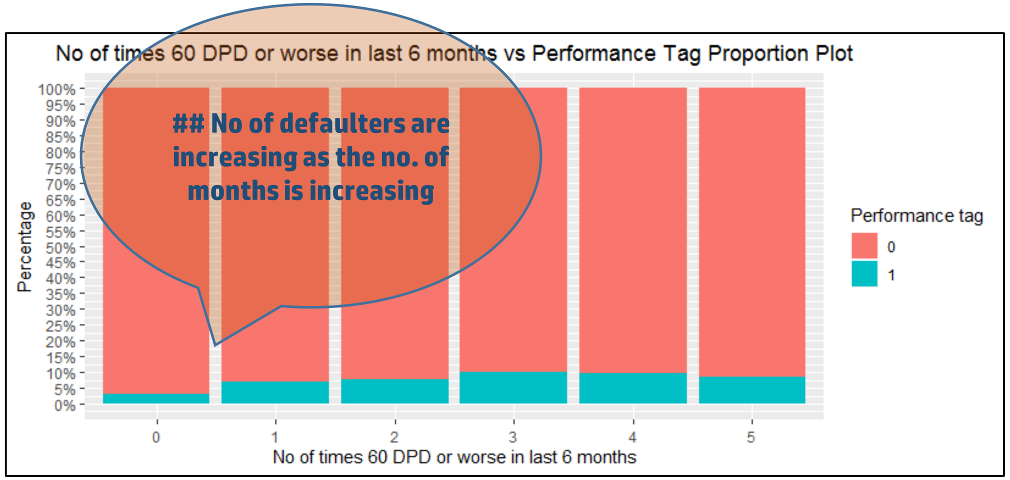


Exploratory Data Analysis – Bivariate Analysis

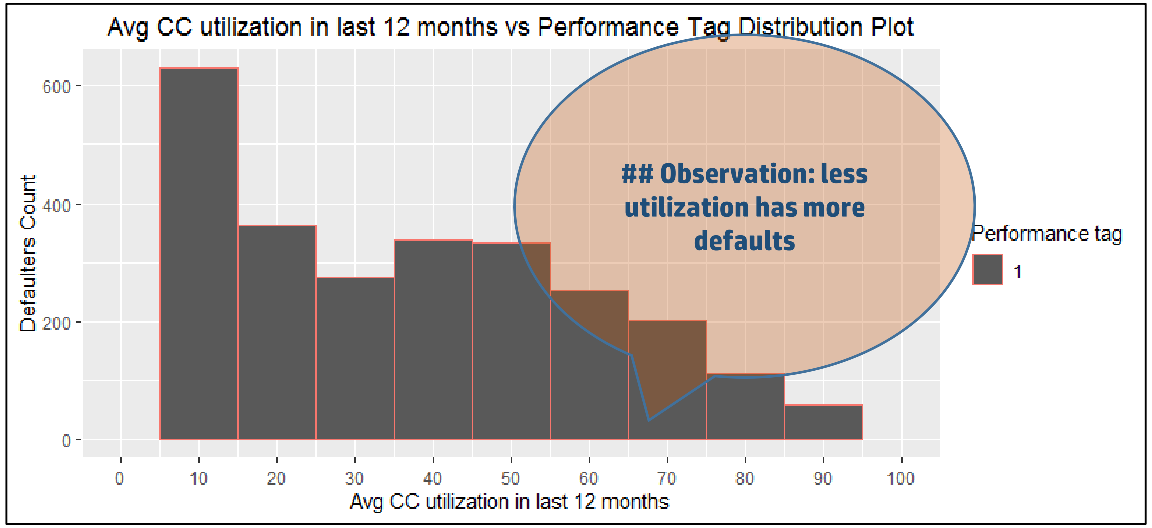


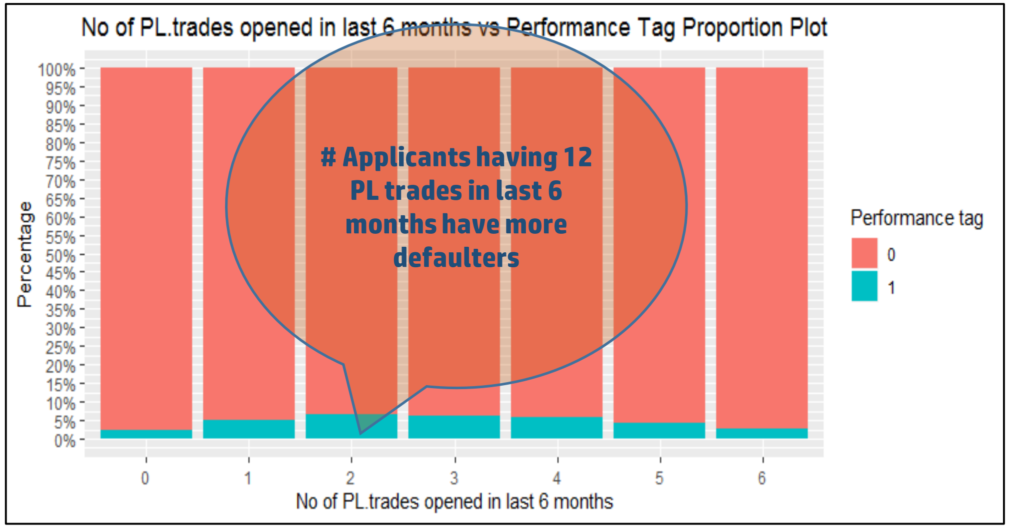


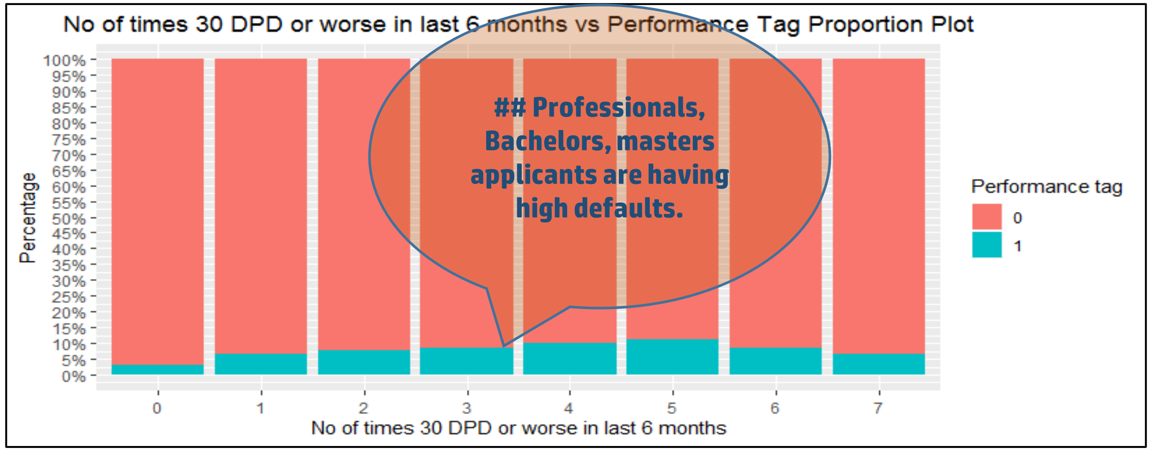


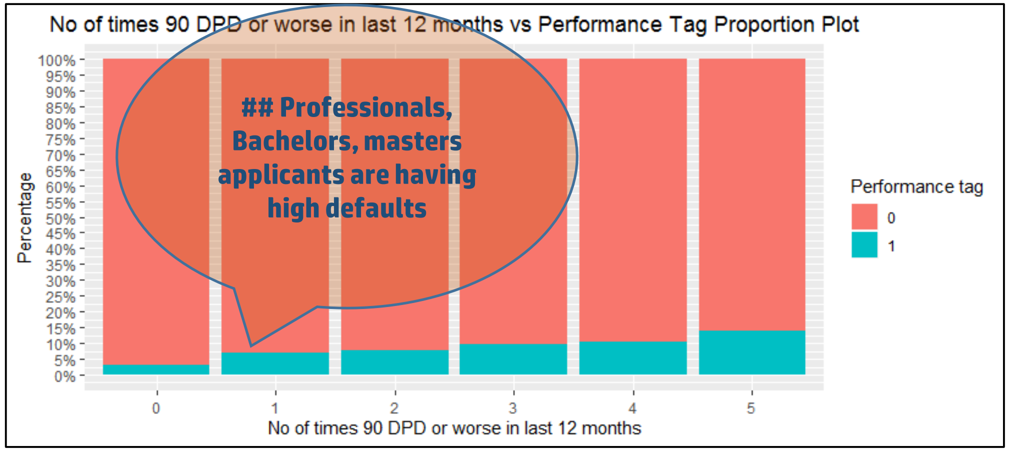


Exploratory Data Analysis – Bivariate Analysis

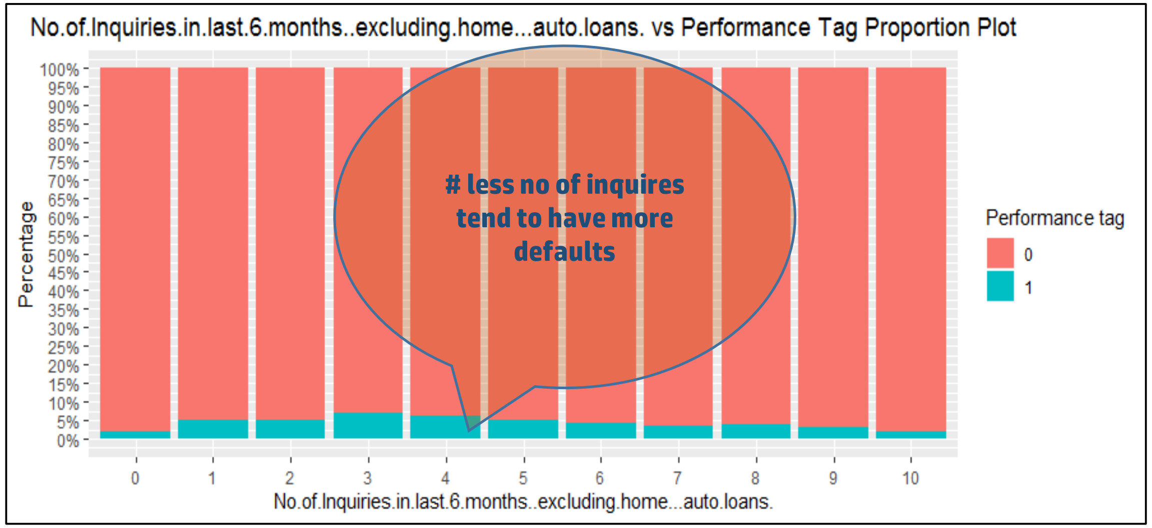


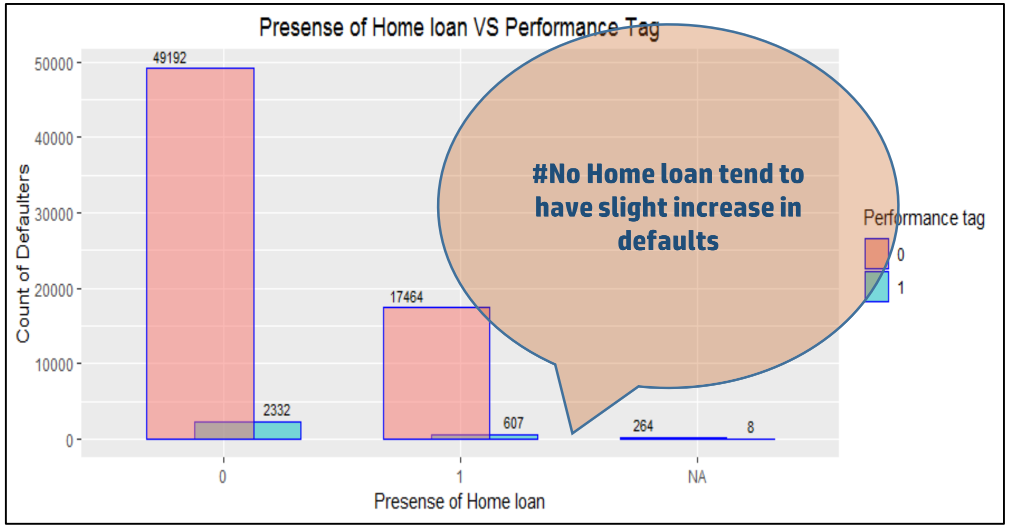


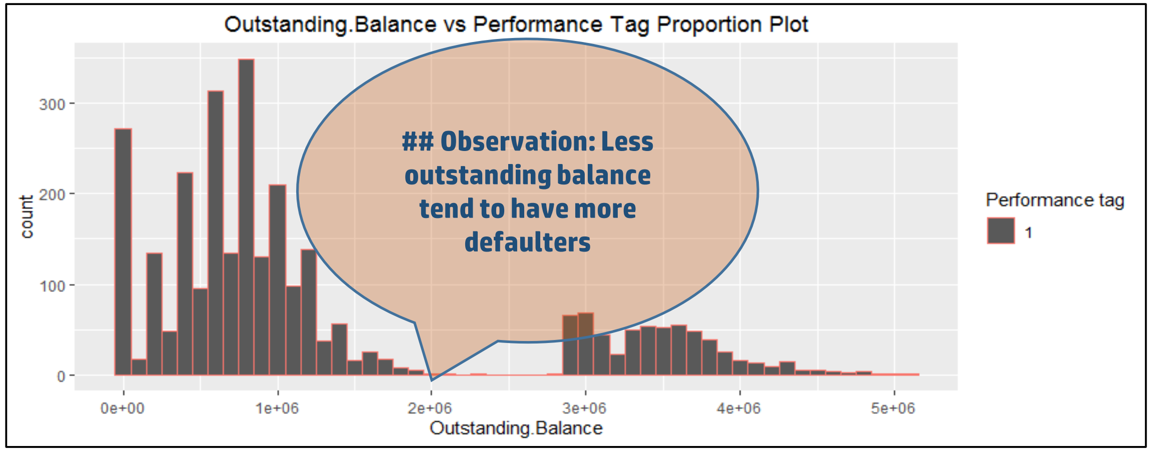


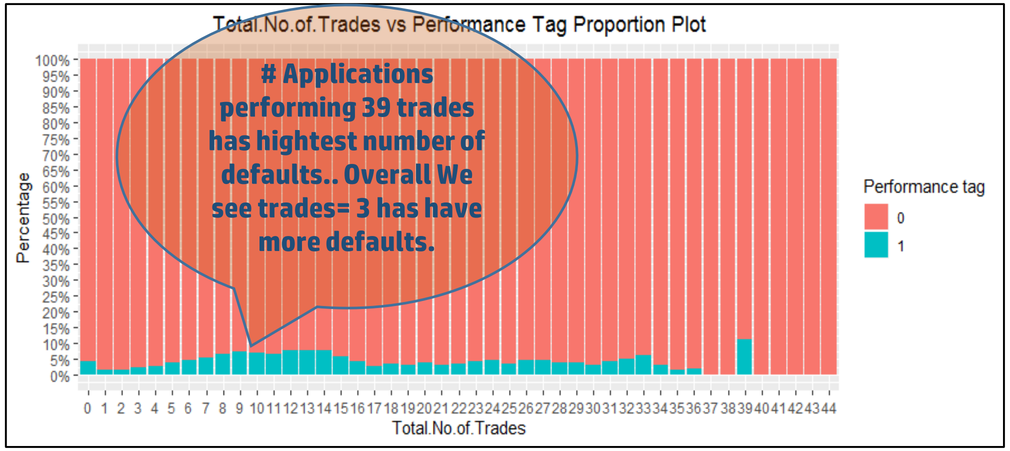


Exploratory Data Analysis – Bivariate Analysis





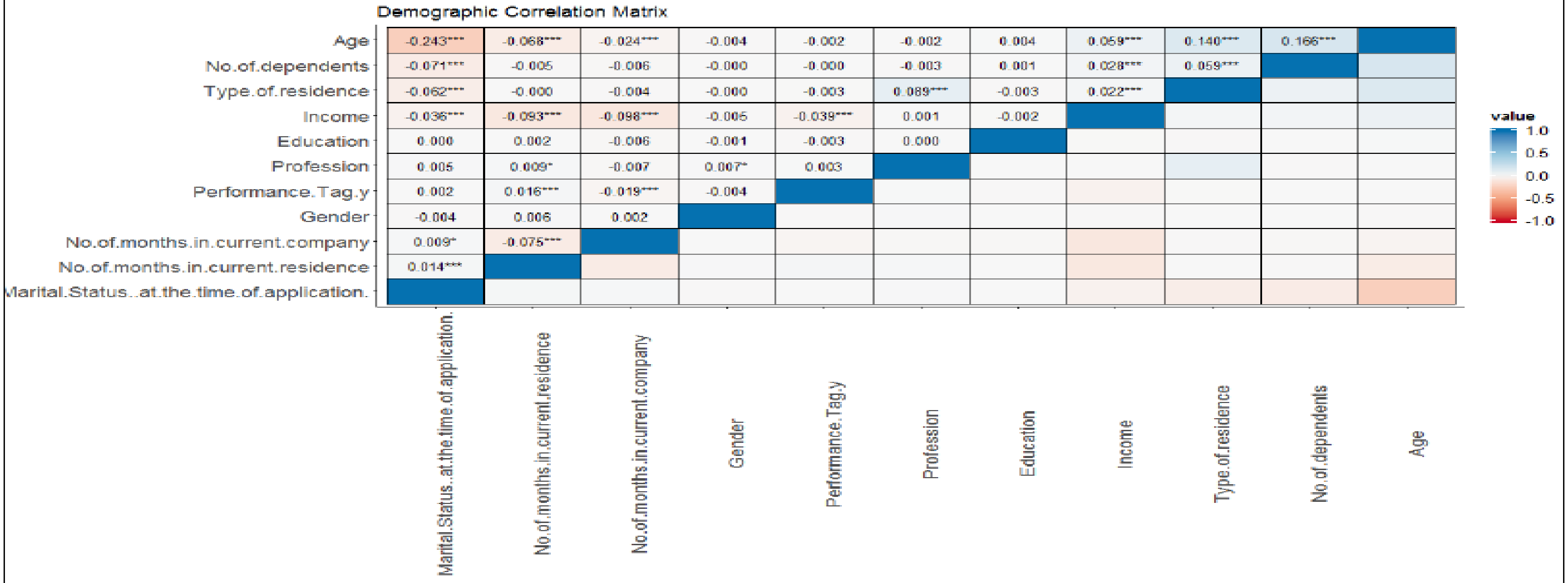




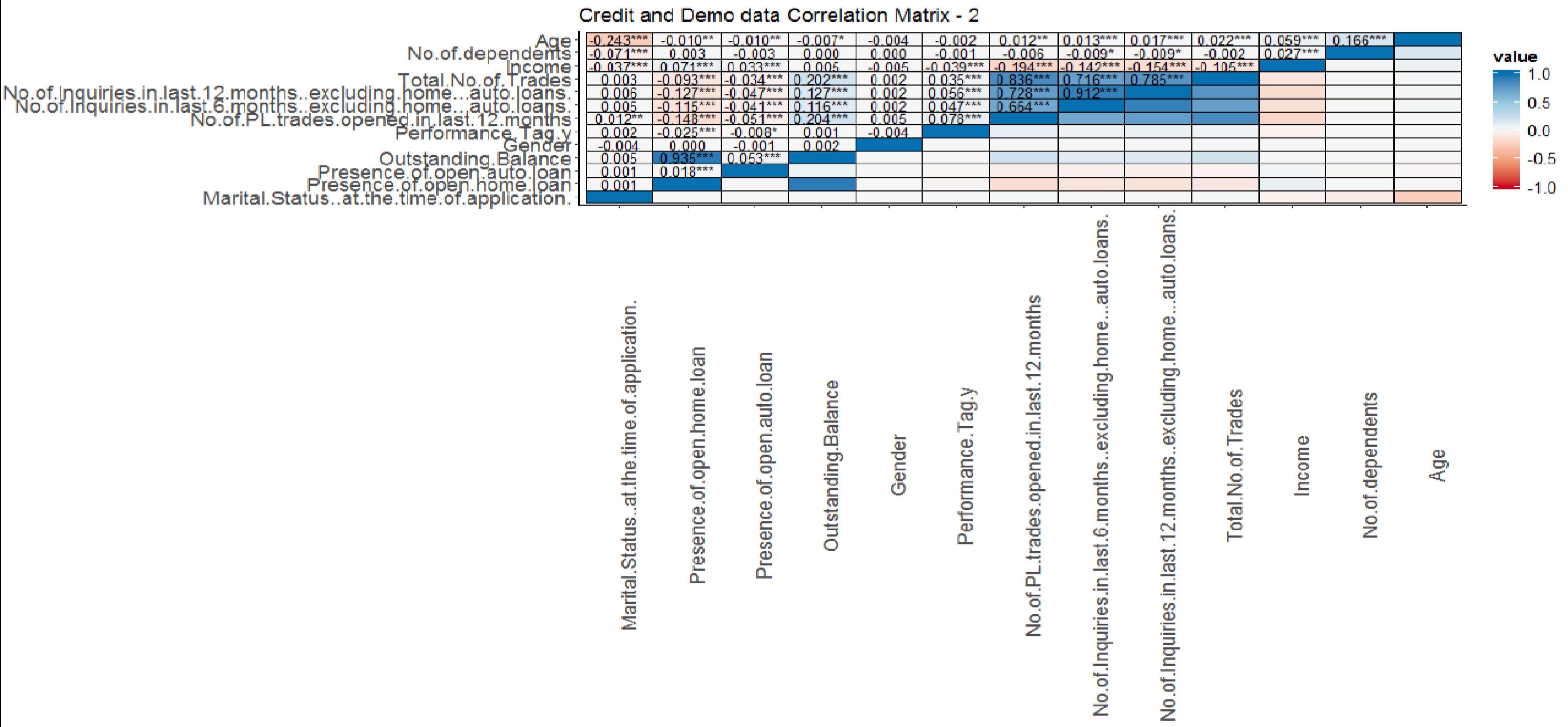
Exploratory Data Analysis – Co-Relation Matrices

Demographic data correlation plot:

Income have inverse correlation with Performance tag with -0.039 index and also highly significant showing p value less than 0.05 (***)



Exploratory Data Analysis – Co-Relation Matrices



Model Building, Evaluation, Model Selection, Benefits to Bank

Models

- Logistic - Demographic Models – Not having good accuracy
- Logistic - Complete Models - More Stable, Not overfitting
- Decision tree Models – Not stable
- Random Forest Models – Over fitting

Evaluation

- Individual Model – **See the Appendix**
- Best model selections – **See the Appendix**
- KS statistics for Logistics – **See the Appendix**
- Scorecard – **See the Appendix**
- Lift and Gain chart – **See the Appendix**

Best Model

- Logistic - Complete Models with WOE – Chosen one as more stable and not overfitting
- **Evaluated the Rejected records with the model – It shows 95% would have been rejected by the model too**
- For results – **See the Next Page**

Benefits to Bank

- Loss of accepting 100 customers per month in absence of the model – **See the Next Page**
- For Results – **See the Next Page**

Model Building, Evaluation, Model Selection, Benefits to Bank

	Model	Accuracy	Kappa	AccuracyLower	AccuracyUpper	Sensitivity	Specificity	Precision	Recall	F1
1	LR Merged	0.6373569	0.05608301	0.6308058	0.6438698	0.6289593	0.6377266	0.07101801	0.6289593	0.1276254
2	LR Demographic	0.5898855	0.03371514	0.5831909	0.5965552	0.5871041	0.5900080	0.05931429	0.5871041	0.1077434
3	Decision Tree	0.6576499	0.31541293	0.6497147	0.6655194	0.6331479	0.6823208	0.66741843	0.6331479	0.6498316
4	Random Forest	0.9960641	0.99212822	0.9948800	0.9970336	0.9989984	0.9931281	0.99317212	0.9989984	0.9960768

```
##########################Loss and Gain Selected Model####################################
##########################################################################################
conf_final$table
#             Reference
#Prediction     No    Yes
#No           12803   328
#Yes          7273    556

Onboarding_Probable_Wrong_Customer <- conf_final$table[3][1]
Rejecting_Probable_Good_Customer <- conf_final$table[2][1]

Onboarding_Probable_Wrong_Customer_Percentage <- (Onboarding_Probable_Wrong_Customer/sum(conf_final$table))*100  #1
Rejecting_Probable_Good_Customer_Percentage <- (Rejecting_Probable_Good_Customer/sum(conf_final$table))*100


# > Onboarding_Probable_Wrong_Customer_Percentage
# [1] 1.564885

# > Rejecting_Probable_Good_Customer_Percentage
# [1] 34.69943

# Assuming the average monthly profit-loss for rejecting 1 good customer is Rs. 2000
# Assuming the average monthly loss for onboarding a defaulter is Rs. 100000
# Let's calculate the loss-gain for bank in a 100 applications
#Without model Bank will accept all 100 customer
# Bank will lose

Bank_lose_per_100_application <- Onboarding_Probable_Wrong_Customer_Percentage*100000 - Rejecting_Probable_Good_Customer_Percentage*2000

# > Bank_lose_per_100_application
# [1] 87089.69
```

CREDIT RISK & ACQUISITION – **ACHIEVEMENTS**

Model Definition & Important Indicators

```
model_8 <- glm(Performance.Tag.y ~ `No.of.dependents:WOE`
               + `Profession:WOE`
               + `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`
               + `binning.Avgas.CC.Utilization.in.last.12.months:WOE`
               + `binning.Outstanding.Balance:WOE`
               + `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`
               , data = train, family = "binomial")

summary(model_8)

# (Intercept)                                                              -3.12164    0.02370 -131.710  < 2e-16 ***
#   `No.of.dependents:WOE`                                                  -1.64797    0.44741   -3.683  0.00023 ***
#   `Profession:WOE`                                                        -1.28401    0.46825   -2.742  0.00610 **
#   `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`                      -0.37804    0.05484   -6.893 5.46e-12 ***
#   `binning.Avgas.CC.Utilization.in.last.12.months:WOE`                    -0.39695    0.06133   -6.472 9.67e-11 ***
#   `binning.Outstanding.Balance:WOE`                                       -0.20856    0.06654   -3.134  0.00172 **
#   `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE` -0.44743    0.06317   -7.083 1.41e-12 ***
```

CREDIT RISK & ACQUISITION – **ACHIEVEMENTS**

Confusion Matrix and Statistics

```
Confusion Matrix and Statistics

          Reference
Prediction    No   Yes
       No  12803   328
       Yes  7273   556

               Accuracy : 0.6374
                 95% CI : (0.6308, 0.6439)
    No Information Rate : 0.9578
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0561
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.62896
            Specificity : 0.63773
         Pos Pred Value : 0.07102
         Neg Pred Value : 0.97502
             Prevalence : 0.04218
         Detection Rate : 0.02653
   Detection Prevalence : 0.37352
      Balanced Accuracy : 0.63334

       'Positive' Class : Yes
```

CREDIT RISK & ACQUISITION – **ACHIEVEMENTS**

Lift & Gain

```
default_decile = lift(test_actual_default, test_pred, groups = 10)

# > default_decile
# # A tibble: 10 x 6
# bucket total totalresp Cumresp  Gain Cumlift
# <int> <int>     <dbl>   <dbl> <dbl>   <dbl>
# 1      1  2096       181     181  20.5    2.05
# 2      2  2096       168     349  39.5    1.97
# 3      3  2096       116     465  52.6    1.75
# 4      4  2096       118     583  66.0    1.65
# 5      5  2096        96     679  76.8    1.54
# 6      6  2096        56     735  83.1    1.39
# 7      7  2096        59     794  89.8    1.28
# 8      8  2096        35     829  93.8    1.17
# 9      9  2096        26     855  96.7    1.07
# 10     10  2096        29     884 100       1

#############################################################################
################################ Odds ratio #############################################
#############################################################################
```

CREDIT RISK & ACQUISITION – **ACHIEVEMENTS**

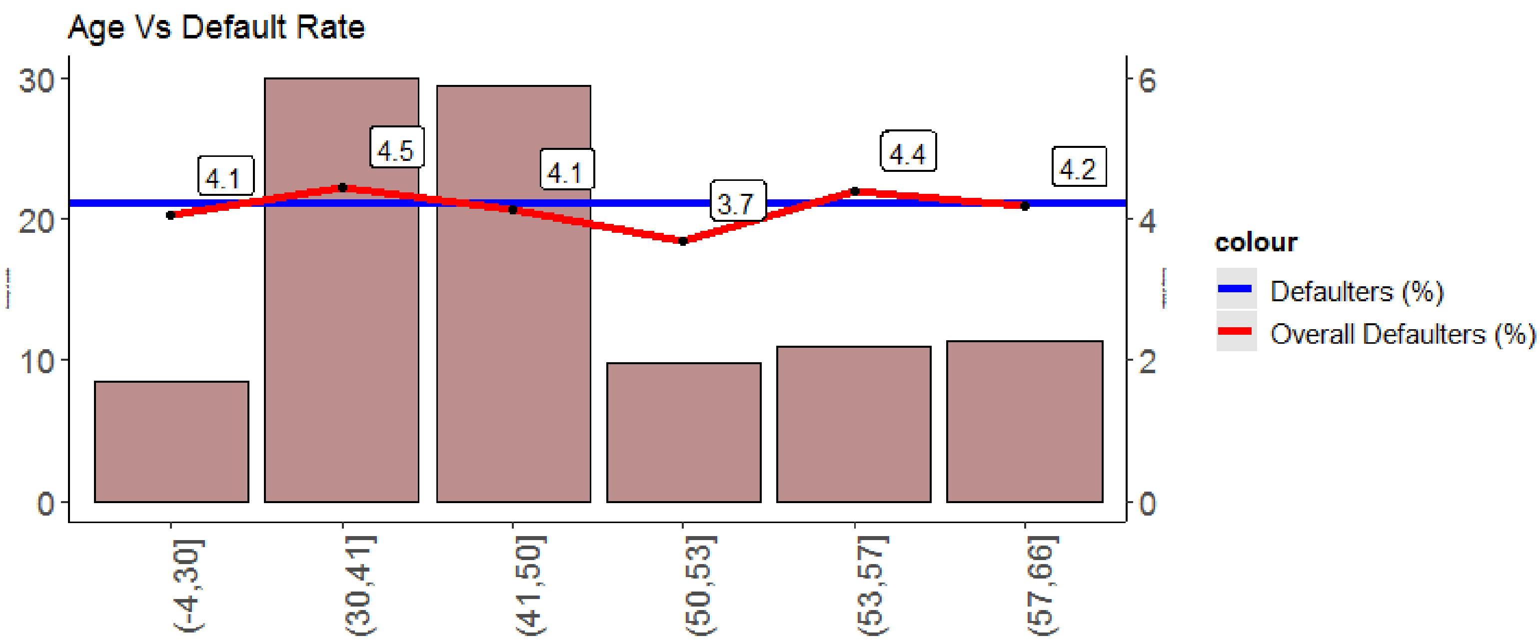
Loss to Bank in Absence of the Model

```
#########################Loss and Gain Selected Model####################################
##########################################################################################
conf_final$table
#            Reference
#Prediction     No    Yes
#No          12803    328
#Yes          7273    556

Onboarding_Probable_Wrong_Customer <- conf_final$table[3][1]
Rejecting_Probable_Good_Customer <- conf_final$table[2][1]

Onboarding_Probable_Wrong_Customer_Percentage <- (Onboarding_Probable_Wrong_Customer/sum(conf_final$table))*100  #1
Rejecting_Probable_Good_Customer_Percentage <- (Rejecting_Probable_Good_Customer/sum(conf_final$table))*100


# > Onboarding_Probable_Wrong_Customer_Percentage
# [1] 1.564885

# > Rejecting_Probable_Good_Customer_Percentage
# [1] 34.69943

# Assuming the average monthly profit-loss for rejecting 1 good customer is Rs. 2000
# Assuming the average monthly loss for onboarding a defaulter is Rs. 100000
# Let's calculate the loss-gain for bank in a 100 applications
#Without model Bank will accept all 100 customer
# Bank will lose

Bank_lose_per_100_application <- Onboarding_Probable_Wrong_Customer_Percentage*100000 - Rejecting_Probable_Good_Customer_Percentage*2000

# > Bank_lose_per_100_application
# [1] 87089.69
```

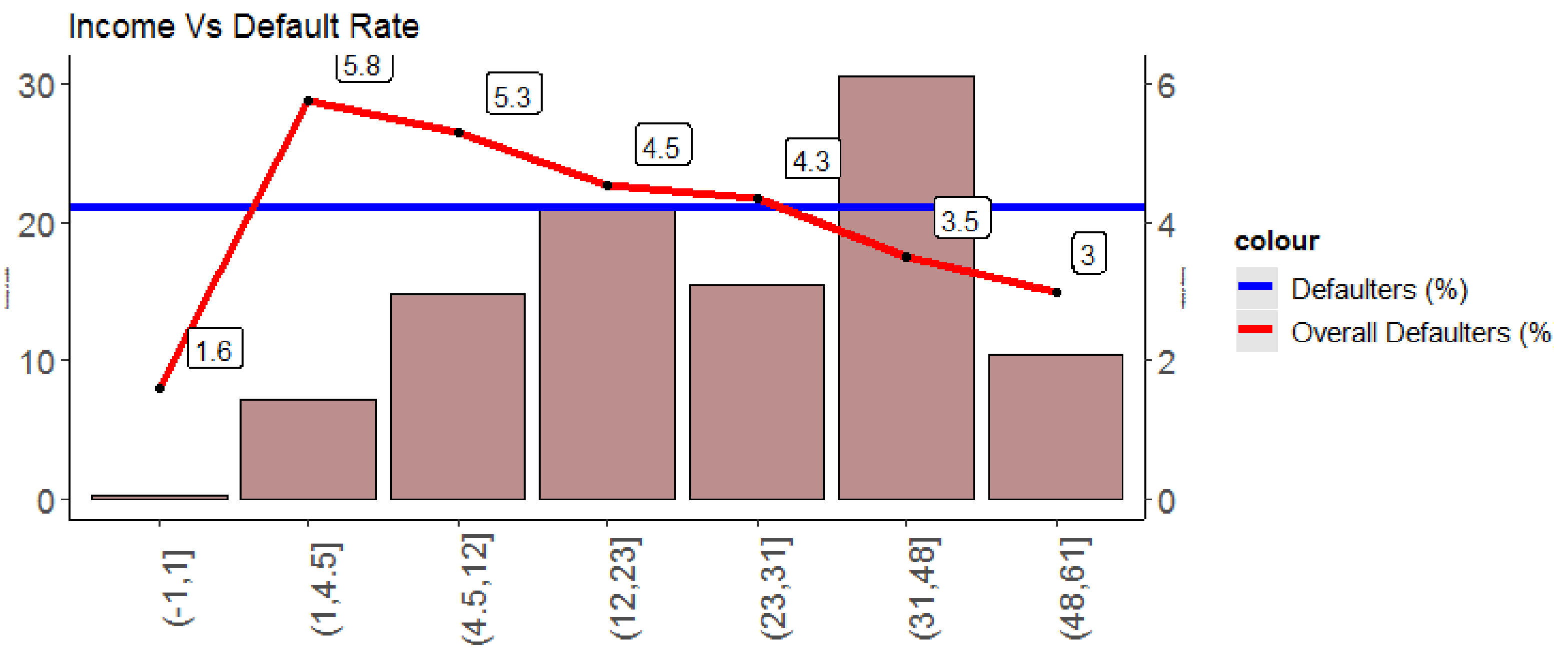
CREDIT RISK & ACQUISITION – **APPENDIX**

APPENDIX – FEW IMPORTANT GRAPHS

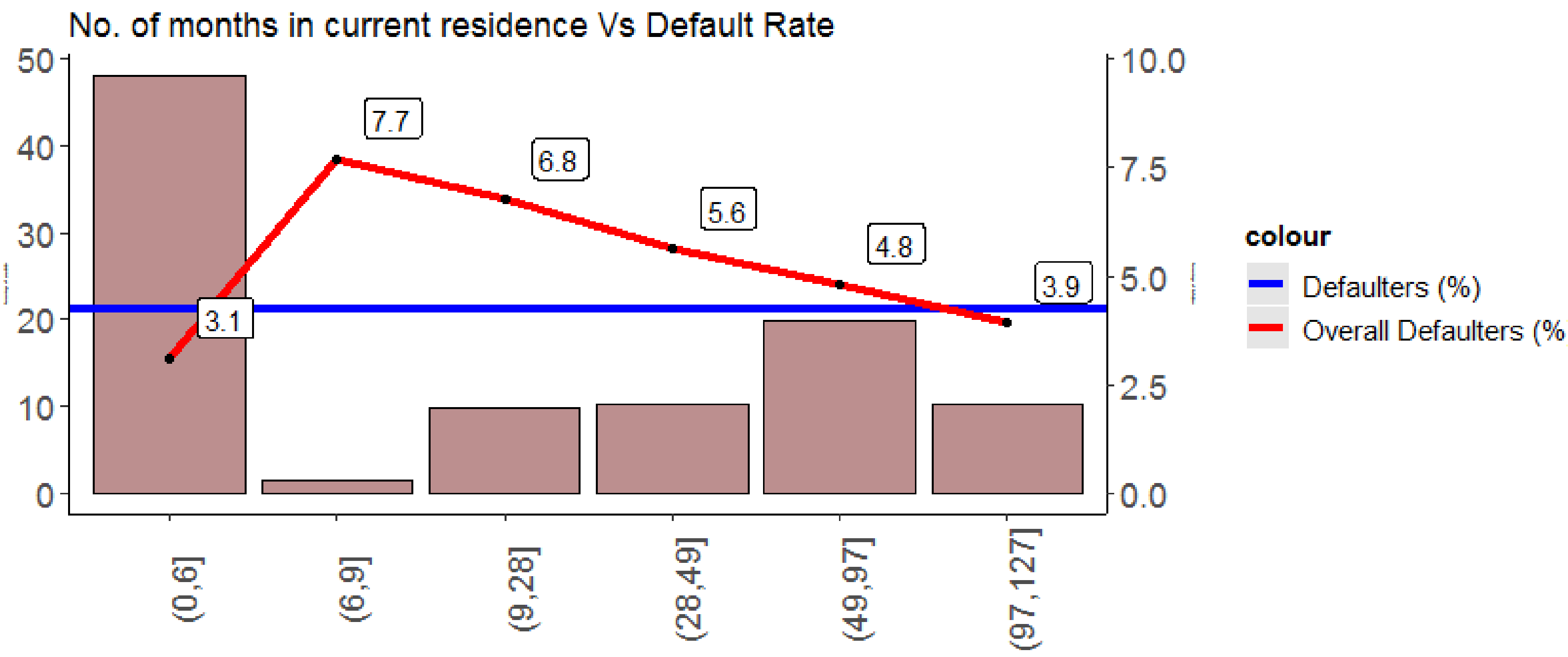
Comparing the defaulters in each age bin with the overall defaulters rate. Almost all the bins has defaulters close to overall default rate (4.22%).

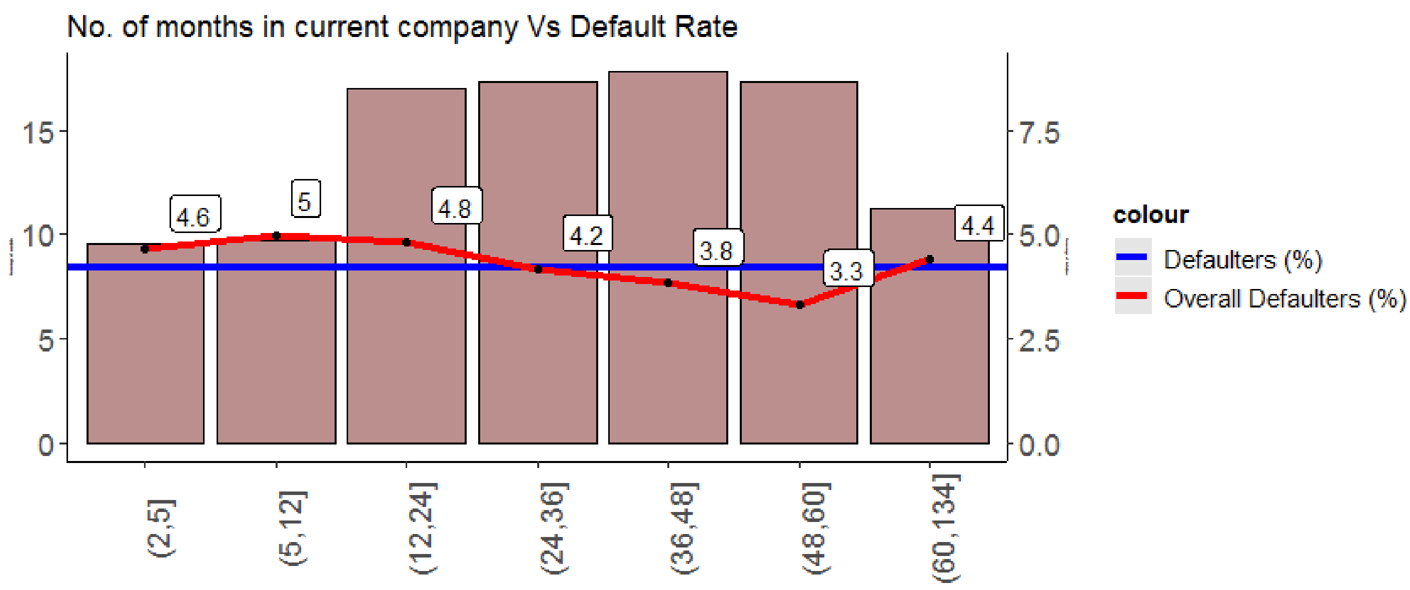


The average default rate is decresing as the income increases. Applicants with Income groups within 1-23 have default rate above than the overall default rate.



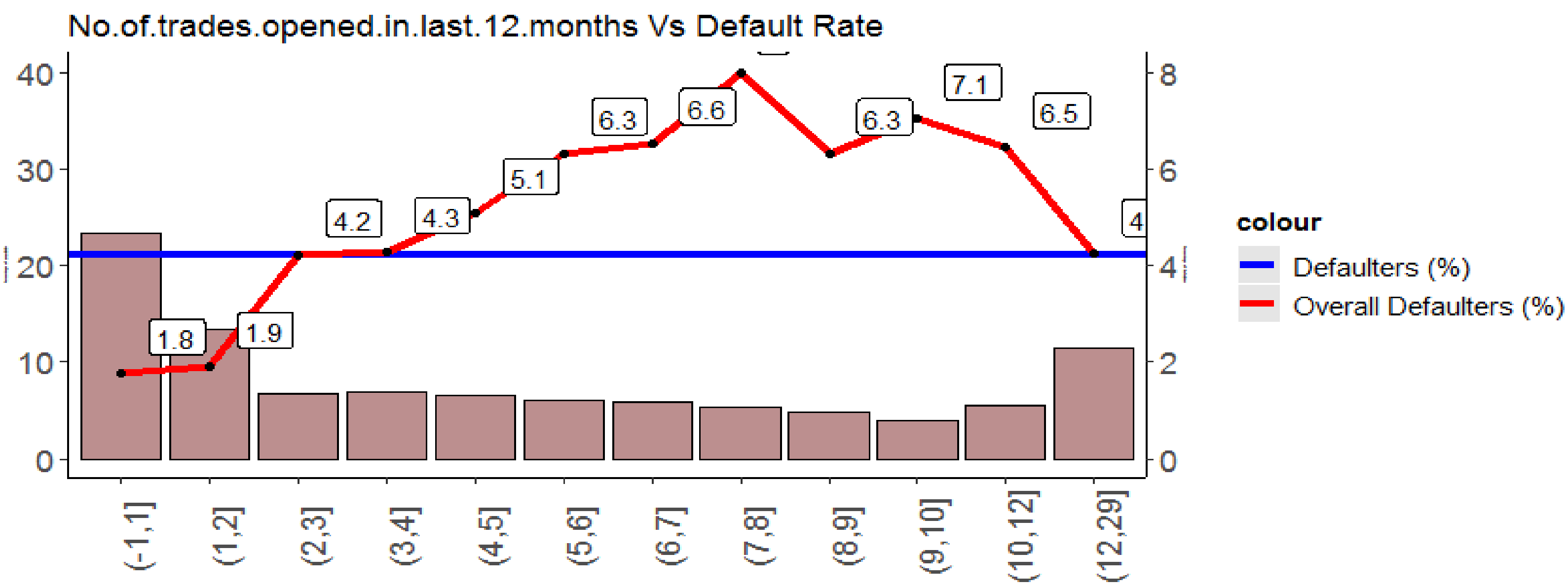
Applicants residing in current residence from 6-9 months observe maximum default rate





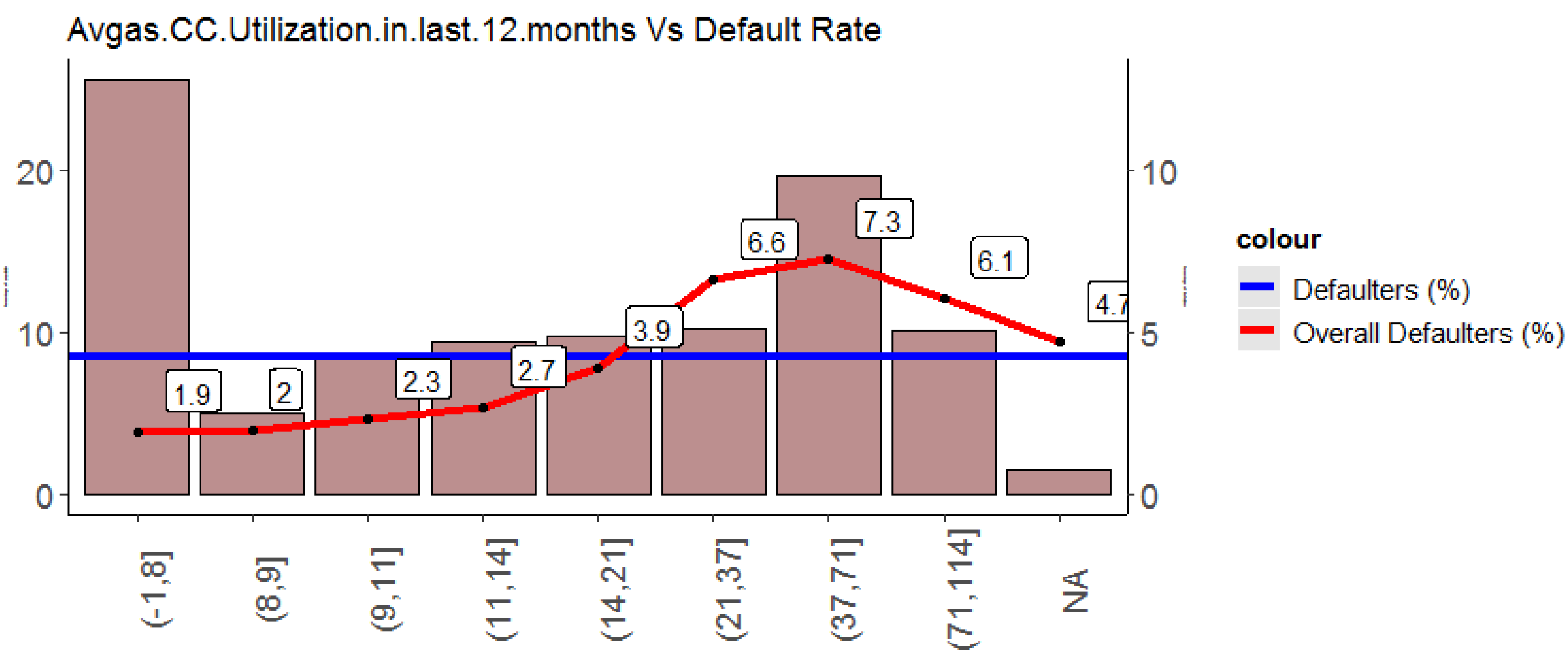
Important Observation:

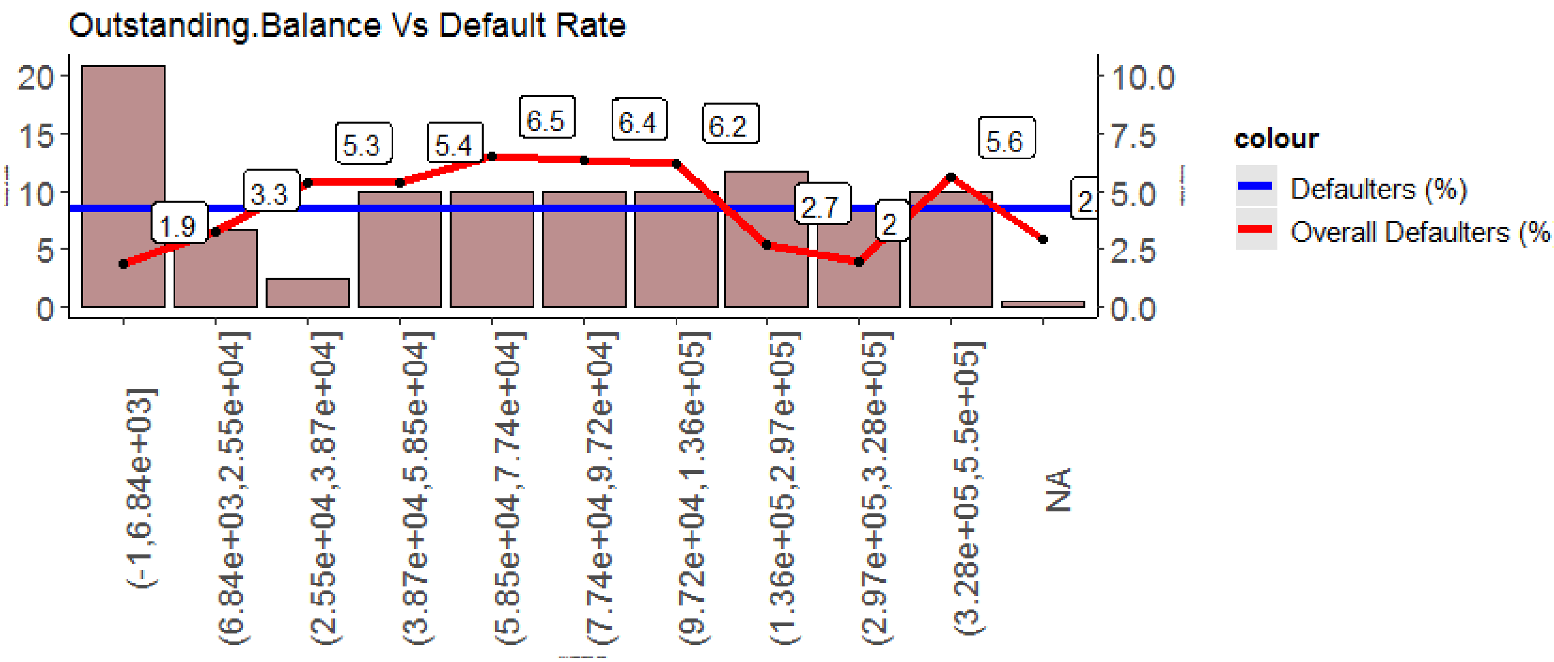
No of trades show the maximum difference between default rate in 1st bin and last bin which clearly explains the default rate pattern..



Important Observation:

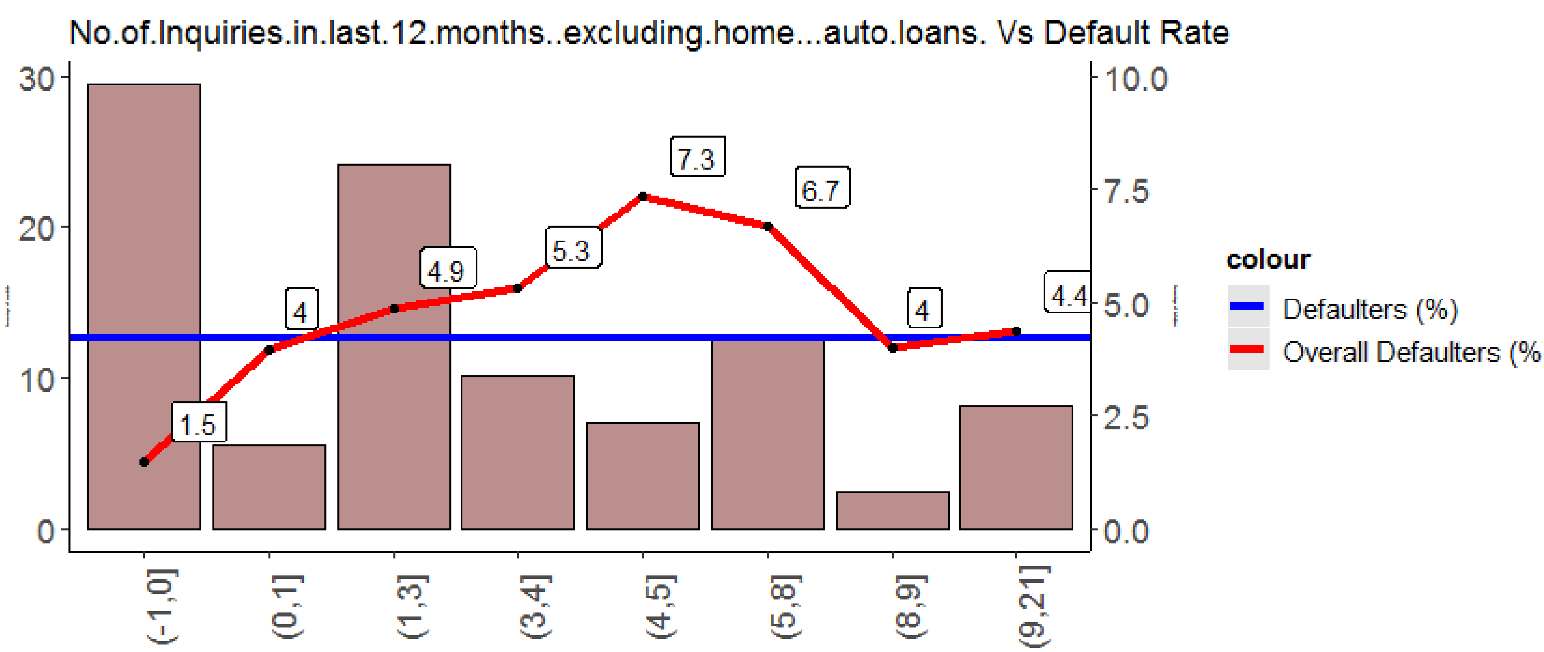
Avg CC utilization show the maximum difference between default rate in 1st bin and last bin which clearly explains the default rate pattern..





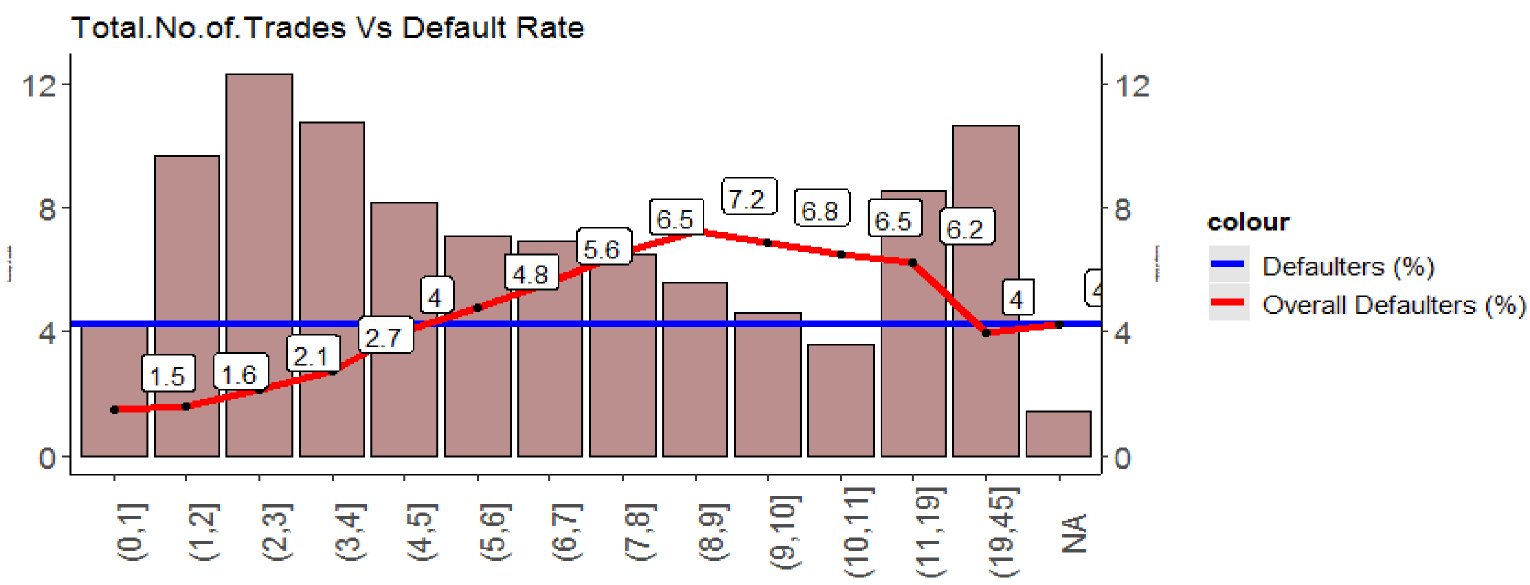
Important observation:

Show the maximum difference between default rate in 1st bin and last bin which clearly explains the default rate pattern..



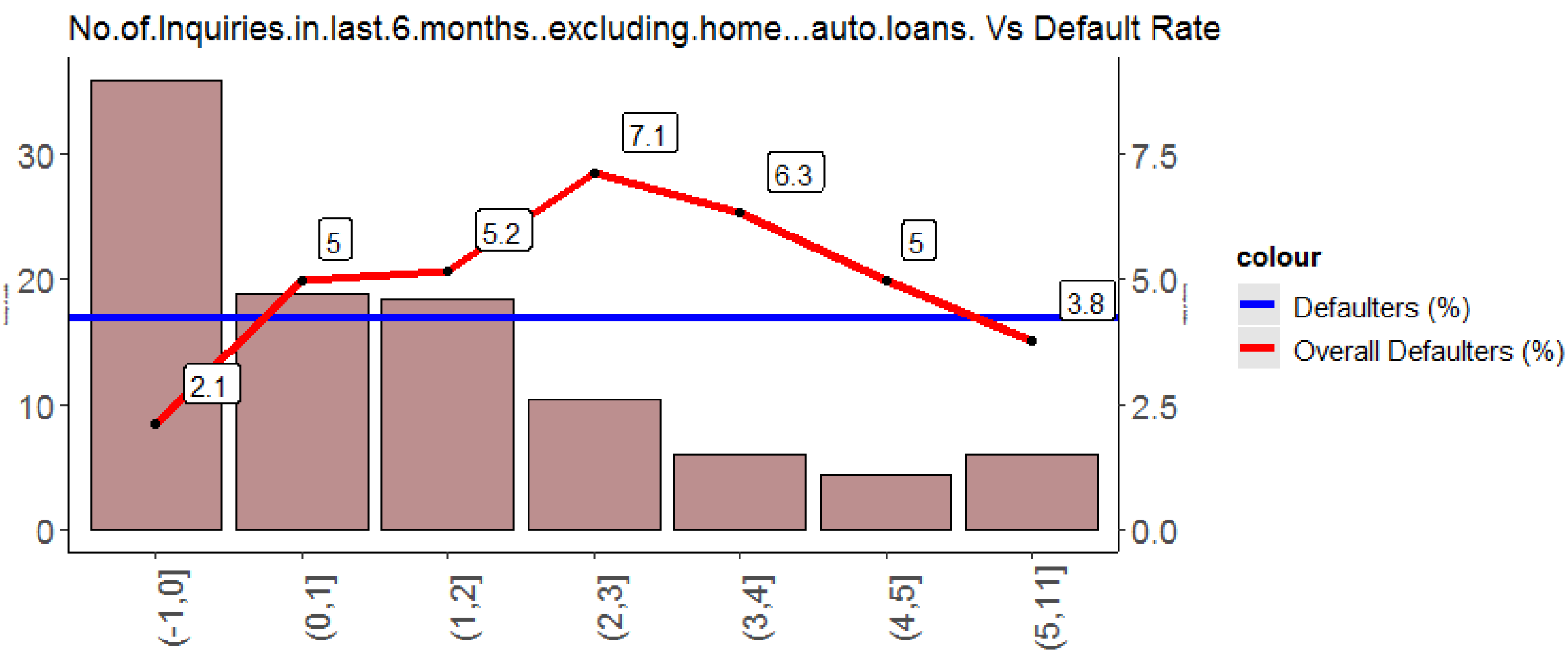
Important Observation:

Show the maximum difference between default rate in 1st bin and last bin which clearly explains the default rate pattern..

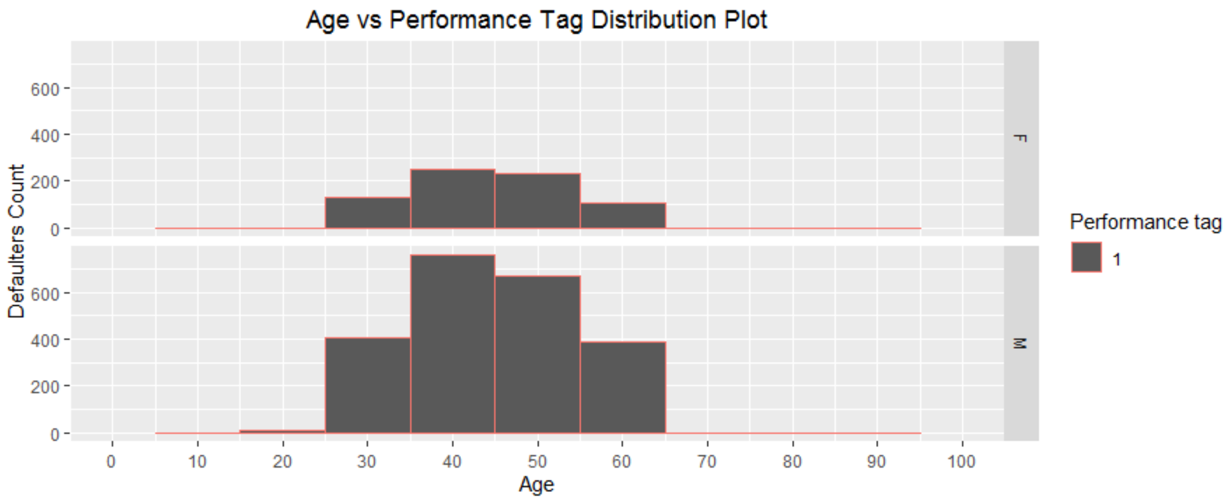


Important Observation

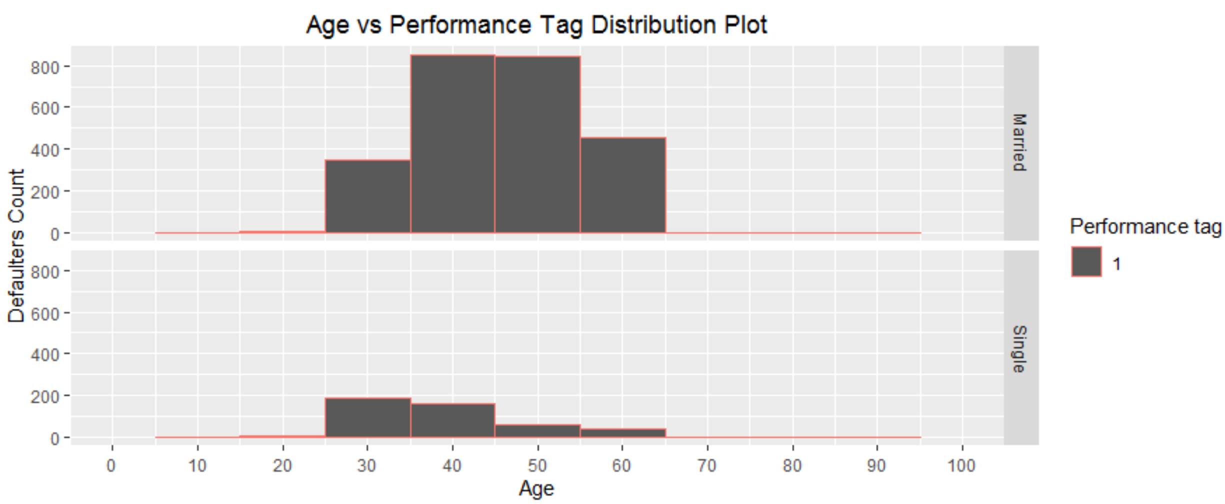
Show the maximum difference between default rate in 1st bin and last bin which clearly explains the default rate pattern..

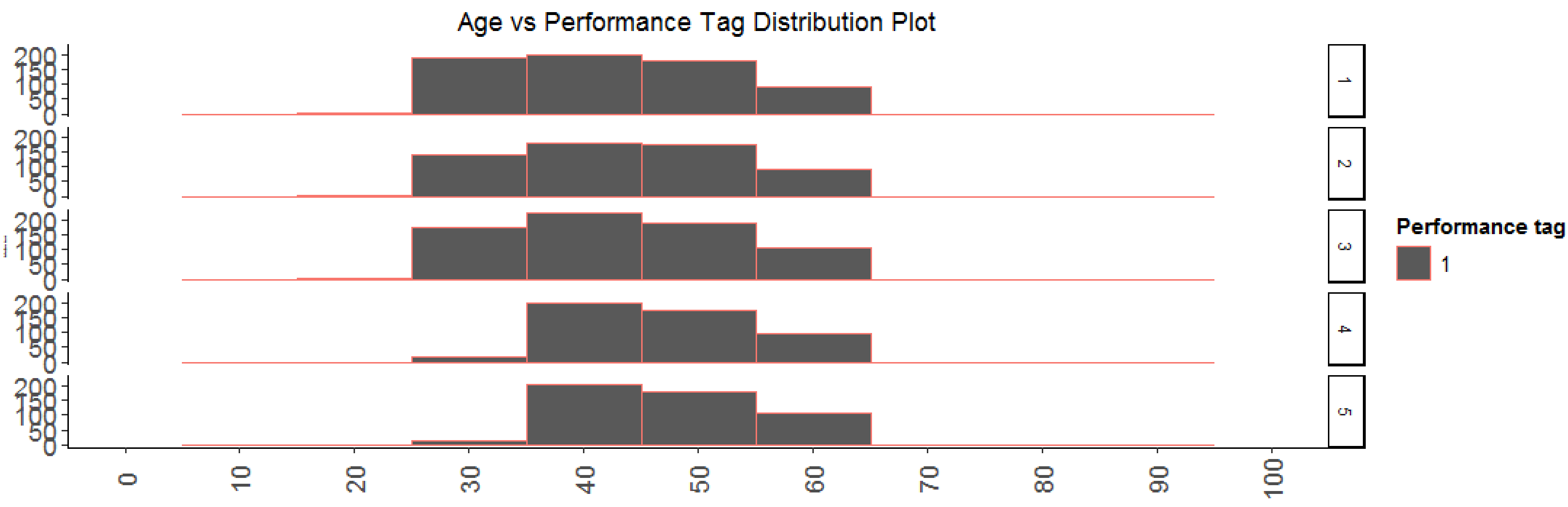


Obervation: 35-55 age groups have majority of defaults and Gender have equal distribution

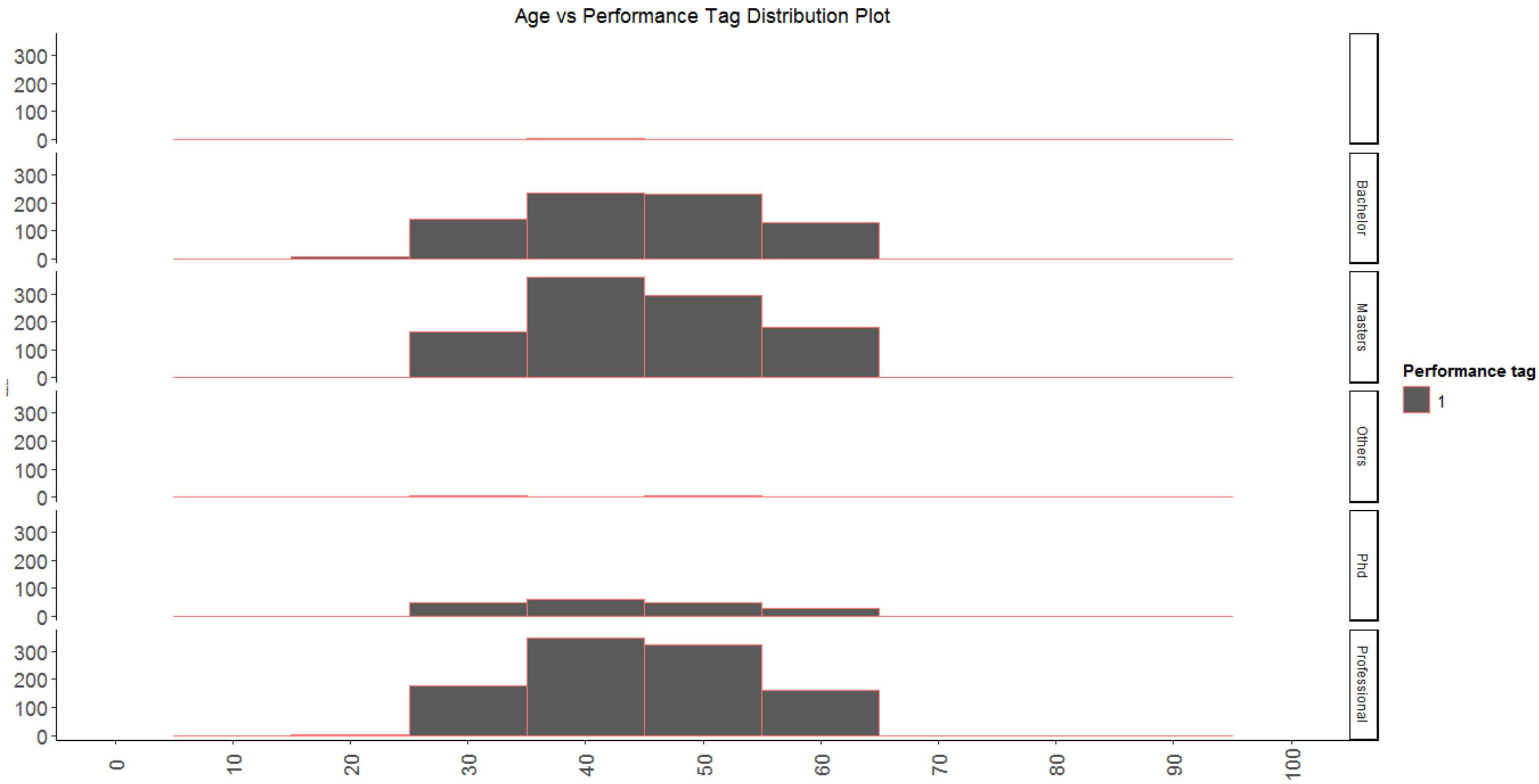


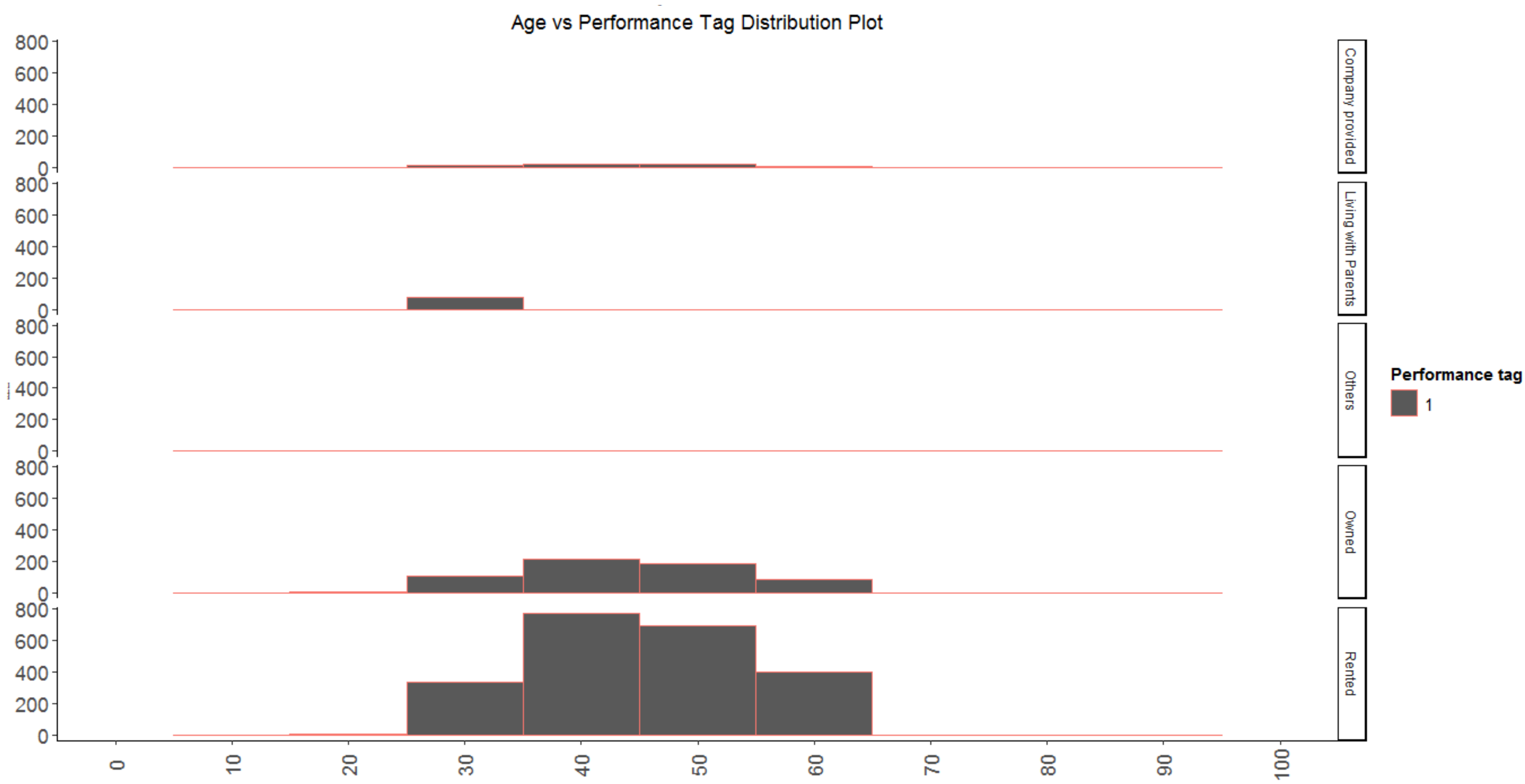
Obervation: Single status applicants tend to default more at early age compare to married

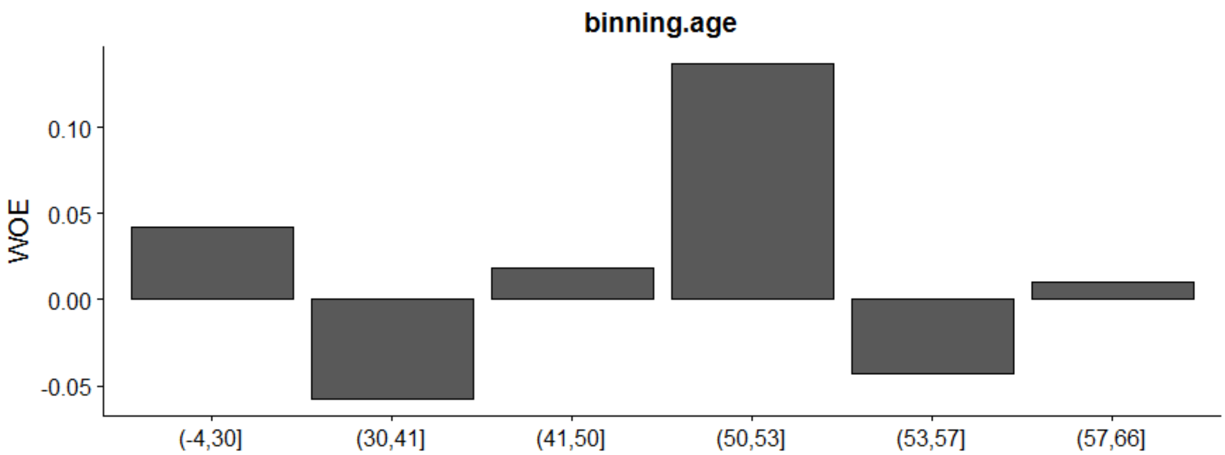


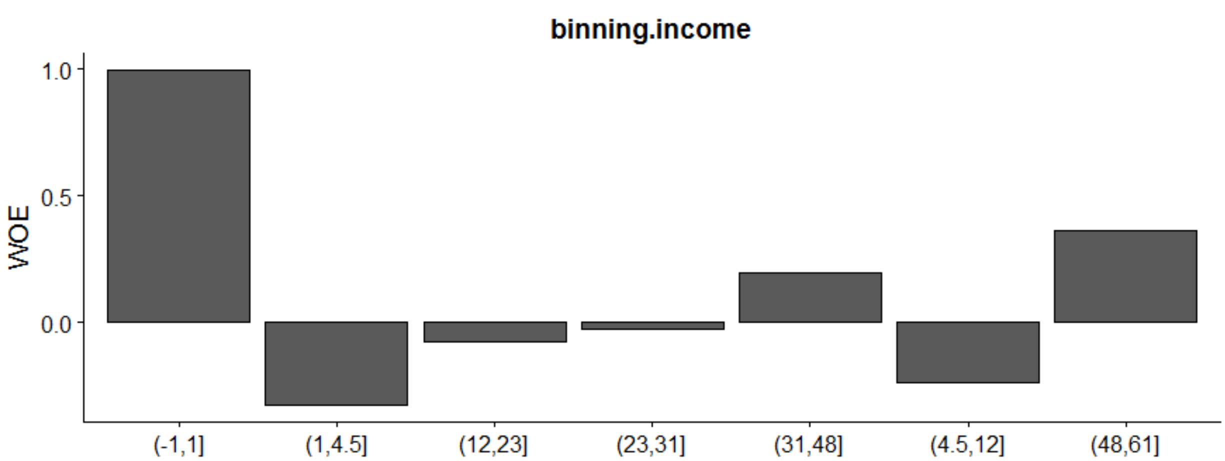


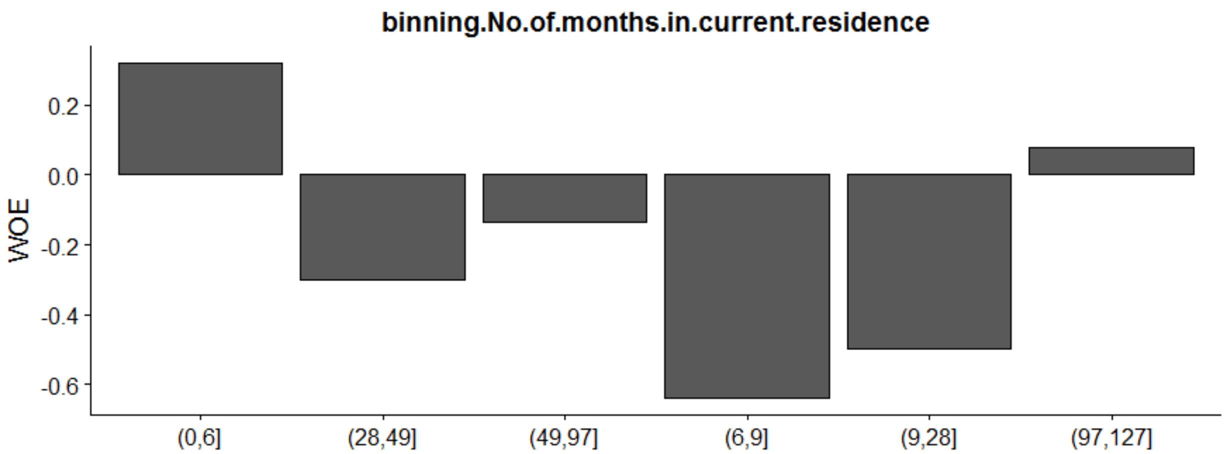
Age vs Performance Tag Distribution Plot
200
150
100
50
0
1
2
3
4
5
Performance tag
1
0
10
20
30
40
50
60
70
80
90
100

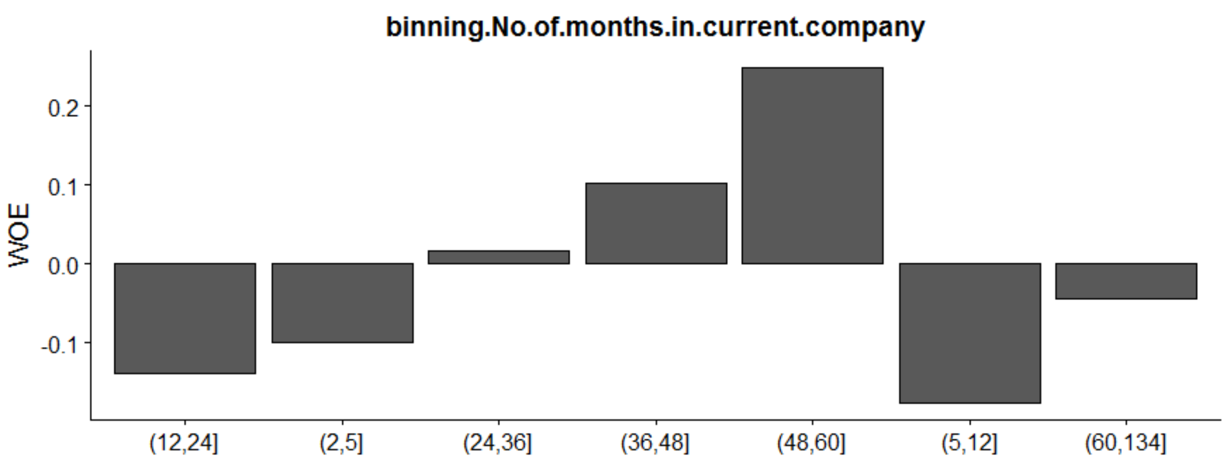

Age vs Performance Tag Distribution Plot
Bachelor
Masters
Others
Phd
Professional
Performance tag
1

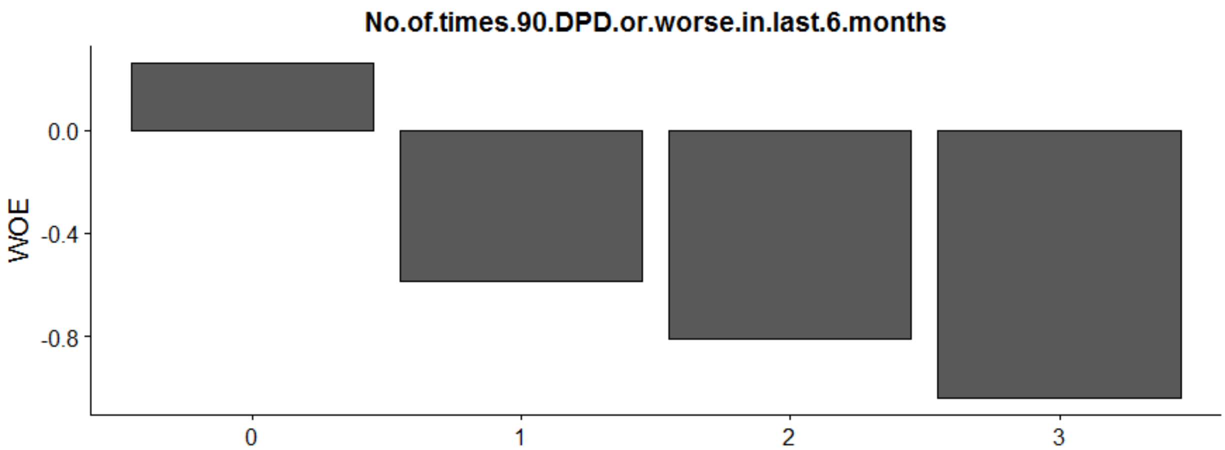

Age vs Performance Tag Distribution Plot
Company provided
Living with Parents
Others
Owned
Rented
Performance tag
1

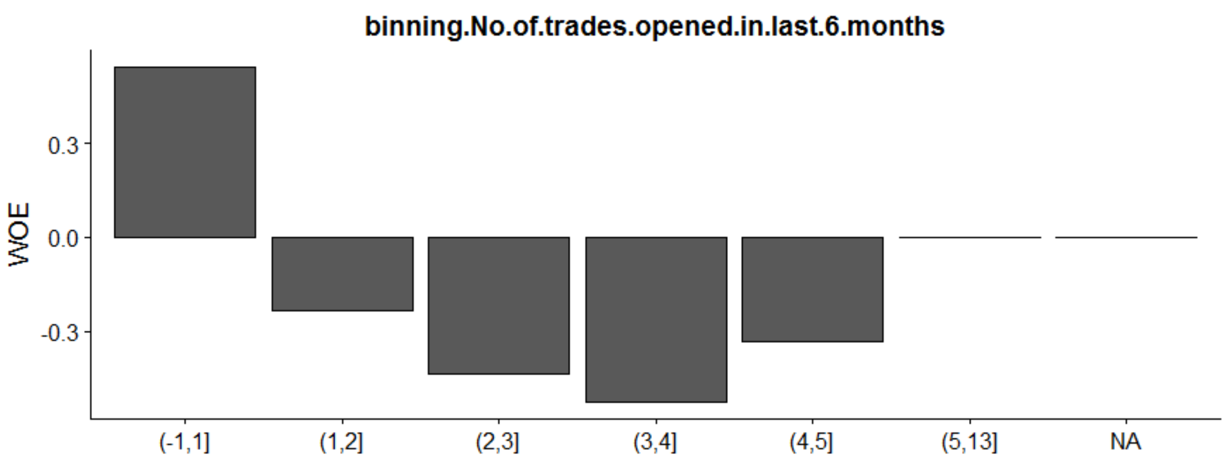

binning.age
WOE
0.10
0.05
0.00
-0.05
(-4,30]
(30,41]
(41,50]
(50,53]
(53,57]
(57,66]


binning.income
WOE
1.0
0.5
0.0
(-1,1]
(1,4.5]
(12,23]
(23,31]
(31,48]
(4.5,12]
(48,61]

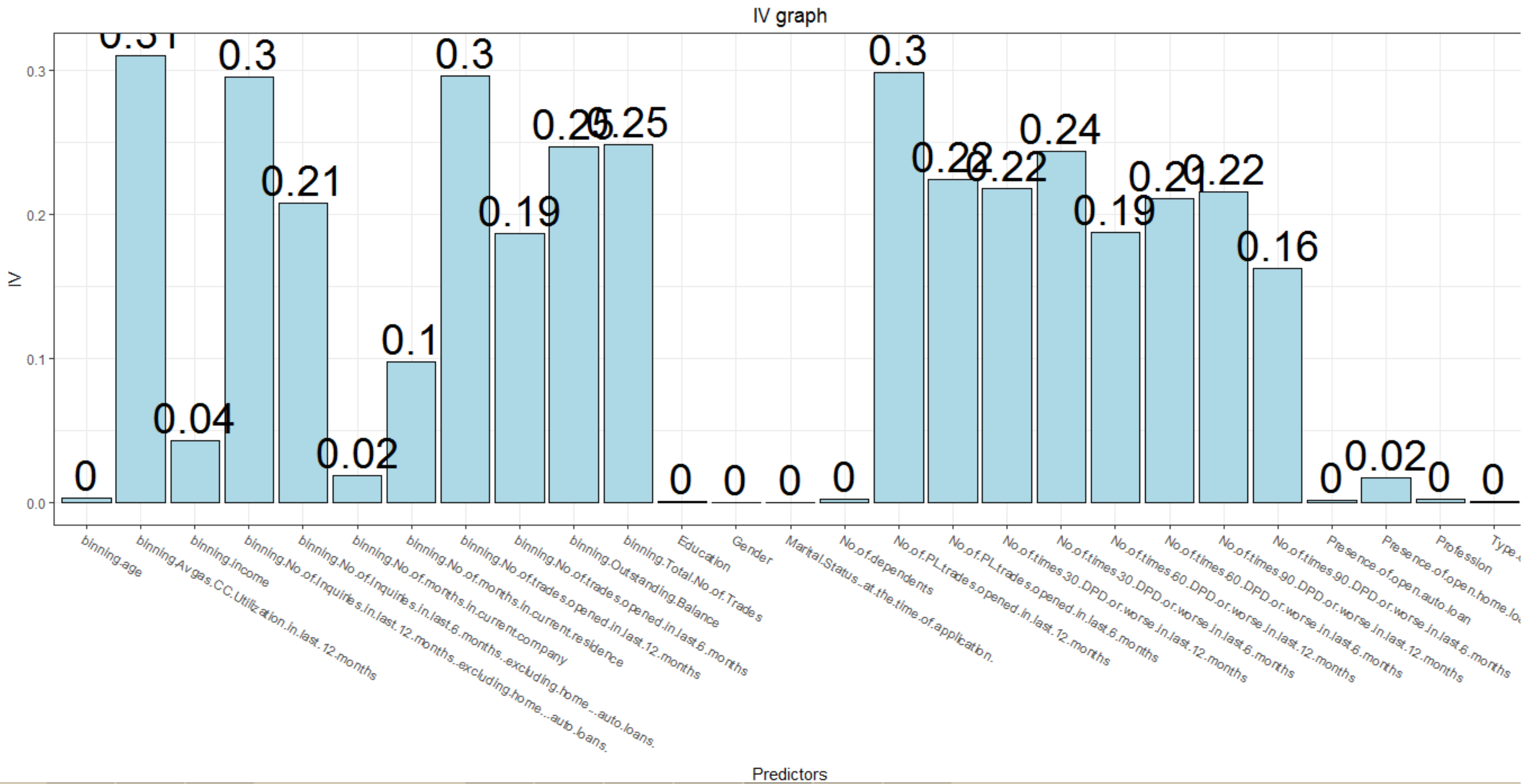

binning.No.of.months.in.current.residence
WOE
0.2
0.0
-0.2
-0.4
-0.6
(0,6]
(28,49]
(49,97]
(6,9]
(9,28]
(97,127]


binning.No.of.months.in.current.company
WOE
0.2
0.1
0.0
-0.1
(12,24]
(2,5]
(24,36]
(36,48]
(48,60]
(5,12]
(60,134]

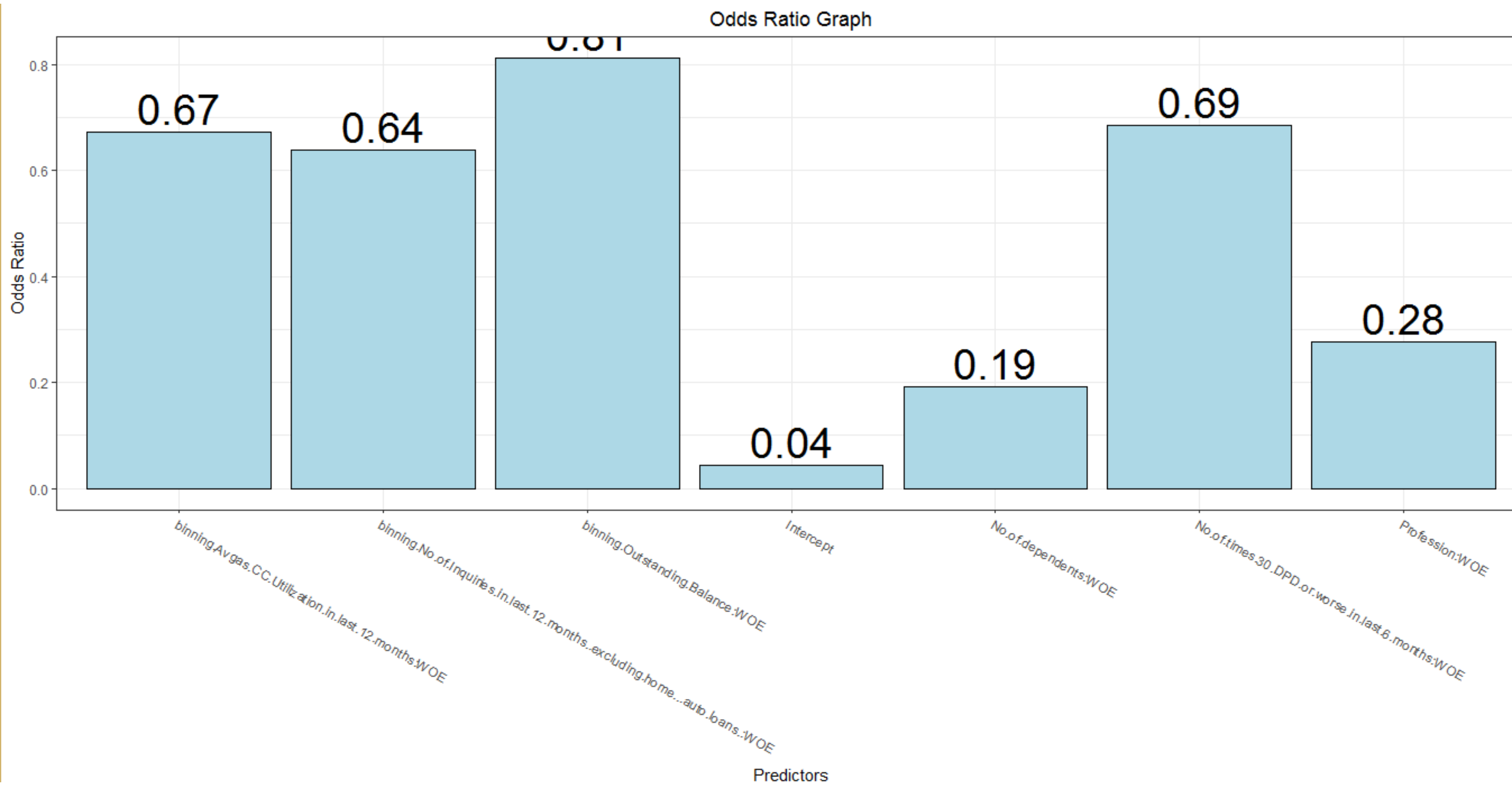

No.of.times.90.DPD.or.worse.in.last.6.months
WOE
0.0
-0.4
-0.8
0
1
2
3


binning.No.of.trades.opened.in.last.6.months
WOE
0.3
0.0
-0.3
(-1,1]
(1,2]
(2,3]
(3,4]
(4,5]
(5,13]
NA

Iv GRAPH

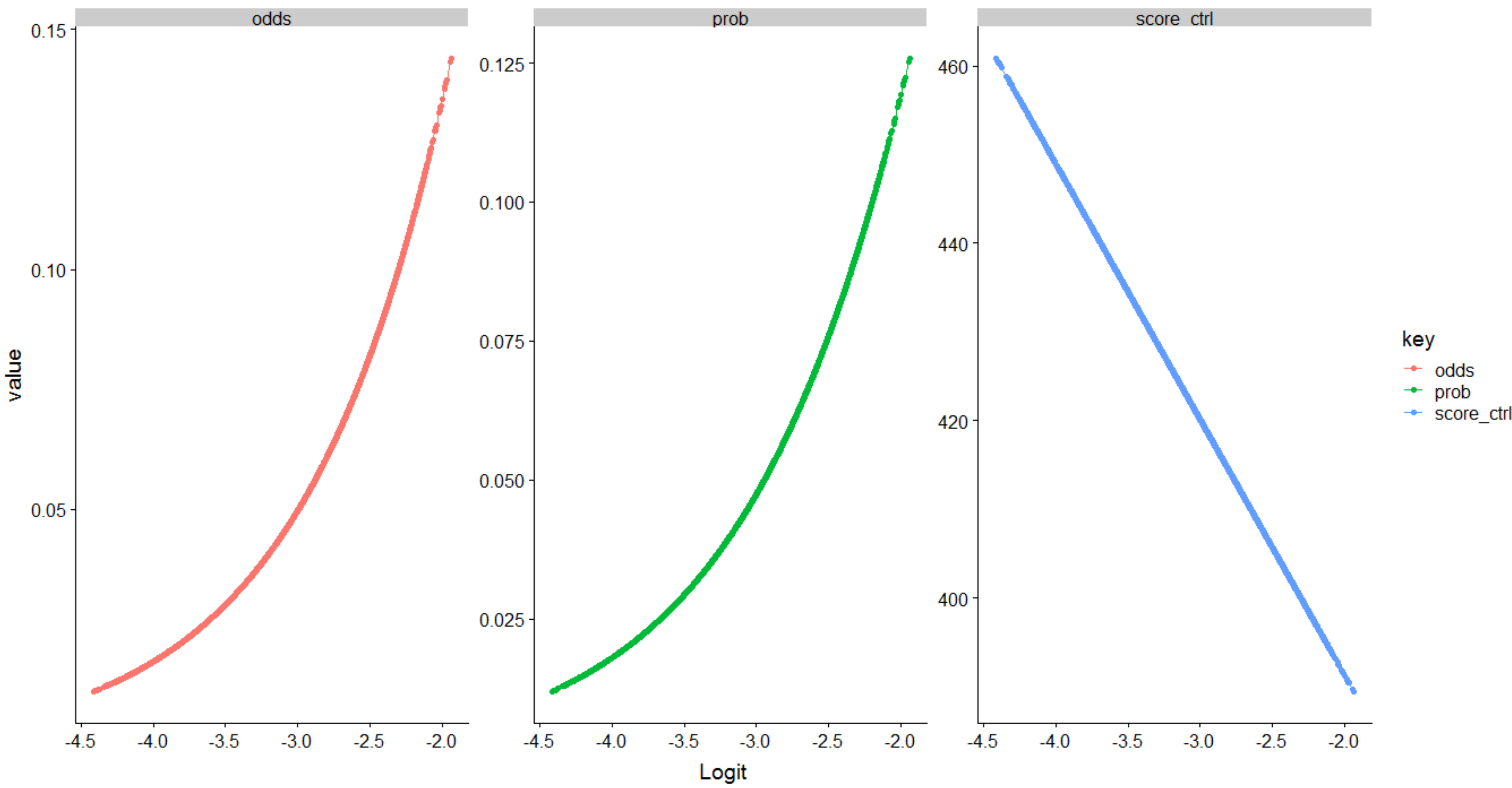

IV graph
IV
0.0
0.1
0.2
0.3
0
0.04
0.3
0.21
0.02
0.1
0.3
0.19
0.25
0.25
0
0
0
0
0.3
0.22
0.22
0.24
0.19
0.21
0.22
0.16
0
0.02
0
0
binning.age
binning.Avgas.CC.Utilization.in.last.12.months
binning.income
binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.
binning.No.of.Inquiries.in.last.6.months..excluding.home...auto.loans.
binning.No.of.months.in.current.company
binning.No.of.months.in.current.residence
binning.No.of.trades.opened.in.last.12.months
binning.No.of.trades.opened.in.last.6.months
binning.Outstanding.Balance
binning.Total.No.of.Trades
Education
Gender
Marital.Status..at.the.time.of.application.
No.of.dependents
No.of.PL.trades.opened.in.last.12.months
No.of.PL.trades.opened.in.last.6.months
No.of.times.30.DPD.or.worse.in.last.12.months
No.of.times.30.DPD.or.worse.in.last.6.months
No.of.times.60.DPD.or.worse.in.last.12.months
No.of.times.60.DPD.or.worse.in.last.6.months
No.of.times.90.DPD.or.worse.in.last.12.months
No.of.times.90.DPD.or.worse.in.last.6.months
Presence.of.open.auto.loan
Profession
Predictors

Odds RATIO:

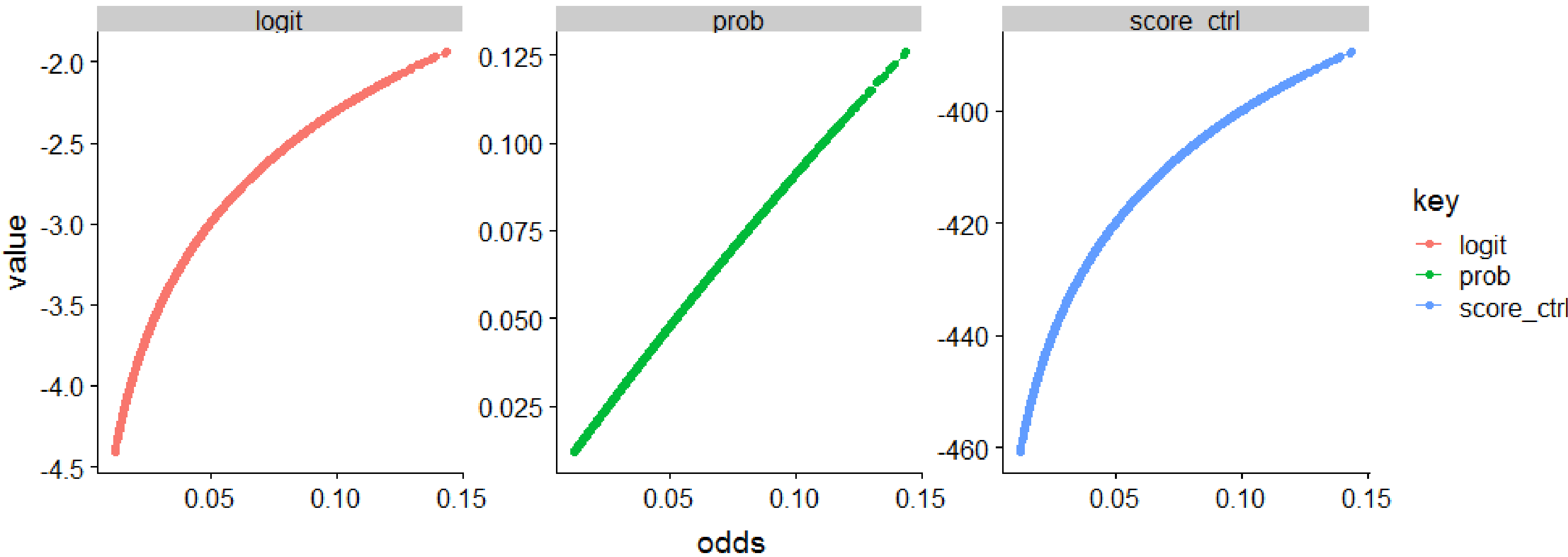


Odds Ratio Graph
0.67
0.64
0.81
0.04
0.19
0.69
0.28
Odds Ratio
0.8
0.6
0.4
0.2
0.0
binning.Avgas.CC.Utilization.in.last.12.months:WOE
binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE
binning.Outstanding.Balance:WOE
Intercept
No.of.dependents:WOE
No.of.times.30.DPD.or.worse.in.last.6.months:WOE
Profession:WOE
Predictors

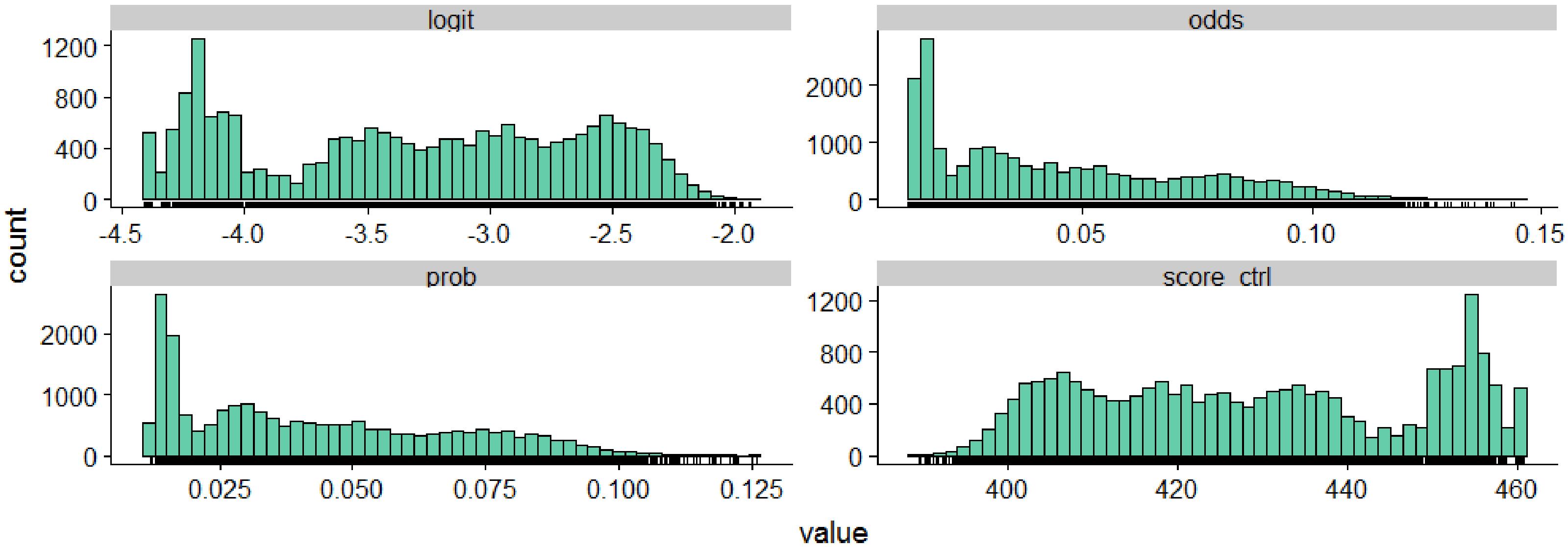
Observation - We can observe the score is perfectly linearly correlated with Logit

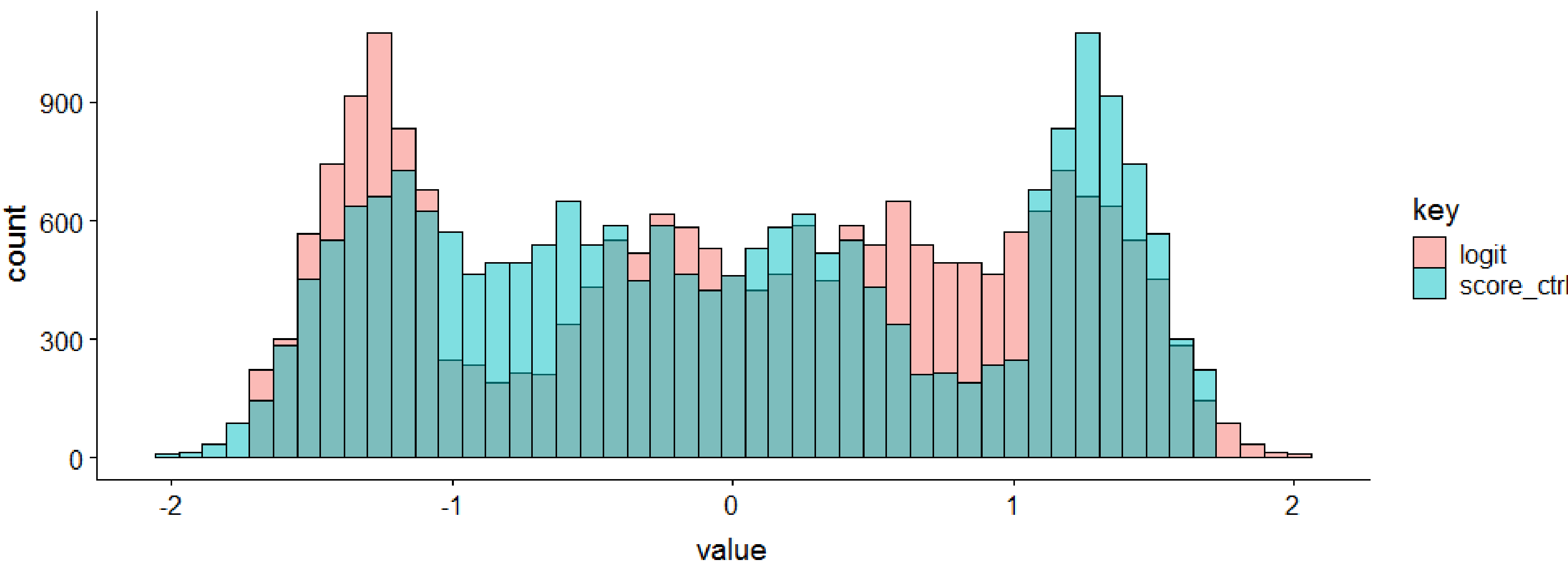


Observation - we can see the relationship between odds and score and odds and logit is identical

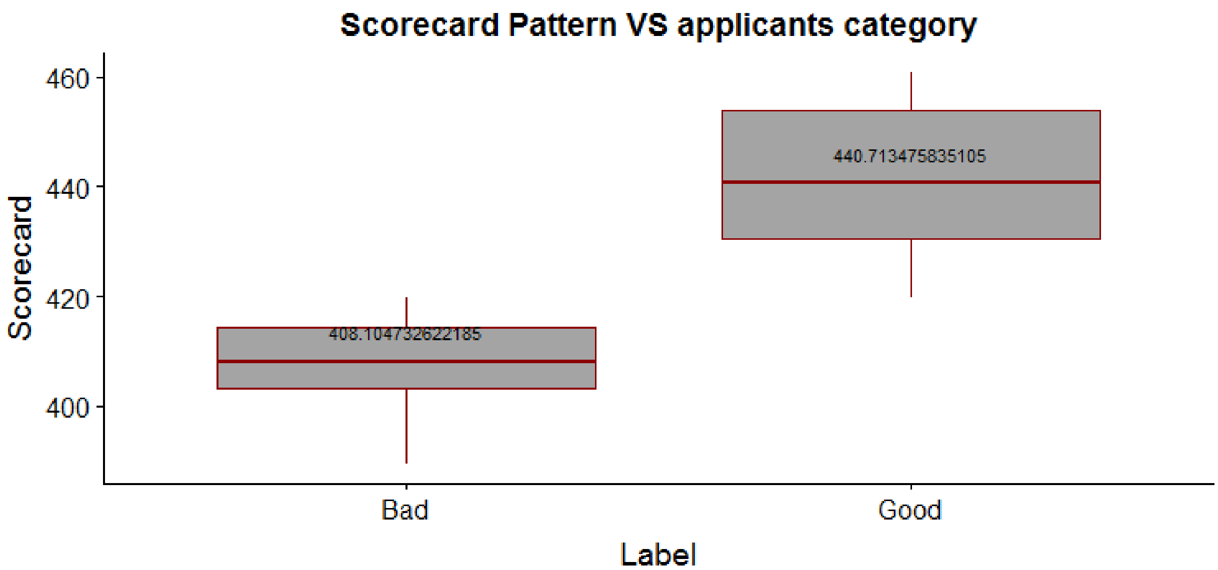


Observations - Score and Logit also have identical distributions





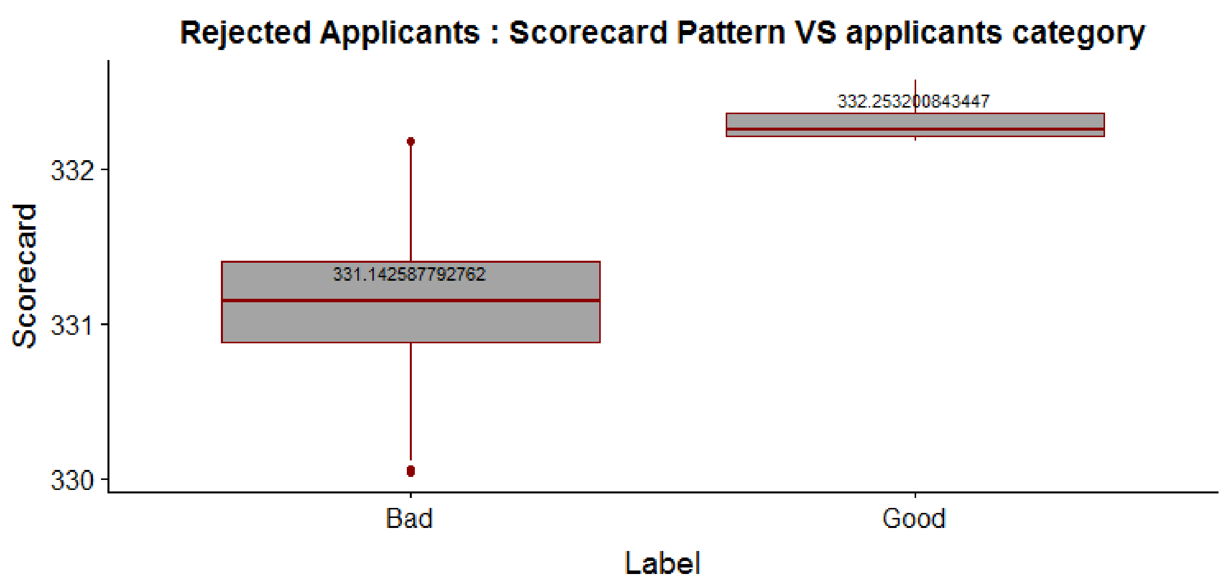
Important


Scorecard Pattern VS applicants category
Scorecard
460
440
420
400
440.713475835105
408.104732622185
Bad
Good
Label

Important

Rejected Applications:

We can clearly see most of the applicants are falling under Bad category.



R CODE AND THE RESULTS

```
##################### Data Analysis & Preprocessing  ##########################

# Setting the file location

# setwd("C:/DataScience_iiitb_upgrade/CapstoneProject")


# Load Data

 demographic_data<- read.csv("Demographic data.csv")

 creditbureau_data <- read.csv("Credit Bureau data.csv")


# Checking structure of dataset

str(demographic_data)

str(creditbureau_data)


# Summary of dataset

summary(demographic_data)

summary(creditbureau_data)


# Check whether no.of rows are equivalent in both files

nrow(demographic_data)

nrow(creditbureau_data)


# Check for the NA's

sapply(list(demographic_data,creditbureau_data), function(x) length(which(is.na(x))))

# There are total of 1428 and 3028 NAs

# Comments:  There are 1425 NAs for "performance tag". These are considered as REJECTED application. Since Peformance Tag is the Target
column and cannot contain

# NAs, these rows can be removed from the dataset but save seperately to do analysis of Rejected applications


# Check for blank rows

sapply(creditbureau_data, function(x) sum( trimws(x) == "",na.rm = TRUE))

sapply(demographic_data, function(x) sum( trimws(x) == "",na.rm = TRUE))


# Check Uniqueness of Application Id's

View(demographic_data[which(duplicated(demographic_data$Application.ID)),])

View(creditbureau_data[which(duplicated(creditbureau_data$Application.ID)),])
```

```
# Comments: There are 3 duplicate rows- Application ID's : 765011468, 653287861, 671989187

# The duplicate rows can be removed as they do not add any value to the analysis


# Remove the duplicate rows

demographic_data <- demographic_data[-which(duplicated(demographic_data$Application.ID)),]

creditbureau_data <- creditbureau_data[-which(duplicated(creditbureau_data$Application.ID)),]


# Check the no. of rows

nrow(demographic_data)            #71292

nrow(creditbureau_data)           #71292


# Merge the datasets to do further analysis

mergedfile <- merge(x = demographic_data, y = creditbureau_data, by = 'Application.ID', all=FALSE)


# Performance tag shoud have same value for both the table

nrow(mergedfile[which(mergedfile$Performance.Tag.y != mergedfile$Performance.Tag.x),])

# Since they both have same value and no conflict so remove one column

mergedfile <- subset(mergedfile,select=-c(Performance.Tag.x))


# Check the no. of rows after merging

nrow(mergedfile)                 #71292


nrow(mergedfile[which(is.na(mergedfile$Performance.Tag.y)),])

# Comments:  There are 1425 NAs for "performance tag". These are considered as REJECTED application. Since Peformance Tag is the Target column and cannot contain

# NAs, these rows can be removed from the dataset but save seperately to do analysis of Rejected applications


rec_without_perf_tag <- mergedfile[which(is.na(mergedfile$Performance.Tag.y)),]

mergedfile <- mergedfile[-which(is.na(mergedfile$Performance.Tag.y)),]


# There are NA values in some columns

# Check with columns have missing values

sum(is.na(mergedfile))

sapply(mergedfile,function(x) sum(is.na(x)))

#Presence.of.open.home.loan  <- 272, Outstanding.Balance <- 272 Avgas.CC.Utilization.in.last.12.months <- 1023
```

```
#Number of dependents <- 3, No.of.trades.opened.in.last.6.months <-
# Avgas.CC.Utilization.in.last.12.months
#Take backup of original mergedfile before starting the data processing
mergedfile_bk <- mergedfile
#Not removing the NA values as we are working with WOE and NA will be treated as a separate bucket
sum(is.na(mergedfile))
#mergedfile <- mergedfile[-which(is.na(mergedfile$No.of.dependents)),]
#mergedfile <- mergedfile[-which(is.na(mergedfile$Presence.of.open.home.loan)),]
#mergedfile <- mergedfile[-which(is.na(mergedfile$Avgas.CC.Utilization.in.last.12.months)),]



nrow(mergedfile)
sum(is.na(mergedfile))
# % of NAs rows : 1.5% - not removed
(1571/69867)*100

###################################################################################################################
######################################## UNIVARIATE Analysis ####################################################
###################################################################################################################
library(ggplot2)
library(dplyr)

## Categorical Univariate function
uv_categorical <- function(dataset,var,var_name,xname){
  dataset %>% ggplot(aes(x = as.factor(var))) +
    geom_bar(aes(y = (..count..)/sum(..count..))) +theme_bw()+
    geom_text(aes(y = ((..count..)/sum(..count..)), label = scales::percent((..count..)/sum(..count..))), stat = "count", vjust = -0.25) +
    scale_y_continuous(labels = scales::percent) +
    labs(title = var_name, y = "Percentage", x = xname)+theme(
      axis.text.x = element_text(hjust = 0.5, vjust = 0.5),
      plot.title = element_text(hjust = 0.5))
}

## Coontinuous Univariate Function
uv_continuous <- function(dataset,var,var_name){
```

```
  dataset %>% ggplot(aes(x = (var))) +
    geom_histogram(breaks=seq(min(var), max(var), by=1),col="red", aes(fill=..count..)) +
    scale_fill_gradient("Count", low="green", high="red")+
    labs(title = var_name, y = "Count", x = var_name)+theme(plot.title = element_text(hjust = 0.5))
}

uv_categorical(mergedfile,mergedfile$Gender,"Gender Distribution","Gender")
## 76.4% Males & 23.6% Female, No missing value
uv_categorical(mergedfile,mergedfile$Marital.Status..at.the.time.of.application.,"Marital status Distribution","Marital Status")
## 85.2% Married 15% Single , No missing value
uv_categorical(mergedfile,mergedfile$No.of.dependents,"dependents Distribution", "No of dependents")
## 21.8% 1 dependents, 21.7% 2 dependents, 22.4% 3 dependents, 17.2% 4 dependents, 17% 5 dependents, No missing value
uv_categorical(mergedfile,mergedfile$Education,"Education Distribution", "Education")
## 0.2% Missing Value, 24.8%,Bachelor, 33.6% Masters, 0.2% Others, 6.4% Phd, 34.9% Professional
uv_categorical(mergedfile,mergedfile$Profession,"Profession Distribution", "Profession")
## 56.8% SAL, 19.9% SE, 23.3% SE_PROF, No Missing Value
uv_categorical(mergedfile,mergedfile$Type.of.residence,"residence type Distribution", "Residence type")
## 2.3% Company Provided,2.5% living with parents,0.3% Others, 20% in owned house,74.8% are in rented house, No Missing value

#Check the target variable
## Frequency table of performance tags
table(mergedfile$Performance.Tag.y)
##  Only 4.22% of defaulters are captured in the data set,  95.78% non defaulter
prop.table(table(mergedfile$Performance.Tag.y)) * 100
uv_categorical(mergedfile,mergedfile$Performance.Tag.y,"Performance Distribution","Performance")
## 95.8% have 0's and only 4.2% have 1's

# Analyzing continuous variable
uv_continuous(mergedfile,mergedfile$Age,"Age Distribution")
## The distribution is high between 30 to 55
summary(mergedfile$Age)
#Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#-3      37      45      45      53      65

## Income Distribution
```

```
uv_continuous(mergedfile,mergedfile$Income,"Income Distribution")

## At around 5L there is a spike and distribution is almost same between 8-45

summary(mergedfile$Income)

#Min. 1st Qu.  Median    Mean 3rd Qu.    Max.

#-0.50   14.00   27.00   27.41   40.00   60.00


##  No of months in current residence

uv_continuous(mergedfile,mergedfile$No.of.months.in.current.residence,"Current residence Distribution")

## Between 6-7 months, there is a spike |

summary(mergedfile$No.of.months.in.current.residence)

#Min. 1st Qu.  Median    Mean 3rd Qu.    Max.

#6.00    6.00   10.00   34.61   61.00  126.00


uv_continuous(mergedfile,mergedfile$No.of.months.in.current.company,"Current company Distribution")

## There is a spike between 4-6 months

summary(mergedfile$No.of.months.in.current.company)

#Min. 1st Qu.  Median    Mean 3rd Qu.    Max.

#3.0    17.0    34.0    34.2    51.0   133.0


#######################################################################################################################

######################################## BIVARIATE Analysis #####################################################

#######################################################################################################################


## We will analyze the defaulter pattern(Performance.Tag.x == 1) w.r.t the features mentioned in the dataset.

## Frequency Bar plot function

plotbar_freq <- function (dataf,xcol,ycol)

{

  p <- ggplot(dataf,aes(x= factor(xcol),fill = factor(ycol))) +

    geom_bar(position = "fill") +

    theme(axis.text.x = element_text(hjust = 0.5, vjust = 0.5),

          plot.title = element_text(hjust = 0.5))+

    scale_y_continuous(labels = scales::percent_format(),breaks = seq(0,1,0.05),limits = c(0,1),sec.axis = sec_axis(~./10, name = "Percentage of Defaulter"))

}
```

```
plotbar_freq(mergedfile,mergedfile$Age,mergedfile$Performance.Tag.y) +
  labs(title="Age vs Performance Tag",x="Age",y="Count of Defaulters",fill = "Performance tag")

plotbar_freq(mergedfile,mergedfile$Income,mergedfile$Performance.Tag.y) +
  labs(title="Income vs Performance Tag",x="Income",y="Count of Defaulters",fill = "Performance tag")

# Same issue more records - No.of.months.in.current.company

plotbar_freq(mergedfile,mergedfile$No.of.trades.opened.in.last.6.months,mergedfile$Performance.Tag.y) +
  labs(title="No.of.trades.opened.in.last.6.months vs Performance Tag",x="No.of.trades.opened.in.last.6.months",y="Count of Defaulters",fill = "Performance tag")

plotbar_freq(mergedfile,mergedfile$No.of.trades.opened.in.last.12.months,mergedfile$Performance.Tag.y) +
  labs(title="No.of.trades.opened.in.last.12.months vs Performance Tag",x="No.of.trades.opened.in.last.12.months",y="Count of Defaulters",fill = "Performance tag")

# Same issue more records - Avgas.CC.Utilization.in.last.12.months

plotbar_freq(mergedfile,mergedfile$No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.,mergedfile$Performance.Tag.y) +
  labs(title="No.of.Inquiries.in.last.12.months..excluding.home...auto.loans. vs Performance Tag",x="No.of.trades.opened.in.last.12.months",y="Count of Defaulters",fill = "Performance tag")

plotbar_freq(mergedfile,mergedfile$Total.No.of.Trades,mergedfile$Performance.Tag.y) +
  labs(title="Total.No.of.Trades vs Performance Tag",x="Total.No.of.Trades",y="Count of Defaulters",fill = "Performance tag")

plotbar_freq(mergedfile,mergedfile$No.of.Inquiries.in.last.6.months..excluding.home...auto.loans.,mergedfile$Performance.Tag.y) +
  labs(title="No.of.Inquiries.in.last.6.months..excluding.home...auto.loans. vs Performance Tag",x="No.of.Inquiries.in.last.6.months..excluding.home...auto.loans.",y="Count of Defaulters",fill = "Performance tag")


###################################################################################################
####################################### Correlation Matrix #####################################
###################################################################################################

## Make copy of original data frame
mergedfile_corr <- mergedfile
```

```
## Lets convert all the columns to integer for correlations

mergedfile_corr[] <- lapply(mergedfile_corr,as.integer)


##install.packages("sjPlot")

##install.packages("snakecase")

##install.packages('TMB', type = 'source')


library(sjPlot)


mergedfile_corr <- mergedfile_corr[,-1]



set_theme(base = theme_classic(),  axis.title.size = 0,  geom.label.size = 10,

         axis.textsize.x = 1.1, axis.textsize.y = 1.1,axis.angle.x = 90)


# Demographic data correlation matrix

sjp.corr(data = mergedfile_corr[,c(1:10,28)],sort.corr = T,na.deletion = c("listwise", "pairwise"),

        corr.method = c("pearson", "spearman", "kendall"),wrap.labels = 40,title = "Demographic Correlation Matrix",decimals = 3,show.legend = TRUE,show.p = TRUE)


## Credit Bureau data correlation matrix- part 1

sjp.corr(data = mergedfile_corr[,c(11:20,28)],sort.corr = T,na.deletion = c("listwise", "pairwise"),

        corr.method = c("pearson", "spearman", "kendall"),wrap.labels = 40,title = "Credit data Correlation Matrix - 1",decimals = 3,show.legend = TRUE,show.p = TRUE)


## Credit Bureau data correlation matrix- part 1

sjp.corr(data = mergedfile_corr[,c(21:28)],sort.corr = T,na.deletion = c("listwise", "pairwise"),

        corr.method = c("pearson", "spearman", "kendall"),wrap.labels = 40,title = "Credit data Correlation Matrix - 2",decimals = 3,show.legend = TRUE,show.p = TRUE)


## Demographic vs Credit Bureau data correlation matrix- part 1

sjp.corr(data = mergedfile_corr[,c(1:5,11:20,28)],sort.corr = T,na.deletion = c("listwise", "pairwise"),

        corr.method = c("pearson", "spearman", "kendall"),wrap.labels = 40,title = "Credit and Demo data Correlation Matrix - 1",decimals = 3,show.legend = TRUE,show.p = TRUE)


## Demographic vs Credit Bureau data correlation matrix- part 2

sjp.corr(data = mergedfile_corr[,c(1:5,21:28)],sort.corr = T,na.deletion = c("listwise", "pairwise"),
```

```
        corr.method = c("pearson", "spearman", "kendall"),wrap.labels = 40,title = "Credit and Demo data Correlation Matrix - 2",decimals = 
3,show.legend = TRUE,show.p = TRUE)


## Demographic vs Credit Bureau data correlation matrix- part 3

sjp.corr(data = mergedfile_corr[,c(6:10,11:20,28)],sort.corr = T,na.deletion = c("listwise", "pairwise"),

        corr.method = c("pearson", "spearman", "kendall"),wrap.labels = 40,title = "Credit and Demo data Correlation Matrix - 3",decimals = 
3,show.legend = TRUE,show.p = TRUE)


## Demographic vs Credit Bureau data correlation matrix- part 4

sjp.corr(data = mergedfile_corr[,c(6:10,21:28)],sort.corr = T,na.deletion = c("listwise", "pairwise"),

        corr.method = c("pearson", "spearman", "kendall"),wrap.labels = 40,title = "Credit and Demo data Correlation Matrix - 4",decimals = 
3,show.legend = TRUE,show.p = TRUE)



#######################################################################################################################

######################################## WOE and IV Calculation (Manual) ############################################

#######################################################################################################################


# code for calculating WOE and IV

# Binning for non factor continuous variables - Age(Range- -3 - 65), Income(-0.50,60), no of months in current residence (6, 126)

# No of months in the current company(3, 133), Avgas. CC. Utilization in last 12 months(0, 113), outstanding balance(0, 5218801)

# No of inquiries in last 12 months(0, 20), No of trades open in last 12 months (0,44)


library(scales)


# Plotting after binning

percentage_plot <- function(dataset, var1, var2, var_name, var_factor)

{

  ana_total <- dataset %>%

    group_by((var1)) %>%

    summarize(count = n()) %>% mutate(percent = (count/sum(count))*100)

  ana <- dataset %>%

    group_by((var1), (var2)) %>%

    summarize(count = n()) %>% mutate(percent = (count/sum(count))*100)

  ana_1 <-  subset(ana, `(var2)` == 1)

  percentage_vector <- ana_1$percent
```

```
  p <- ggplot()

  p <- p + geom_bar(data=ana_total,aes(x=`(var1)`, y=percent ),stat="identity",fill="grey50") + xlab(paste(var_name, ' binning'))

  p <- p + geom_hline(aes(yintercept=(sum(ana_1$count)/sum(ana$count))*var_factor*100,color = "blue"), size=1.5)

  p <- p + geom_line(data=ana_1, aes(x=`(var1)`, y=percent*var_factor, color = "red"), group=1, size=1.5)

  p <- p + geom_point(data=ana_1, aes(x=`(var1)`, y=percent*var_factor), group=1)

  p <- p + geom_label(data=ana_1, aes(x=`(var1)`, y=percent*var_factor,label=round(percent,digits=1)),hjust= -0.5, vjust=-0.5)

  p <- p + scale_y_continuous(name = "Percentage of population", sec.axis = sec_axis(~./var_factor, name = "Percentage of Defaulter") )

  p <- p + ggtitle(paste(var_name , "Vs Deafult Rate")) + scale_color_manual(labels = c("Binning Percentage", "Population Percentage"), values = c("blue", "red"))

  p

}


mergedfile$binning.age <- as.factor(cut(mergedfile$Age, breaks = c(min(mergedfile$Age)-1, 30,41,50,53,57, max(mergedfile$Age)+1),right=T))

rec_without_perf_tag$binning.age <- as.factor(cut(rec_without_perf_tag$Age, breaks = c(min(rec_without_perf_tag$Age)-1, 30,41,50,53,57, max(rec_without_perf_tag$Age)+1),right=T))

percentage_plot(mergedfile, mergedfile$binning.age, mergedfile$Performance.Tag.y,"Age",5)

mergedfile <- subset(mergedfile,select=-c(Age))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(Age))



mergedfile$binning.income <- as.factor(cut(mergedfile$Income, breaks = c(-1, 4.5, 12, 23, 31, 48, 61,right=T)))

rec_without_perf_tag$binning.income <- as.factor(cut(rec_without_perf_tag$Income, breaks = c(-1, 4.5, 12, 23, 31, 48, 61,right=T)))

percentage_plot(mergedfile, mergedfile$binning.income, mergedfile$Performance.Tag.y,"Income",5)

mergedfile <- subset(mergedfile,select=-c(Income))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(Income))


mergedfile$binning.No.of.months.in.current.residence <- as.factor(cut(mergedfile$No.of.months.in.current.residence, breaks = c(0, 6, 9, 28, 49, 97, 127),right=T))

rec_without_perf_tag$binning.No.of.months.in.current.residence <- as.factor(cut(rec_without_perf_tag$No.of.months.in.current.residence, breaks = c(0, 6, 9, 28, 49, 97, 127),right=T))

percentage_plot(mergedfile,mergedfile$binning.No.of.months.in.current.residence,mergedfile$Performance.Tag.y,"No. of months in current residence",5)

mergedfile <- subset(mergedfile,select=-c(No.of.months.in.current.residence))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(No.of.months.in.current.residence))


mergedfile$binning.No.of.months.in.current.company <- as.factor(cut(mergedfile$No.of.months.in.current.company, breaks = c(2, 5, 12, 24, 36, 48, 60, 134),right=T))
```

```
rec_without_perf_tag$binning.No.of.months.in.current.company <- as.factor(cut(rec_without_perf_tag$No.of.months.in.current.company, breaks = c(2, 5, 12, 24, 36, 48, 60, 134),right=T))

percentage_plot(mergedfile,mergedfile$binning.No.of.months.in.current.company,mergedfile$Performance.Tag.y,"No. of months in current company",2)

mergedfile <- subset(mergedfile,select=-c(No.of.months.in.current.company))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(No.of.months.in.current.company))


mergedfile$binning.No.of.trades.opened.in.last.6.months <- as.factor(cut(mergedfile$No.of.trades.opened.in.last.6.months, breaks = c(-1,1,2,3,4,5,13),right=T))

rec_without_perf_tag$binning.No.of.trades.opened.in.last.6.months <- as.factor(cut(rec_without_perf_tag$No.of.trades.opened.in.last.6.months, breaks = c(-1,1,2,3,4,5,13),right=T))

percentage_plot(mergedfile,mergedfile$binning.No.of.trades.opened.in.last.6.months,mergedfile$Performance.Tag.y,"No.of.trades.opened.in.last.6.months",5)

mergedfile <- subset(mergedfile,select=-c(No.of.trades.opened.in.last.6.months))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(No.of.trades.opened.in.last.6.months))


mergedfile$binning.No.of.trades.opened.in.last.12.months <- as.factor(cut(mergedfile$No.of.trades.opened.in.last.12.months, breaks = c(-1,1,2,3,4,5,6,7,8,9,10,12,29),right=T))

rec_without_perf_tag$binning.No.of.trades.opened.in.last.12.months <- as.factor(cut(rec_without_perf_tag$No.of.trades.opened.in.last.12.months, breaks = c(-1,1,2,3,4,5,6,7,8,9,10,12,29),right=T))

percentage_plot(mergedfile,mergedfile$binning.No.of.trades.opened.in.last.12.months,mergedfile$Performance.Tag.y,"No.of.trades.opened.in.last.12.months",2)

mergedfile <- subset(mergedfile,select=-c(No.of.trades.opened.in.last.12.months))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(No.of.trades.opened.in.last.12.months))


mergedfile$binning.Avgas.CC.Utilization.in.last.12.months <- as.factor(cut(mergedfile$Avgas.CC.Utilization.in.last.12.months, breaks=c(NA,-1,8,9,11,14,21,37,71,114),right=T))

rec_without_perf_tag$binning.Avgas.CC.Utilization.in.last.12.months <- as.factor(cut(rec_without_perf_tag$Avgas.CC.Utilization.in.last.12.months, breaks=c(NA,-1,8,9,11,14,21,37,71,114),right=T))

percentage_plot(mergedfile,mergedfile$binning.Avgas.CC.Utilization.in.last.12.months,mergedfile$Performance.Tag.y,"Avgas.CC.Utilization.in.last.12.months",2)

mergedfile <- subset(mergedfile,select=-c(Avgas.CC.Utilization.in.last.12.months))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(Avgas.CC.Utilization.in.last.12.months))


mergedfile$binning.Outstanding.Balance <- as.factor(cut(mergedfile$Outstanding.Balance/10,breaks = c(-1, 6843, 25522, 38681, 58540,77423,97246,135730,297000,328231,550000), right=T))

rec_without_perf_tag$binning.Outstanding.Balance <- as.factor(cut(rec_without_perf_tag$Outstanding.Balance/10,breaks = c(-1, 6843, 25522, 38681, 58540,77423,97246,135730,297000,328231,550000), right=T))

percentage_plot(mergedfile,mergedfile$binning.Outstanding.Balance,mergedfile$Performance.Tag.y,"Outstanding.Balance",2)

mergedfile <- subset(mergedfile,select=-c(Outstanding.Balance))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(Outstanding.Balance))
```

```
mergedfile$binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans. <-
as.factor(cut(mergedfile$No.of.Inquiries.in.last.12.months..excluding.home...auto.loans., breaks = c(-1,0,1,3,4,5,8,9,21), right=T))

rec_without_perf_tag$binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans. <-
as.factor(cut(rec_without_perf_tag$No.of.Inquiries.in.last.12.months..excluding.home...auto.loans., breaks = c(-1,0,1,3,4,5,8,9,21), right=T))

percentage_plot(mergedfile,mergedfile$binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.,mergedfile$Performance.Tag.y
,"No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.",3)

mergedfile <- subset(mergedfile,select=-c(No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.))


mergedfile$binning.Total.No.of.Trades <- as.factor(cut(mergedfile$Total.No.of.Trades,breaks = c(0,1,2,3,4,5,6,7,8,9,10,11,19,45), right=T))

rec_without_perf_tag$binning.Total.No.of.Trades <- as.factor(cut(rec_without_perf_tag$Total.No.of.Trades,breaks =
c(0,1,2,3,4,5,6,7,8,9,10,11,19,45), right=T))

percentage_plot(mergedfile,mergedfile$binning.Total.No.of.Trades,mergedfile$Performance.Tag.y,"Total.No.of.Trades",1)

mergedfile <- subset(mergedfile,select=-c(Total.No.of.Trades))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(Total.No.of.Trades))


mergedfile$binning.No.of.Inquiries.in.last.6.months..excluding.home...auto.loans. <-
as.factor(cut(mergedfile$No.of.Inquiries.in.last.6.months..excluding.home...auto.loans.,breaks = c(-1,0,1,2,3,4,5,11),right=T))

rec_without_perf_tag$binning.No.of.Inquiries.in.last.6.months..excluding.home...auto.loans. <-
as.factor(cut(rec_without_perf_tag$No.of.Inquiries.in.last.6.months..excluding.home...auto.loans.,breaks = c(-1,0,1,2,3,4,5,11),right=T))

percentage_plot(mergedfile,mergedfile$binning.No.of.Inquiries.in.last.6.months..excluding.home...auto.loans.,mergedfile$Performance.Tag.y,"
No.of.Inquiries.in.last.6.months..excluding.home...auto.loans.",4)

mergedfile <- subset(mergedfile,select=-c(No.of.Inquiries.in.last.6.months..excluding.home...auto.loans.))

rec_without_perf_tag <- subset(rec_without_perf_tag,select=-c(No.of.Inquiries.in.last.6.months..excluding.home...auto.loans.))


mergedfile$No.of.dependents <- as.factor(mergedfile$No.of.dependents)

mergedfile$No.of.times.90.DPD.or.worse.in.last.6.months <- as.factor(mergedfile$No.of.times.90.DPD.or.worse.in.last.6.months)

mergedfile$No.of.times.60.DPD.or.worse.in.last.6.months <- as.factor(mergedfile$No.of.times.60.DPD.or.worse.in.last.6.months)

mergedfile$No.of.times.30.DPD.or.worse.in.last.6.months <- as.factor(mergedfile$No.of.times.30.DPD.or.worse.in.last.6.months)

mergedfile$No.of.times.90.DPD.or.worse.in.last.12.months <- as.factor(mergedfile$No.of.times.90.DPD.or.worse.in.last.12.months)

mergedfile$No.of.times.60.DPD.or.worse.in.last.12.months <- as.factor(mergedfile$No.of.times.60.DPD.or.worse.in.last.12.months)

mergedfile$No.of.times.30.DPD.or.worse.in.last.12.months <- as.factor(mergedfile$No.of.times.30.DPD.or.worse.in.last.12.months)

mergedfile$No.of.PL.trades.opened.in.last.6.months <- as.factor(mergedfile$No.of.PL.trades.opened.in.last.6.months)

mergedfile$No.of.PL.trades.opened.in.last.12.months <- as.factor(mergedfile$No.of.PL.trades.opened.in.last.12.months)

mergedfile$Presence.of.open.home.loan <- as.factor(mergedfile$Presence.of.open.home.loan)

mergedfile$Presence.of.open.auto.loan<- as.factor(mergedfile$Presence.of.open.auto.loan)
```

```
rec_without_perf_tag$No.of.dependents <- as.factor(rec_without_perf_tag$No.of.dependents)

rec_without_perf_tag$No.of.times.90.DPD.or.worse.in.last.6.months <-
as.factor(rec_without_perf_tag$No.of.times.90.DPD.or.worse.in.last.6.months)

rec_without_perf_tag$No.of.times.60.DPD.or.worse.in.last.6.months <-
as.factor(rec_without_perf_tag$No.of.times.60.DPD.or.worse.in.last.6.months)

rec_without_perf_tag$No.of.times.30.DPD.or.worse.in.last.6.months <-
as.factor(rec_without_perf_tag$No.of.times.30.DPD.or.worse.in.last.6.months)

rec_without_perf_tag$No.of.times.90.DPD.or.worse.in.last.12.months <-
as.factor(rec_without_perf_tag$No.of.times.90.DPD.or.worse.in.last.12.months)

rec_without_perf_tag$No.of.times.60.DPD.or.worse.in.last.12.months <-
as.factor(rec_without_perf_tag$No.of.times.60.DPD.or.worse.in.last.12.months)

rec_without_perf_tag$No.of.times.30.DPD.or.worse.in.last.12.months <-
as.factor(rec_without_perf_tag$No.of.times.30.DPD.or.worse.in.last.12.months)

rec_without_perf_tag$No.of.PL.trades.opened.in.last.6.months <- as.factor(rec_without_perf_tag$No.of.PL.trades.opened.in.last.6.months)

rec_without_perf_tag$No.of.PL.trades.opened.in.last.12.months <- as.factor(rec_without_perf_tag$No.of.PL.trades.opened.in.last.12.months)

rec_without_perf_tag$Presence.of.open.home.loan <- as.factor(rec_without_perf_tag$Presence.of.open.home.loan)

rec_without_perf_tag$Presence.of.open.auto.loan<- as.factor(rec_without_perf_tag$Presence.of.open.auto.loan)


# No Missing Value

sapply(mergedfile, function(x) length(which(is.na(x))))


# Code to calculate WOE and IV

library(tidyr)


woe_master <- function(df, df_rejected, total_events, total_non_events)

{

  merged_woe <- as.data.frame(matrix(0,nrow(mergedfile),ncol(mergedfile)-1))

  rejected_woe <- as.data.frame(matrix(0,nrow(rec_without_perf_tag),ncol(rec_without_perf_tag)-1))

  merged_iv <- as.data.frame(matrix(0,ncol(mergedfile),2))


  for(j in 1:ncol(df))

  {

    feature <- colnames(df[j])

    if(feature != "Performance.Tag.y")

    {

      total_iv_value <- 0

      woe_lookup <- df %>%

        group_by(df[,j],Performance.Tag.y) %>% count()
```

```
  woe_lookup <- spread(woe_lookup,Performance.Tag.y,n)
  woe_lookup[,2:3] <- sapply(woe_lookup[,2:3], function(x){replace(x, is.na(x), 0)})
  level_iv <- as.data.frame(matrix(0,nrow(woe_lookup),2))
  names(level_iv) <- c("Factor_Level","Value")
  names(woe_lookup)[1] <- 'Value'
  colnames(merged_iv) <- c("level","IV")
  for(i in 1:nrow(woe_lookup))
  {
    woe_value <- 0
    iv_value <- 0
    total_iv_value <- 0
    events <- as.integer(woe_lookup[i,'1'])
    non_events <- as.integer(woe_lookup[i,'0'])
    if(non_events != 0 & events !=0)
    {
      woe_value <- log(((non_events/total_non_events)/(events/total_events)),base=exp(1))
      iv_value <- ((non_events/total_non_events) - (events/total_events))*100*woe_value
      total_iv_value <- iv_value + total_iv_value
      level_iv$Factor_Level[i] <-  woe_lookup$Value[i]
      level_iv$Value <- iv_value
      merged_woe[which(df[,j] == woe_lookup$Value[i]),j] <- woe_value
      rejected_woe[which(as.character(df_rejected[,j]) == as.character(woe_lookup$Value[i])),j] <- woe_value
    }

  }
  names(merged_woe)[j] <- paste0(feature,'_WOE')
  names(rejected_woe)[j] <- paste0(feature,'_WOE')
  merged_iv$level[j] <- feature
  merged_iv$IV[j] <- total_iv_value
  #merged_iv[j,3] <- level_iv
}
else
{
  merged_woe[,j] <- df[,j]
  names(merged_woe)[j] <- paste0(feature)
```

```
      k <- j
    }
  }
  merged_iv <- merged_iv[-c(k),]
  return(list(merged_woe,rejected_woe,merged_iv))
}

total_events <- nrow(subset(mergedfile, mergedfile$Performance.Tag.y == 1 ))
total_non_events <- nrow(subset(mergedfile, mergedfile$Performance.Tag.y == 0 ))
woe_if_list <- woe_master(mergedfile[,-1],rec_without_perf_tag[,-1],total_events,total_non_events)
merged_woe <- woe_if_list[[1]]
View(merged_woe)
rejected_woe <- woe_if_list[[2]]
View(rejected_woe)
merged_iv <- woe_if_list[[3]]
View(merged_iv)


#####################################################################################################
############################## WOE and IV calculation Infopackage ######################################
#####################################################################################################

# WOE and IV calculation using existing package
# Create Information Value Table to identify the predictive power of the variables
library(Information)
mergedfile_1 <- mergedfile
# Changing the 0's with 1 and 1's with zero . Information package uses 1 as good and zero as bad
mergedfile_1$Performance.Tag.y <- mergedfile_1$Performance.Tag.y + 1
mergedfile_1$Performance.Tag.y[mergedfile_1$Performance.Tag.y == 2] <- 0
rec_without_perf_tag$Performance.Tag.y <- rec_without_perf_tag$Performance.Tag.y + 1
rec_without_perf_tag$Performance.Tag.y[rec_without_perf_tag$Performance.Tag.y == 2] <- 0

IV_table <- create_infotables(data = mergedfile_1[,-1],y = "Performance.Tag.y",parallel = TRUE)
head(IV_table$Summary)
```

```
# Variable for which IV more than 0.3 indicate strong predictors

#                                                Variable        IV

#23                        binning.Avgas.CC.Utilization.in.last.12.months 0.3105283

#14                             No.of.PL.trades.opened.in.last.12.months 0.2989814

#22                        binning.No.of.trades.opened.in.last.12.months 0.2966334

#25 binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans. 0.2955443

#26                                      binning.Total.No.of.Trades 0.2488475

#24                                     binning.Outstanding.Balance 0.2475730

library(AtConP)

# Create dataset for WOE analysis


woe_master <- DF.Replace.WOE(mergedfile_1[,-1],IV_table,"Performance.Tag.y")

woe_rejected <- DF.Replace.WOE(rec_without_perf_tag[,-1],IV_table,"Performance.Tag.y")

View(woe_master)

View(woe_rejected)




# woe values are matching with the woe values calculated by our code, using woe_master for further analysis

# Verify the distribuiton of each of the variables and predict the defaults


plot_infotables(IV_table,"binning.age")   # Age group between 51-53 shows high chances of defaulting

plot_infotables(IV_table,"Gender")   # MALEs shows high default rate

plot_infotables(IV_table,"Marital.Status..at.the.time.of.application.")  # Married people are more vulnerable to default

plot_infotables(IV_table,"No.of.dependents")  # Applicants having 2 Depends

plot_infotables(IV_table,"binning.income")  # Income between 49-60K

plot_infotables(IV_table,"Education") # Ph.d's

plot_infotables(IV_table,"Profession") # SALARIED professionals

plot_infotables(IV_table,"Type.of.residence")  # marked OTHERS. This doesn't give the right prediction as others can be anyone

plot_infotables(IV_table,"binning.No.of.months.in.current.residence")  # applicants residing between 6-9 yrs

plot_infotables(IV_table,"binning.No.of.months.in.current.company") # Applicants associated between 54-61 months

plot_infotables(IV_table,"No.of.times.90.DPD.or.worse.in.last.6.months") # Applicants whose DPD is 0

plot_infotables(IV_table,"No.of.times.60.DPD.or.worse.in.last.6.months") # Applicants whose DPD is 0

plot_infotables(IV_table,"No.of.times.30.DPD.or.worse.in.last.6.months") # Applicants whose DPD is 0

plot_infotables(IV_table,"binning.Avgas.CC.Utilization.in.last.12.months") # Whose credit card utilization is between 0-4 times
```

```
plot_infotables(IV_table,"binning.No.of.trades.opened.in.last.6.months") # Who hasn't done any trades in 6 months

plot_infotables(IV_table,"binning.No.of.trades.opened.in.last.12.months") # Who has done 1 trade in 12 month

plot_infotables(IV_table,"No.of.PL.trades.opened.in.last.12.months") # who opened 0 PL trades in last 12 months

plot_infotables(IV_table,"binning.No.of.Inquiries.in.last.6.months..excluding.home...auto.loans.") # Who do not have any inquiries

plot_infotables(IV_table,"binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.") # who do not have any inquiries

plot_infotables(IV_table,"Presence.of.open.home.loan") # who has home loan atleast 1

plot_infotables(IV_table,"binning.Outstanding.Balance") # Whose outstanding balance is between 0-7789

plot_infotables(IV_table,"binning.Total.No.of.Trades") # who has 1 trade

plot_infotables(IV_table,"Presence.of.open.auto.loan") # who has autoloan


# Revert 1 and 0 value of Performance Tag again to work on the model

woe_master$Performance.Tag.y <- woe_master$Performance.Tag.y + 1

woe_master$Performance.Tag.y[woe_master$Performance.Tag.y == 2] <- 0


woe_rejected$Performance.Tag.y <- woe_rejected$Performance.Tag.y + 1

woe_rejected$Performance.Tag.y[woe_rejected$Performance.Tag.y == 2] <- 0


woe_demographic <- woe_master[,c(1:6,17:20,28)]


library(colr)


woe_demographic <- csub(woe_demographic, "\\:", "_")


woe_rejected_demographic <- woe_rejected[,c(1:6,17:20,28)]


woe_rejected_demographic <- csub(woe_rejected_demographic, "\\:", "_")


#############################################################################################


########### Logistic- Demographic woe data ##############################################
```

```
#######################################################################################

library(GGally)

library(e1071)

library(lattice)

library(caret)

library(cowplot)

library(caTools)


## Copy dataset to another one to use in the model without impacting the main data

demographic_woe_model_data <- woe_demographic


## Quick data check before starting the model

sum(is.na(demographic_woe_model_data))

nrow(demographic_woe_model_data)



# splitting the data between train and test

set.seed(100)


indices_demo = sample.split(demographic_woe_model_data$Performance.Tag.y, SplitRatio = 0.7)


train_demo = demographic_woe_model_data[indices_demo,]


#test_actual <-merged_woe_model[indices,]


test_demo = demographic_woe_model_data[!(indices_demo),]





#########################################################################################################

############################  Logistic model with the merged woe ########################################

#########################################################################################################
```

```
merged_woe_model <- woe_master

sum(is.na(merged_woe_model))

nrow(merged_woe_model)


# splitting the data between train and test

set.seed(100)


# Add the application id

merged_woe_model$Application.ID <- mergedfile$Application.ID


indices_mearged = sample.split(merged_woe_model$Performance.Tag.y, SplitRatio = 0.7)


train_merged = merged_woe_model[indices_mearged,]


#test_actual <-merged_woe_model[indices,]


test_merged = merged_woe_model[!(indices_mearged),]



# Take backup and remove the application id column

test_bk <- test_merged

test_merged <- test_merged[,-29]

train_bk <- train_merged

train_merged <- train_merged[,-29]


##############################################################################################################

######################################### Logistic Regression Model #########################################

##############################################################################################################






##################################### Demographic Logistic model ############################################


modeldemo_1 = glm(Performance.Tag.y ~ ., data = train_demo, family = "binomial")
```

```
summary(modeldemo_1)

# Null deviance: 17100  on 48906  degrees of freedom

# Residual deviance: 16322  on 48879  degrees of freedom

# AIC: 16378


# Stepwise selection

library("MASS")

modeldemo_2<- stepAIC(modeldemo_1, direction="both")


summary(modeldemo_2)


# Call:

#   glm(formula = Performance.Tag.y ~ No.of.dependents_WOE + Education_WOE +

#        Profession_WOE + binning.age_WOE + binning.income_WOE + binning.No.of.months.in.current.residence_WOE +

#        binning.No.of.months.in.current.company_WOE, family = "binomial",

#      data = train_demo)

#

# Deviance Residuals:

#   Min      1Q   Median      3Q      Max

# -2.8808   0.2416   0.2758   0.3216   0.5289

#

# Coefficients:

#   Estimate Std. Error z value Pr(>|z|)

# (Intercept)                              3.12240    0.02291 136.268  < 2e-16 ***

#   No.of.dependents_WOE                       1.70079    0.44608   3.813 0.000137 ***

#   Education_WOE                              1.16225    0.72422   1.605 0.108529

# Profession_WOE                             1.48862    0.46605   3.194 0.001402 **

#   binning.age_WOE                            0.62023    0.40156   1.545 0.122451

# binning.income_WOE                         0.79706    0.11100   7.180 6.95e-13 ***

#   binning.No.of.months.in.current.residence_WOE  0.86098    0.07163  12.020  < 2e-16 ***

#   binning.No.of.months.in.current.company_WOE    0.92124    0.16516   5.578 2.44e-08 ***

#   ---

#   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#

# (Dispersion parameter for binomial family taken to be 1)
```

```
#
# Null deviance: 17100  on 48906  degrees of freedom
# Residual deviance: 16807  on 48899  degrees of freedom
# AIC: 16823
#
# Number of Fisher Scoring iterations: 6


modeldemo_3 <- glm(Performance.Tag.y ~ No.of.dependents_WOE
           + Profession_WOE
           + binning.income_WOE
           + binning.No.of.months.in.current.residence_WOE
           + binning.No.of.months.in.current.company_WOE
           , data = train_demo, family = "binomial")


summary(modeldemo_3)


# Call:
#   glm(formula = Performance.Tag.y ~ No.of.dependents_WOE + Profession_WOE +
#       binning.income_WOE + binning.No.of.months.in.current.residence_WOE +
#       binning.No.of.months.in.current.company_WOE, family = "binomial",
#     data = train_demo)
#
# Deviance Residuals:
#   Min      1Q   Median      3Q     Max
# -0.4887  -0.3211  -0.2757  -0.2422   2.8708
#
# Coefficients:
#   Estimate Std. Error  z value Pr(>|z|)
# (Intercept)                          -3.12233    0.02291 -136.308  < 2e-16 ***
#   No.of.dependents_WOE                   -1.70026    0.44610   -3.811 0.000138 ***
#   Profession_WOE                         -1.47890    0.46598   -3.174 0.001505 **
#   binning.income_WOE                     -0.80653    0.11073   -7.284 3.24e-13 ***
```

```
#   binning.No.of.months.in.current.residence_WOE -0.86257    0.07161  -12.045  < 2e-16 ***
#   binning.No.of.months.in.current.company_WOE   -0.91793    0.16517   -5.558 2.74e-08 ***
#   ---
#   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# (Dispersion parameter for binomial family taken to be 1)
#
# Null deviance: 17100  on 48906  degrees of freedom
# Residual deviance: 16811  on 48901  degrees of freedom
# AIC: 16823
#
# Number of Fisher Scoring iterations: 6

library(car)
vif(modeldemo_3)


# > vif(modeldemo_3)
# No.of.dependents_WOE                    Profession_WOE                    binning.income_WOE
# 1.000447                           1.000091                           1.016385
# binning.No.of.months.in.current.residence_WOE   binning.No.of.months.in.current.company_WOE
# 1.019149                           1.004139
# >



##################################################################################################
# Test and evaluation
##################################################################################################

# To keep the original Test dataset, which will be required later
test_actual <- test_demo

test <- test_actual[,-11]
```

```
View(test)

#

# View(test_actual)


###########################################################


final_model <- modeldemo_3


###############################################################################################################

##############################  Logistic  Regression  - Model Evaluation #############################################

###############################################################################################################


#predicted probabilities of default 1 for test data



test_pred = predict(final_model, type = "response",

                 newdata = test)


library(e1071)

View(test_pred)


# Let's see the summary


summary(test_pred)


# > summary(test_pred)

# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.

# 0.01231 0.03021 0.03848 0.04229 0.05170 0.12010


test$prob <- test_pred


# Let's use the probability cutoff of at 3rd quartile.


test_pred_default <- factor(ifelse(test_pred >= 0.051, "Yes", "No"))

test_actual_default <- factor(ifelse(test_actual$Performance.Tag.y==1,"Yes","No"))
```

```
summary(test_pred_default)

summary(test_actual_default)

table(test_actual_default,test_pred_default)

conf_final_demo <- confusionMatrix(test_pred_default, test_actual_default, positive = "Yes")

conf_final_demo

# Confusion Matrix and Statistics
#
# Reference
# Prediction    No   Yes
# No  15012   532
# Yes  5064   352
#
# Accuracy : 0.733
# 95% CI : (0.727, 0.739)
# No Information Rate : 0.9578
# P-Value [Acc > NIR] : 1
#
# Kappa : 0.0423
# Mcnemar's Test P-Value : <2e-16
#
# Sensitivity : 0.39819
# Specificity : 0.74776
# Pos Pred Value : 0.06499
# Neg Pred Value : 0.96577
# Prevalence : 0.04218
# Detection Rate : 0.01679
# Detection Prevalence : 0.25840
# Balanced Accuracy : 0.57297
#
# 'Positive' Class : Yes
```

```
### Although Accuracy is 73% and Specificity is around 75%, Sensitivity is less around 40%



# > summary(test_pred)
# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 0.01231 0.03021 0.03848 0.04229 0.05170 0.12010

# let's testwith in between values of mean and 3rd quartile'

#######################################################################
test_pred_default_1 <- factor(ifelse(test_pred >= 0.042, "Yes", "No"))

summary(test_pred_default_1)

table(test_actual_default,test_pred_default_1)

test_conf <- confusionMatrix(test_pred_default_1, test_actual_default, positive = "Yes")
test_conf

# Confusion Matrix and Statistics
#
# Reference
# Prediction    No   Yes
# No  11897   367
# Yes  8179   517
#
# Accuracy : 0.5923
# 95% CI : (0.5856, 0.5989)
# No Information Rate : 0.9578
# P-Value [Acc > NIR] : 1
#
# Kappa : 0.034
# Mcnemar's Test P-Value : <2e-16
```

```
#

# Sensitivity : 0.58484

# Specificity : 0.59260

# Pos Pred Value : 0.05945

# Neg Pred Value : 0.97008

# Prevalence : 0.04218

# Detection Rate : 0.02467

# Detection Prevalence : 0.41489

# Balanced Accuracy : 0.58872

#

# 'Positive' Class : Yes



### Accuracy dropped to 59% and Specificity is around 59%, Sensitivity is less around 58%


### This looks more of a stable cut-off



################################################################################


####################################################################################################

# Let's Choose the cutoff value.

#


# Let's find out the optimal probalility cutoff


perform_fn <- function(cutoff)

{

  predicted_default <- factor(ifelse(test_pred >= cutoff, "Yes", "No"))

  conf <- confusionMatrix(predicted_default, test_actual_default, positive = "Yes")

  acc <- conf$overall[1]

  sens <- conf$byClass[1]

  spec <- conf$byClass[2]

  out <- t(as.matrix(c(sens, spec, acc, cutoff)))

  colnames(out) <- c("sensitivity", "specificity", "accuracy", "cut-off")
```

```
  return(out)

}

# Creating cutoff values from 0.042 to 0.56 for plotting and initiallizing a matrix of 100 X 3.

# Summary of test probability

summary(test_pred)

s = seq(.040,.050,length=50)

OUT = matrix(0,50,4)

for(i in 1:50)

{

  OUT[i,] = perform_fn(s[i])

}

View(OUT)

plot(s, OUT[,1],xlab="Cutoff",ylab="Value",cex.lab=1.5,cex.axis=1.5,ylim=c(0,1),type="l",lwd=2,axes=FALSE,col=2)

axis(1,seq(0,1,length=5),seq(0,1,length=5),cex.lab=1.5)

axis(2,seq(0,1,length=5),seq(0,1,length=5),cex.lab=1.5)

lines(s,OUT[,2],col="darkgreen",lwd=2)

lines(s,OUT[,3],col=4,lwd=2)

box()

legend(0,.50,col=c(2,"darkgreen",4,"darkred"),lwd=c(2,2,2,2),c("Sensitivity","Specificity","Accuracy"))

# Let's choose the best cut-off values for a stable model

cutoff <- s[which(abs(OUT[,1]-OUT[,2])<0.005)]

# > cutoff <- s[which(abs(OUT[,1]-OUT[,2])<0.005)]
```

```
# > cutoff
# [1] 0.04183673

# Let's choose  the cutoff value

test_cutoff_default <- factor(ifelse(test_pred >= 0.04184, "Yes", "No"))

summary(test_cutoff_default)

table(test_actual_default,test_cutoff_default)

conf_final_demo <- confusionMatrix(test_cutoff_default, test_actual_default, positive = "Yes")

acc <- conf_final_demo$overall[1]

sens <- conf_final_demo$byClass[1]

spec <- conf_final_demo$byClass[2]

acc

sens

spec

# > acc
# Accuracy
# 0.5898855
# >
#   > sens
# Sensitivity
# 0.5871041
# >
#   > spec
# Specificity
```

```
# 0.590008
# >

##########################################################################################
################################# KS Statistics ############################################
##########################################################################################


test_cutoff_default <- ifelse(test_cutoff_default=="Yes",1,0)
test_cutoff_default

test_actual_default <- ifelse(test_actual_default=="Yes",1,0)


library(ROCR)
#on testing  data
pred_object_test<- prediction(test_cutoff_default, test_actual_default)

length(test_cutoff_default)==length(test_actual_default)

performance_measures_test<- performance(pred_object_test, "tpr", "fpr")

ks_table_test <- attr(performance_measures_test, "y.values")[[1]] -
  (attr(performance_measures_test, "x.values")[[1]])

max(ks_table_test)

# > max(ks_table_test)
# [1] 0.177112

##########################################################################################
################################# Lift & Gain Chart ############################################
##########################################################################################

# plotting the lift chart
```

```
# Loading dplyr package

require(dplyr)

library(dplyr)


lift <- function(labels , predicted_prob,groups=10) {


  if(is.factor(labels)) labels  <- as.integer(as.character(labels ))

  if(is.factor(predicted_prob)) predicted_prob <- as.integer(as.character(predicted_prob))

  helper = data.frame(cbind(labels , predicted_prob))

  helper[,"bucket"] = ntile(-helper[,"predicted_prob"], groups)

  gaintable = helper %>% group_by(bucket)  %>%

    summarise_at(vars(labels ), funs(total = n(),

                                     totalresp=sum(., na.rm = TRUE))) %>%


    mutate(Cumresp = cumsum(totalresp),

           Gain=Cumresp/sum(totalresp)*100,

           Cumlift=Gain/(bucket*(100/groups)))

  return(gaintable)

}


default_decile = lift(test_actual_default, test_pred, groups = 10)


# > default_decile

# # A tibble: 10 x 6

# bucket total totalresp Cumresp  Gain Cumlift

# <int> <int>     <dbl>   <dbl> <dbl>   <dbl>

#   1     1  2096      147     147  16.6    1.66

# 2     2  2096      122     269  30.4    1.52

# 3     3  2096      129     398  45.0    1.50

# 4     4  2096      107     505  57.1    1.43

# 5     5  2096       72     577  65.3    1.31

# 6     6  2096       83     660  74.7    1.24

# 7     7  2096       68     728  82.4    1.18

# 8     8  2096       56     784  88.7    1.11
```

```
# 9     9  2096       48     832  94.1    1.05

# 10    10  2096       52     884 100     1

##########################################################################################

################################ Odds ratio ############################################

##########################################################################################


## Lets consider the final model and calculate the odds ratio for each predcitor


exp(final_model$coefficients)


### The odds ratio explains the for every one unit increase in each of the predictors will

### the odds of having defaulters increases by the factor of the odds ratio value in the table.


# Example: For every 1 unit in crease of average cc untilization, the odds of having

#         defaults increases by a factor of 1.00.


### lets plot Odds ratio

odds_ratio_df <- data.frame(exp(final_model$coefficients))

odds_ratio_df$Predictors <- c("Intercept","No.of.dependents_WOE","Profession_WOE", "binning.income_WOE",
"binning.No.of.months.in.current.residence_WOE","binning.No.of.months.in.current.company_WOE")

odds_ratio_df$odds_ratio <- odds_ratio_df$exp.final_model.coefficients.

odds_ratio_df <- odds_ratio_df[,-1]


##### Odds RATIO PLOT

ggplot(odds_ratio_df)+

  geom_bar(aes(Predictors, odds_ratio),stat = "identity",fill = "light blue",colour = "black")+

  theme_bw()+theme(axis.text.x = element_text(hjust = 0, vjust = 1,angle = 330),plot.title = element_text(hjust = 0.5))+

  labs(title="Odds Ratio Graph",x ="Predictors",y ="Odds Ratio")+

  geom_text(aes(x = odds_ratio_df$Predictors,y = odds_ratio_df$odds_ratio, label = round(odds_ratio_df$odds_ratio,2), stat = "count", vjust = -
0.25))




########################################################################################################################
##

####################################################### Building Scorecard ############################################

########################################################################################################################
##
```

```
#install.packages("scorecard")
library(scorecard)

## Lets define a target

## Target Score value (ts) : 400
## Inverted target odds (to):  10

## Points to double the odds: 20

########### Formula

# score <- offset-factor* ln(odds) or score = offset-factor* logit values
# factor  <- pdo*ln(2)
# offset=ts-factor*ln(to)

#predicted probabilities of default 1 for test data
test_pred = predict(final_model, type = "response",newdata = test)
test_logit = predict(final_model, newdata = test)

res = tibble( logit = test_logit
            , odds = exp(test_logit)
            , prob = odds / (odds + 1))

#### Create a score card for Test data

## Target Score Value -
points0 <-  400

#Inverted Target Odds - at the target score 600 the ods should be 1:10
odds0 <- 10

#points to double the odds
pdo <-  20
```

```
factor <- pdo / log(2)

offset <- points0 -  factor * log( odds0)

res$score_ctrl = offset - factor * res$logit


View(res)

summary(res)


##Logit vs. Odds, Probabilities and Score --

## Observation - We can observe the score is perfectly linearly correlated with Logit

res %>%

  gather( key = 'key', value = 'value', - logit ) %>%

  ggplot( aes( logit, value, color = key) ) +

  geom_point() +

  geom_line() +

  facet_wrap(~key, scales = 'free_y')+xlab("Logit")+title("Logit Vs Other Parameters")


#### Odds vs. scaled Logit, Probabilities and Scores

### Observation - we can see the  relationship between odds and score and odds and logit is identical

res %>%

  mutate( score_ctrl = score_ctrl * - 1 ) %>%

  gather( key = 'key', value = 'value', - odds ) %>%

  ggplot( aes( odds, value, color = key) ) +

  geom_point() +

  geom_line() +

  facet_wrap(~key, scales = 'free_y')



#### Odds vs. scaled Logit, Probabilities and Scores in one graph

res %>%

  mutate( score_ctrl = score_ctrl * - 1 ) %>%

  mutate_at( vars(logit, prob, score_ctrl), scale ) %>%

  gather( key = 'key', value = 'value', - odds ) %>%

  ggplot( aes( odds, value, color = key) ) +

  geom_point( alpha = 0.5 ) +
```

```
  geom_line()



## Histogram graph - To check the patterns of logit, odds , prob and score distributions
## Observations - Score and Logit also have identical distributions
res %>%
  gather( key = 'key', value = 'value' ) %>%
  ggplot( aes(value) ) +
  geom_histogram( bins = 50
                , fill = 'aquamarine3'
                , color = 'black' ) +
  geom_rug()+
  facet_wrap(~key, scales = 'free')

## Scores and Logit in one graph
res %>% select(logit, score_ctrl) %>%
  mutate_all( scale, center = T) %>%
  mutate_all( as.vector ) %>%
  gather( key = 'key', value = 'value' ) %>%
  ggplot()+
  geom_histogram( aes( x = value, fill = key )
                , bins = 50
                , position="identity"
                , alpha = 0.5 )



#### The scorecard threshold value where the probability cutoff value lies
cutoff <- 0.04184
scorecard_thereshold <- data.frame((res[which((res$prob > cutoff)),]))
max(scorecard_thereshold$score_ctrl) ## lower this value, we will consider the applicants are bad(defaulted)

#### The threshold score above which applicant will be labelled good or bad which is exactly
#### equivalant to the cutoff calculated for probability threshold in logistic regression.
res$score_label <- ifelse(res$prob > cutoff,"Bad","Good")
```

```
## Check the data
View(res)

library(plyr)
## Lets compare the scorecard vs good/bad applicants
meds <- ddply(data.frame(res), .(score_label), summarise, med = median(score_ctrl))

### Plot to check the scorecards pattern among good and bad applicants
ggplot(data.frame(res), aes(x = score_label,y = score_ctrl)) +
  geom_boxplot(fill='#A4A4A4', color="darkred") +
  labs(title="Scorecard Pattern VS applicants category",x="Label",y="Scorecard")+
  geom_text(data = meds, aes(x = score_label, y = round(med,2), label = med),
        size = 3, vjust = -1.5)


##### Now, lets check the rejected data scorecard and see if it falls below the threshold.
#View(woe_rejected_demographic)
test_rejected = predict(final_model, type = "response",newdata = woe_rejected_demographic[,-11])
test_rejected_logit = predict(final_model, newdata = woe_rejected_demographic[,-11])

rejected_res = tibble( logit = test_rejected
                     , odds = exp(test_rejected_logit)
                     , prob = odds / (odds + 1))

#points to double the odds
pdo <-  20

factor <- pdo / log(2)
offset <- points0 -  factor * log( odds0)
rejected_res$score_ctrl = offset - factor * rejected_res$logit

#### The scorecard threshold value where the probability cutoff value lies
scorecard_thereshold <- data.frame((rejected_res[which((rejected_res$prob > cutoff)),]))
max(scorecard_thereshold$score_ctrl) ## lower this value, we will consider the applicants are bad(defaulted)
```

```
#### The threshold score above which applicant will be labelled good or bad which is exactly
#### equivalant to the cutoff calculated for probability threshold in logistic regression.
rejected_res$score_label <- ifelse(rejected_res$prob > cutoff,"Bad","Good")
rejected_res$Application.Id <- rec_without_perf_tag$Application.ID
View(rejected_res$score_label)

prop.table(table(rejected_res$score_label))

# > prop.table(table(rejected_res$score_label))
#
# Bad      Good
# 0.8063158 0.1936842

table(rejected_res$score_label)

# > table(rejected_res$score_label)
#
# Bad Good
# 1149   276

# As per the score, 81% rejected records are Bad




########################################################################################################################
#

####################################################################################################################

######################################### Complete Logistic model #########################################

####################################################################################################################
```

```
##########################################################################################################################
#

#Initial model

#train <- na.omit(train)


train <- train_merged


test <- test_merged


model_1 = glm(Performance.Tag.y ~ ., data = train, family = "binomial")

summary(model_1)

# Null deviance: 17100  on 48906  degrees of freedom

# Residual deviance: 16322  on 48879  degrees of freedom

# AIC: 16378


# Stepwise selection

library("MASS")

model_2<- stepAIC(model_1, direction="both")


summary(model_2)


# Null deviance: 17100  on 48906  degrees of freedom

# Residual deviance: 16333  on 48896  degrees of freedom

# AIC: 16355


# Coefficients:

#   Estimate Std. Error  z value Pr(>|z|)

# (Intercept)                                   -3.12187    0.02371 -131.666  < 2e-16 ***

#   `No.of.dependents:WOE`                                -1.64208    0.44753   -3.669 0.000243 ***

#   `Education:WOE`                               -1.12063    0.72800   -1.539 0.123724

# `Profession:WOE`                              -1.28037    0.46839   -2.734 0.006265 **

#   `No.of.times.90.DPD.or.worse.in.last.6.months:WOE`            0.15865    0.10857    1.461 0.143942

# `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`           -0.35932    0.08936   -4.021 5.80e-05 ***

#   `No.of.times.90.DPD.or.worse.in.last.12.months:WOE`           -0.17040    0.10132   -1.682 0.092614 .

# `binning.No.of.months.in.current.company:WOE`            -0.25338    0.16777   -1.510 0.130954
```

`binning.Avgas.CC.Utilization.in.last.12.months:WOE`   -0.37842   0.06201   -6.103 1.04e-09 ***

`binning.Outstanding.Balance:WOE`   -0.20389   0.06660   -3.061 0.002204 **

`binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE` -0.43865   0.06327   -6.933 4.13e-12 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Removing multicollinearity through VIF check

library(car)

vif(model_2)

`No.of.dependents:WOE`

1.000867

`Education:WOE`

1.000224

`Profession:WOE`

1.000421

`No.of.times.90.DPD.or.worse.in.last.6.months:WOE`

4.260631

`No.of.times.30.DPD.or.worse.in.last.6.months:WOE`

4.105105

`No.of.times.90.DPD.or.worse.in.last.12.months:WOE`

4.677653

`binning.No.of.months.in.current.company:WOE`

1.028466

`binning.Avgas.CC.Utilization.in.last.12.months:WOE`

2.066034

`binning.Outstanding.Balance:WOE`

1.794394

`binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`

1.722626

```
train$`binning.income:WOE`

# Taken
# `No.of.dependents:WOE`                                                -1.64208    0.44753   -3.669 0.000243 ***
# `Profession:WOE`                                                      -1.28037    0.46839   -2.734 0.006265 **
# `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`                    -0.35932    0.08936   -4.021 5.80e-05 ***
# `binning.Avgas.CC.Utilization.in.last.12.months:WOE`                  -0.37842    0.06201   -6.103 1.04e-09 ***
# `binning.Outstanding.Balance:WOE`                                     -0.20389    0.06660   -3.061 0.002204 **
# `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE` -0.43865    0.06327   -6.933 4.13e-12 ***

# Not taken
#   `Education:WOE`                                                     -1.12063    0.72800   -1.539 0.123724
#   `No.of.times.90.DPD.or.worse.in.last.6.months:WOE`                   0.15865    0.10857    1.461 0.143942
#   `No.of.times.90.DPD.or.worse.in.last.12.months:WOE`                 -0.17040    0.10132   -1.682 0.092614 .
#   `binning.No.of.months.in.current.company:WOE`                       -0.25338    0.16777   -1.510 0.130954

model_3 <- glm(Performance.Tag.y ~ `No.of.dependents:WOE`
        + `Profession:WOE`
        + `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`
        + `binning.Avgas.CC.Utilization.in.last.12.months:WOE`
        + `binning.Outstanding.Balance:WOE`
        + `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`
        , data = train, family = "binomial")

summary(model_3)

# Coefficients:
#   Estimate Std. Error  z value Pr(>|z|)
# (Intercept)                                                   -3.12164    0.02370 -131.710  < 2e-16 ***
#   `No.of.dependents:WOE`                                       -1.64797    0.44741   -3.683  0.00023 ***
```

```
#   `Profession:WOE`                                          -1.28401    0.46825   -2.742  0.00610 **
#   `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`                       -0.37804    0.05484   -6.893 5.46e-12 ***
#   `binning.Avgas.CC.Utilization.in.last.12.months:WOE`                     -0.39695    0.06133   -6.472 9.67e-11 ***
#   `binning.Outstanding.Balance:WOE`                                 -0.20856    0.06654   -3.134  0.00172 **
#   `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE` -0.44743    0.06317   -7.083 1.41e-12 ***
#   ---
#   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# (Dispersion parameter for binomial family taken to be 1)
#
# Null deviance: 17100  on 48906  degrees of freedom
# Residual deviance: 16340  on 48900  degrees of freedom
# AIC: 16354

vif(model_3)

# `No.of.dependents:WOE`
# 1.000716
# `Profession:WOE`
# 1.000297
# `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`
# 1.547495
# `binning.Avgas.CC.Utilization.in.last.12.months:WOE`
# 2.018213
# `binning.Outstanding.Balance:WOE`
# 1.790085
# `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`
# 1.712139

# Check the IV table importance if any variables are important which is missed out.

IV_table$Summary[order(-IV_table$Summary$IV),]
```

```
# Variable IV
# 23 binning.Avgas.CC.Utilization.in.last.12.months 3.105283e-01
# 14 No.of.PL.trades.opened.in.last.12.months 2.989814e-01
# 22 binning.No.of.trades.opened.in.last.12.months 2.966334e-01
# 25 binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans. 2.955443e-01
# 26 binning.Total.No.of.Trades 2.488475e-01
# 24 binning.Outstanding.Balance 2.475730e-01
# 9 No.of.times.30.DPD.or.worse.in.last.6.months 2.442369e-01
# 13 No.of.PL.trades.opened.in.last.6.months 2.242421e-01
# 12 No.of.times.30.DPD.or.worse.in.last.12.months 2.182230e-01
# 10 No.of.times.90.DPD.or.worse.in.last.12.months 2.156436e-01
# 8 No.of.times.60.DPD.or.worse.in.last.6.months 2.112635e-01
# 27 binning.No.of.Inquiries.in.last.6.months..excluding.home...auto.loans. 2.080964e-01
# 11 No.of.times.60.DPD.or.worse.in.last.12.months 1.881931e-01
# 21 binning.No.of.trades.opened.in.last.6.months 1.872886e-01
# 7 No.of.times.90.DPD.or.worse.in.last.6.months 1.626497e-01
# 19 binning.No.of.months.in.current.residence 9.790257e-02
# 18 binning.income 4.314981e-02
# 20 binning.No.of.months.in.current.company 1.934730e-02
# 15 Presence.of.open.home.loan 1.761939e-02
# 17 binning.age 3.209923e-03
# 3 No.of.dependents 2.653501e-03
# 5 Profession 2.219893e-03
# 16 Presence.of.open.auto.loan 1.658061e-03
# 6 Type.of.residence 9.198065e-04
# 4 Education 7.825416e-04
# 1 Gender 3.258695e-04
# 2 Marital.Status..at.the.time.of.application. 9.473857e-05

# The top 10 Important variables from IV table summary which is not part of model variables

# 14 No.of.PL.trades.opened.in.last.12.months 2.989814e-01
# 22 binning.No.of.trades.opened.in.last.12.months 2.966334e-01
# 26 binning.Total.No.of.Trades 2.488475e-01
# 13 No.of.PL.trades.opened.in.last.6.months 2.242421e-01
```

12 No.of.times.30.DPD.or.worse.in.last.12.months 2.182230e-01

10 No.of.times.90.DPD.or.worse.in.last.12.months 2.156436e-01

8 No.of.times.60.DPD.or.worse.in.last.6.months 2.112635e-01

Adding the important variables from IV table summary one by one in model

model_4 <- glm(Performance.Tag.y ~ `No.of.dependents:WOE`

+ `Profession:WOE`

+ `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`

+ `binning.Avgas.CC.Utilization.in.last.12.months:WOE`

+ `binning.Outstanding.Balance:WOE`

+ `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`

+ `No.of.PL.trades.opened.in.last.12.months:WOE`

, data = train, family = "binomial")

summary(model_4)

`No.of.dependents:WOE` -1.647932 0.447412 -3.683 0.00023 ***

`Profession:WOE` -1.284138 0.468270 -2.742 0.00610 **

`No.of.times.30.DPD.or.worse.in.last.6.months:WOE` -0.378113 0.054884 -6.889 5.60e-12 ***

`binning.Avgas.CC.Utilization.in.last.12.months:WOE` -0.397418 0.062801 -6.328 2.48e-10 ***

`binning.Outstanding.Balance:WOE` -0.210941 0.095968 -2.198 0.02795 *

`binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE` -0.448485 0.070284 -6.381 1.76e-10 ***

`No.of.PL.trades.opened.in.last.12.months:WOE` 0.003669 0.106739 0.034 0.97258

Not much significant the added variable, so let's check the multicolinearity

vif(model_4)

`No.of.dependents:WOE`

1.000722

`Profession:WOE`

1.000359

```
# `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`

# 1.549744

# `binning.Avgas.CC.Utilization.in.last.12.months:WOE`

# 2.116049

# `binning.Outstanding.Balance:WOE`

# 3.723450

# `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`

# 2.119707

# `No.of.PL.trades.opened.in.last.12.months:WOE`

# 5.295934


# multicolinearity increased for the added variable with outstanding balance, so we will remove the variable from the model.




# The top 10 Important variables from IV table summary which is not part of model variables and not already checked


# 22                    binning.No.of.trades.opened.in.last.12.months 2.966334e-01

# 26                                 binning.Total.No.of.Trades 2.488475e-01

# 13                        No.of.PL.trades.opened.in.last.6.months 2.242421e-01

# 12                    No.of.times.30.DPD.or.worse.in.last.12.months 2.182230e-01

# 10                    No.of.times.90.DPD.or.worse.in.last.12.months 2.156436e-01

# 8                     No.of.times.60.DPD.or.worse.in.last.6.months 2.112635e-01



model_5 <- glm(Performance.Tag.y ~ `No.of.dependents:WOE`

        + `Profession:WOE`

        + `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`

        + `binning.Avgas.CC.Utilization.in.last.12.months:WOE`

        + `binning.Outstanding.Balance:WOE`

        + `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`

        + `binning.No.of.trades.opened.in.last.12.months:WOE`

        , data = train, family = "binomial")


summary(model_5)
```

```
# (Intercept)                                                     -3.12176    0.02371 -131.673  < 2e-16 ***
#   `No.of.dependents:WOE`                                         -1.65146    0.44741   -3.691 0.000223 ***
#   `Profession:WOE`                                               -1.28254    0.46826   -2.739 0.006164 **
#   `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`             -0.36469    0.05557   -6.563 5.27e-11 ***
#   `binning.Avgas.CC.Utilization.in.last.12.months:WOE`           -0.38614    0.06173   -6.255 3.96e-10 ***
#   `binning.Outstanding.Balance:WOE`                              -0.15066    0.07785   -1.935 0.052956 .
#   `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE` -0.40645    0.06962   -5.838 5.29e-09 ***
#   `binning.No.of.trades.opened.in.last.12.months:WOE`            -0.11741    0.08282   -1.418 0.156323
```

Not much significant and outstanding balance become insignificant as well. So next model we will remove this.

```
vif(model_5)

# `No.of.dependents:WOE`
# 1.000744
# `Profession:WOE`
# 1.000294
# `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`
# 1.588415
# `binning.Avgas.CC.Utilization.in.last.12.months:WOE`
# 2.044647
# `binning.Outstanding.Balance:WOE`
# 2.455451
# `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`
# 2.070089
# `binning.No.of.trades.opened.in.last.12.months:WOE`
# 3.262532
```

```
# The top 10 Important variables from IV table summary which is not part of model variables and not already checked

# 26                         binning.Total.No.of.Trades 2.488475e-01
# 13                 No.of.PL.trades.opened.in.last.6.months 2.242421e-01
# 12              No.of.times.30.DPD.or.worse.in.last.12.months 2.182230e-01
# 10              No.of.times.90.DPD.or.worse.in.last.12.months 2.156436e-01
# 8               No.of.times.60.DPD.or.worse.in.last.6.months 2.112635e-01

model_6 <- glm(Performance.Tag.y ~ `No.of.dependents:WOE`
        + `Profession:WOE`
        + `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`
        + `binning.Avgas.CC.Utilization.in.last.12.months:WOE`
        + `binning.Outstanding.Balance:WOE`
        + `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`
        + `binning.Total.No.of.Trades:WOE`
        , data = train, family = "binomial")

summary(model_6)

# (Intercept)                                    -3.12161    0.02370 -131.706  < 2e-16 ***
#   `No.of.dependents:WOE`                         -1.64813    0.44741   -3.684  0.00023 ***
#   `Profession:WOE`                               -1.28413    0.46825   -2.742  0.00610 **
#   `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`          -0.37537    0.05559   -6.752 1.46e-11 ***
#   `binning.Avgas.CC.Utilization.in.last.12.months:WOE`        -0.39556    0.06151   -6.431 1.26e-10 ***
#   `binning.Outstanding.Balance:WOE`                  -0.20030    0.07228   -2.771  0.00558 **
#   `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE` -0.44045    0.06752   -6.523 6.90e-11 ***
#   `binning.Total.No.of.Trades:WOE`                  -0.02176    0.07442   -0.292  0.77002

# Not much significant at all, we will remove it.
```

```
vif(model_6)

# `No.of.dependents:WOE`
# 1.000717
# `Profession:WOE`
# 1.000297
# `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`
# 1.590017
# `binning.Avgas.CC.Utilization.in.last.12.months:WOE`
# 2.029488
# `binning.Outstanding.Balance:WOE`
# 2.113167
# `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`
# 1.954874
# `binning.Total.No.of.Trades:WOE`
# 2.298724



# The top 10 Important variables from IV table summary which is not part of model variables and not already checked

# 13                            No.of.PL.trades.opened.in.last.6.months 2.242421e-01
# 12                        No.of.times.30.DPD.or.worse.in.last.12.months 2.182230e-01
# 10                        No.of.times.90.DPD.or.worse.in.last.12.months 2.156436e-01
# 8                         No.of.times.60.DPD.or.worse.in.last.6.months 2.112635e-01



model_7 <- glm(Performance.Tag.y ~ `No.of.dependents:WOE`
          + `Profession:WOE`
          + `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`
          + `binning.Avgas.CC.Utilization.in.last.12.months:WOE`
          + `binning.Outstanding.Balance:WOE`
          + `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`
          + `No.of.PL.trades.opened.in.last.6.months:WOE`
          , data = train, family = "binomial")
```

```
summary(model_7)

# (Intercept)                                              -3.12174    0.02370 -131.697  < 2e-16 ***
#   `No.of.dependents:WOE`                                 -1.64845    0.44742   -3.684 0.000229 ***
#   `Profession:WOE`                                       -1.28286    0.46826   -2.740 0.006150 **
#   `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`     -0.37735    0.05484   -6.881 5.93e-12 ***
#   `binning.Avgas.CC.Utilization.in.last.12.months:WOE`   -0.39346    0.06158   -6.389 1.67e-10 ***
#   `binning.Outstanding.Balance:WOE`                      -0.18590    0.07699   -2.415 0.015747 *
#   `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE` -0.43819    0.06520   -6.721 1.80e-11 ***
#   `No.of.PL.trades.opened.in.last.6.months:WOE`          -0.04482    0.07697   -0.582 0.560371

# Not significant will remove

vif(model_7)

# `No.of.dependents:WOE`
# 1.000721
# `Profession:WOE`
# 1.000313
# `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`
# 1.547188
# `binning.Avgas.CC.Utilization.in.last.12.months:WOE`
# 2.034155
# `binning.Outstanding.Balance:WOE`
# 2.397877
# `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`
# 1.821549
# `No.of.PL.trades.opened.in.last.6.months:WOE`
# 2.245036
```

The top 10 Important variables from IV table summary which is not part of model variables and not already checked

12 No.of.times.30.DPD.or.worse.in.last.12.months 2.182230e-01

10 No.of.times.90.DPD.or.worse.in.last.12.months 2.156436e-01

8 No.of.times.60.DPD.or.worse.in.last.6.months 2.112635e-01

after testing one by one we found all are insignificant when `No.of.times.30.DPD.or.worse.in.last.6.months:WOE` is there in the model

so final model model 8 should be same as model 3

```
model_8 <- glm(Performance.Tag.y ~ `No.of.dependents:WOE`
    + `Profession:WOE`
    + `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`
    + `binning.Avgas.CC.Utilization.in.last.12.months:WOE`
    + `binning.Outstanding.Balance:WOE`
    + `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE`
    , data = train, family = "binomial")

summary(model_8)

# (Intercept)                                                                  -3.12164    0.02370 -131.710  < 2e-16 ***
#   `No.of.dependents:WOE`                                                     -1.64797    0.44741   -3.683  0.00023 ***
#   `Profession:WOE`                                                           -1.28401    0.46825   -2.742  0.00610 **
#   `No.of.times.30.DPD.or.worse.in.last.6.months:WOE`                         -0.37804    0.05484   -6.893 5.46e-12 ***
#   `binning.Avgas.CC.Utilization.in.last.12.months:WOE`                       -0.39695    0.06133   -6.472 9.67e-11 ***
#   `binning.Outstanding.Balance:WOE`                                          -0.20856    0.06654   -3.134  0.00172 **
#   `binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE` -0.44743    0.06317   -7.083 1.41e-12 ***

####################################################################################################
```

```
# Test and evaluation
###############################################################################################################

# To keep the original Test dataset, which will be required later
test_actual <- test

test <- test_actual[,-28]

#View(test)

View(test_actual)

####################################################################

final_model <- model_8

###############################################################################################################################
##############################  Logistic  Regression  - Model Evaluation ####################################################
###############################################################################################################################

#predicted probabilities of default 1 for test data


test_pred = predict(final_model, type = "response",
          newdata = test)

library(e1071)
View(test_pred)

# Let's see the summary

summary(test_pred)

# > summary(test_pred)
# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
# 0.01198 0.01748 0.03624 0.04229 0.06168 0.12587

test$prob <- test_pred

# Let's use the probability cutoff of at 3rd quartile.

test_pred_default <- factor(ifelse(test_pred >= 0.06, "Yes", "No"))
test_actual_default <- factor(ifelse(test_actual$Performance.Tag.y==1,"Yes","No"))

summary(test_pred_default)
summary(test_actual_default)

table(test_actual_default,test_pred_default)

test_conf <- confusionMatrix(test_pred_default, test_actual_default, positive = "Yes")
test_conf


# Confusion Matrix and Statistics
#
# Reference
# Prediction    No   Yes
# No  14992   456
# Yes  5084   428
#
# Accuracy : 0.7357
# 95% CI : (0.7297, 0.7416)
# No Information Rate : 0.9578
# P-Value [Acc > NIR] : 1
#
# Kappa : 0.0659
# Mcnemar's Test P-Value : <2e-16
#
# Sensitivity : 0.48416
# Specificity : 0.74676
```

Pos Pred Value : 0.07765

Neg Pred Value : 0.97048

Prevalence : 0.04218

Detection Rate : 0.02042

Detection Prevalence : 0.26298

Balanced Accuracy : 0.61546

#

'Positive' Class : Yes

Although Accuracy is 74% and Specificity is around 75%, Sensitivity is less around 48%

```
# > summary(test_pred)
# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 0.01198 0.01748 0.03624 0.04229 0.06168 0.12587

# let's testwith in between values of mean and 3rd quartile'

#####################################################################
test_pred_default_1 <- factor(ifelse(test_pred >= 0.052, "Yes", "No"))

summary(test_pred_default_1)

table(test_actual_default,test_pred_default_1)

test_conf <- confusionMatrix(test_pred_default_1, test_actual_default, positive = "Yes")
test_conf

# Confusion Matrix and Statistics
#
# Reference
# Prediction    No   Yes
# No  13688   382
# Yes  6388   502
```

```
#

# Accuracy : 0.677

# 95% CI : (0.6706, 0.6833)

# No Information Rate : 0.9578

# P-Value [Acc > NIR] : 1

#

# Kappa : 0.0588

# Mcnemar's Test P-Value : <2e-16

#

#            Sensitivity : 0.56787

#            Specificity : 0.68181

#         Pos Pred Value : 0.07286

#         Neg Pred Value : 0.97285

#             Prevalence : 0.04218

#         Detection Rate : 0.02395

#   Detection Prevalence : 0.32872

#      Balanced Accuracy : 0.62484

#

#       'Positive' Class : Yes


### Although Accuracy is 68% and Specificity is around 68%, Sensitivity is less around 57%


# > summary(test_pred)

# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.

# 0.01198 0.01748 0.03624 0.04229 0.06168 0.12587


# let's test with in between values of mean and 0.052


####################################################################################################

test_pred_default_2 <- factor(ifelse(test_pred >= 0.047, "Yes", "No"))


summary(test_pred_default_2)
```

table(test_actual_default,test_pred_default_2)

test_conf <- confusionMatrix(test_pred_default_2, test_actual_default, positive = "Yes")

test_conf

Confusion Matrix and Statistics

#

Reference

Prediction   No   Yes

No 12621   319

Yes 7455   565

#

Accuracy : 0.6291

95% CI : (0.6225, 0.6356)

No Information Rate : 0.9578

P-Value [Acc > NIR] : 1

#

Kappa : 0.0551

Mcnemar's Test P-Value : <2e-16

#

Sensitivity : 0.63914

Specificity : 0.62866

Pos Pred Value : 0.07045

Neg Pred Value : 0.97535

Prevalence : 0.04218

Detection Rate : 0.02696

Detection Prevalence : 0.38263

Balanced Accuracy : 0.63390

#

'Positive' Class : Yes

This looks more of a stable cut-off as Accuracy is 63% and Specificity is around 63%, Sensitivity is less around 64%

```
##############################################################################

##################################################################################################

# Let's Choose the cutoff value.

#

# Let's find out the optimal probalility cutoff

perform_fn <- function(cutoff)
{
  predicted_default <- factor(ifelse(test_pred >= cutoff, "Yes", "No"))
  conf <- confusionMatrix(predicted_default, test_actual_default, positive = "Yes")
  acc <- conf$overall[1]
  sens <- conf$byClass[1]
  spec <- conf$byClass[2]
  out <- t(as.matrix(c(sens, spec, acc, cutoff)))
  colnames(out) <- c("sensitivity", "specificity", "accuracy", "cut-off")
  return(out)
}

# Creating cutoff values from 0.042 to 0.56 for plotting and initiallizing a matrix of 100 X 3.

# Summary of test probability

summary(test_pred)



s = seq(.042,.056,length=50)

OUT = matrix(0,50,4)



for(i in 1:50)
```

```
{
  OUT[i,] = perform_fn(s[i])
}

View(OUT)
View(s)

plot(s, OUT[,1],xlab="Cutoff",ylab="Value",cex.lab=1.5,cex.axis=1.5,ylim=c(0,1),type="l",lwd=2,axes=FALSE,col=2)
axis(1,seq(0,1,length=5),seq(0,1,length=5),cex.lab=1.5)
axis(2,seq(0,1,length=5),seq(0,1,length=5),cex.lab=1.5)
lines(s,OUT[,2],col="darkgreen",lwd=2)
lines(s,OUT[,3],col=4,lwd=2)
box()
legend(0,.50,col=c(2,"darkgreen",4,"darkred"),lwd=c(2,2,2,2),c("Sensitivity","Specificity","Accuracy"))

# Let's choose the best cut-off values for a stable model

cutoff <- s[which(abs(OUT[,1]-OUT[,2])<0.005)]

# > cutoff <- s[which(abs(OUT[,1]-OUT[,2])<0.005)]
# > cutoff
# [1] 0.04742857 0.04771429

# Let's choose both the cutoff value correct till 3 decimal

test_cutoff_default <- factor(ifelse(test_pred >= 0.048, "Yes", "No"))

summary(test_cutoff_default)

table(test_actual_default,test_cutoff_default)

conf_final <- confusionMatrix(test_cutoff_default, test_actual_default, positive = "Yes")

acc <- conf_final$overall[1]
```

```
sens <- conf_final$byClass[1]

spec <- conf_final$byClass[2]

acc

sens

spec

# > acc
# Accuracy
# 0.6373569
# >
#   > sens
# Sensitivity
# 0.6289593
# >
#   > spec
# Specificity
# 0.6377266

###########################################################################################
################################# KS Statistics #############################################
###########################################################################################


test_cutoff_default <- ifelse(test_cutoff_default=="Yes",1,0)
test_cutoff_default

test_actual_default <- ifelse(test_actual_default=="Yes",1,0)


library(ROCR)
#on testing  data
```

```
pred_object_test<- prediction(test_cutoff_default, test_actual_default)

length(test_cutoff_default)==length(test_actual_default)

performance_measures_test<- performance(pred_object_test, "tpr", "fpr")

ks_table_test <- attr(performance_measures_test, "y.values")[[1]] -
  (attr(performance_measures_test, "x.values")[[1]])

max(ks_table_test)

# > max(ks_table_test)
# [1] 0.2666859

##########################################################################################
################################# Lift & Gain Chart ############################################
##########################################################################################

# plotting the lift chart

# Loading dplyr package
require(dplyr)
library(dplyr)

lift <- function(labels , predicted_prob,groups=10) {

  if(is.factor(labels)) labels  <- as.integer(as.character(labels ))
  if(is.factor(predicted_prob)) predicted_prob <- as.integer(as.character(predicted_prob))
  helper = data.frame(cbind(labels , predicted_prob))
  helper[,"bucket"] = ntile(-helper[,"predicted_prob"], groups)
  gaintable = helper %>% group_by(bucket)  %>%
    summarise_at(vars(labels ), funs(total = n(),
                                    totalresp=sum(., na.rm = TRUE))) %>%

    mutate(Cumresp = cumsum(totalresp),
```

```
        Gain=Cumresp/sum(totalresp)*100,
        Cumlift=Gain/(bucket*(100/groups)))
  return(gaintable)
}

default_decile = lift(test_actual_default, test_pred, groups = 10)

# > default_decile
# # A tibble: 10 x 6
# bucket total totalresp Cumresp  Gain Cumlift
# <int> <int>     <dbl>   <dbl> <dbl>   <dbl>
# 1      1  2096      181     181  20.5    2.05
# 2      2  2096      168     349  39.5    1.97
# 3      3  2096      116     465  52.6    1.75
# 4      4  2096      118     583  66.0    1.65
# 5      5  2096       96     679  76.8    1.54
# 6      6  2096       56     735  83.1    1.39
# 7      7  2096       59     794  89.8    1.28
# 8      8  2096       35     829  93.8    1.17
# 9      9  2096       26     855  96.7    1.07
# 10    10  2096       29     884 100      1

############################################################################################
################################ Odds ratio ###########################################
############################################################################################

## Lets consider the final model and calculate the odds ratio for each predcitor

exp(final_model$coefficients)

### The odds ratio explains the for every one unit increase in each of the predictors will
### the odds of having defaulters increases by the factor of the odds ratio value in the table.

# Example: For every 1 unit in crease of average cc untilization, the odds of having
#         defaults increases by a factor of 1.00.
```

```
### lets plot Odds ratio

odds_ratio_df <- data.frame(exp(final_model$coefficients))

odds_ratio_df$Predictors <- c("Intercept","No.of.dependents:WOE","Profession:WOE", "No.of.times.30.DPD.or.worse.in.last.6.months:WOE","binning.Avgas.CC.Utilization.in.last.12.months:WOE","binning.Outstanding.Balance:WOE","binning.No.of.Inquiries.in.last.12.months..excluding.home...auto.loans.:WOE")

odds_ratio_df$odds_ratio <- odds_ratio_df$exp.final_model.coefficients.

odds_ratio_df <- odds_ratio_df[,-1]


##### Odds RATIO PLOT

ggplot(odds_ratio_df)+

  geom_bar(aes(Predictors, odds_ratio),stat = "identity",fill = "light blue",colour = "black")+

  theme_bw()+theme(axis.text.x = element_text(hjust = 0, vjust = 1,angle = 330),plot.title = element_text(hjust = 0.5))+

  labs(title="Odds Ratio Graph",x ="Predictors",y ="Odds Ratio")+

  geom_text(aes(x = odds_ratio_df$Predictors,y = odds_ratio_df$odds_ratio, label = round(odds_ratio_df$odds_ratio,2), stat = "count", vjust = -0.25))



#######################################################################################################################

####################################################### Building Scorecard ###########################################

#######################################################################################################################


#install.packages("scorecard")

library(scorecard)


## Lets define a target


## Target Score value (ts) : 400

## Inverted target odds (to):  10


## Points to double the odds: 20


########### Formula


# score <- offset-factor* ln(odds) or score = offset-factor* logit values
```

```
# factor  <- pdo*ln(2)

# offset=ts-factor*ln(to)


#predicted probabilities of default 1 for test data

test_pred = predict(final_model, type = "response",newdata = test)

test_logit = predict(final_model, newdata = test)


res = tibble( logit = test_logit

            , odds = exp(test_logit)

            , prob = odds / (odds + 1))


#### Create a score card for Test data


## Target Score Value -

points0 <-  400


#Inverted Target Odds - at the target score 600 the ods should be 1:10

odds0 <- 10


#points to double the odds

pdo <-  20


factor <- pdo / log(2)

offset <- points0 -  factor * log( odds0)

res$score_ctrl = offset - factor * res$logit


View(res)

summary(res)


##Logit vs. Odds, Probabilities and Score --

## Observation - We can observe the score is perfectly linearly correlated with Logit

res %>%

  gather( key = 'key', value = 'value', - logit ) %>%

  ggplot( aes( logit, value, color = key) ) +

  geom_point() +
```

```
  geom_line() +
  facet_wrap(~key, scales = 'free_y')+xlab("Logit")+title("Logit Vs Other Parameters")


#### Odds vs. scaled Logit, Probabilities and Scores
### Observation - we can see the  relationship between odds and score and odds and logit is identical
res %>%
  mutate( score_ctrl = score_ctrl * - 1 ) %>%
  gather( key = 'key', value = 'value', - odds ) %>%
  ggplot( aes( odds, value, color = key) ) +
  geom_point() +
  geom_line() +
  facet_wrap(~key, scales = 'free_y')



#### Odds vs. scaled Logit, Probabilities and Scores in one graph
res %>%
  mutate( score_ctrl = score_ctrl * - 1 ) %>%
  mutate_at( vars(logit, prob, score_ctrl), scale ) %>%
  gather( key = 'key', value = 'value', - odds ) %>%
  ggplot( aes( odds, value, color = key) ) +
  geom_point( alpha = 0.5 ) +
  geom_line()



## Histogram graph - To check the patterns of logit, odds , prob and score distributions
## Observations - Score and Logit also have identical distributions
res %>%
  gather( key = 'key', value = 'value' ) %>%
  ggplot( aes(value) ) +
  geom_histogram( bins = 50
                , fill = 'aquamarine3'
                , color = 'black' ) +
  geom_rug()+
  facet_wrap(~key, scales = 'free')
```

```
## Scores and Logit in one graph

res %>% select(logit, score_ctrl) %>%

  mutate_all( scale, center = T) %>%

  mutate_all( as.vector ) %>%

  gather( key = 'key', value = 'value' ) %>%

  ggplot()+

  geom_histogram( aes( x = value, fill = key )

                , bins = 50

                , position="identity"

                , alpha = 0.5 )



#### The scorecard threshold value where the probability cutoff value lies

cutoff <- 0.048

scorecard_thereshold <- data.frame((res[which((res$prob > cutoff)),]))

max(scorecard_thereshold$score_ctrl) ## lower this value, we will consider the applicants are bad(defaulted)


#### The threshold score above which applicant will be labelled good or bad which is exactly

#### equivalant to the cutoff calculated for probability threshold in logistic regression.

res$score_label <- ifelse(res$prob > cutoff,"Bad","Good")


## Check the data

View(res)


library(plyr)

## Lets compare the scorecard vs good/bad applicants

meds <- ddply(data.frame(res), .(score_label), summarise, med = median(score_ctrl))


### Plot to check the scorecards pattern among good and bad applicants

ggplot(data.frame(res), aes(x = score_label,y = score_ctrl)) +

  geom_boxplot(fill='#A4A4A4', color="darkred") +

  labs(title="Scorecard Pattern VS applicants category",x="Label",y="Scorecard")+

  geom_text(data = meds, aes(x = score_label, y = round(med,2), label = med),

           size = 3, vjust = -1.5)
```

```
##### Now, lets check the rejected data scorecard and see if it falls below the threshold.

#View(woe_rejected)

test_rejected = predict(final_model, type = "response",newdata = woe_rejected[,-28])

test_rejected_logit = predict(final_model, newdata = woe_rejected[,-28])


rejected_res = tibble( logit = test_rejected

                    , odds = exp(test_rejected_logit)

                    , prob = odds / (odds + 1))


#points to double the odds

pdo <-  20


factor <- pdo / log(2)

offset <- points0 -  factor * log( odds0)

rejected_res$score_ctrl = offset - factor * rejected_res$logit


#### The scorecard threshold value where the probability cutoff value lies

scorecard_thereshold <- data.frame((rejected_res[which((rejected_res$prob > cutoff)),]))

max(scorecard_thereshold$score_ctrl) ## lower this value, we will consider the applicants are bad(defaulted)


#### The threshold score above which applicant will be labelled good or bad which is exactly

#### equivalant to the cutoff calculated for probability threshold in logistic regression.

rejected_res$score_label <- ifelse(rejected_res$prob > cutoff,"Bad","Good")

rejected_res$Application.Id <- rec_without_perf_tag$Application.ID

View(rejected_res$score_label)


prop.table(table(rejected_res$score_label))


# > prop.table(table(rejected_res$score_label))

#

# Bad      Good

# 0.9845614 0.0154386


table(rejected_res$score_label)
```

```
# > table(rejected_res$score_label)
#
# Bad Good
# 1403   22

# As per the score, 98.5% rejected records are Bad

###################################################################################################################
#######################################  Decision Tree Starts ####################################################
###################################################################################################################

library(foreach)
library(doParallel)
registerDoSEQ()



### Data preparation for decision tree and random forest model without Woe
mergedfile_DT_withoutwoe <- mergedfile_bk
sapply(mergedfile_DT_withoutwoe, function(x) sum( trimws(x) == "",na.rm = TRUE))
sum(is.na(mergedfile_DT_withoutwoe))

sapply(mergedfile_DT_withoutwoe,function(x) sum(is.na(x)))

#removing NA's
mergedfile_DT_withoutwoe <- mergedfile_DT_withoutwoe[-which(is.na(mergedfile_DT_withoutwoe$No.of.dependents)),]
mergedfile_DT_withoutwoe <- mergedfile_DT_withoutwoe[-which(is.na(mergedfile_DT_withoutwoe$Avgas.CC.Utilization.in.last.12.months)),]
mergedfile_DT_withoutwoe<-mergedfile_DT_withoutwoe[!is.na(mergedfile_DT_withoutwoe$No.of.trades.opened.in.last.6.months), ]
mergedfile_DT_withoutwoe <- mergedfile_DT_withoutwoe[!is.na(mergedfile_DT_withoutwoe$Presence.of.open.home.loan),]
sum(is.na(mergedfile_DT_withoutwoe))

mergedfile_DT_withoutwoe_with_app_id <- mergedfile_DT_withoutwoe

mergedfile_DT_withoutwoe <- mergedfile_DT_withoutwoe[,-1]
```

```
mergedfile_DT <- woe_master

prop.table(table(mergedfile_DT$Performance.Tag.y))  # 0.96

prop.table(table(mergedfile_DT_withoutwoe$Performance.Tag.y))  # 0.96

############ As the data is imbalanced we need to take certain steps to make sure the model is not biased ##############

############## Under sampling technique  #####################

tag_0_dataset <- mergedfile_DT[-which(mergedfile$Performance.Tag.y==1),]
tag_1_dataset <- mergedfile_DT[which(mergedfile$Performance.Tag.y==1),]

tag_0_dataset_original <- mergedfile_DT_withoutwoe[-which(mergedfile_DT_withoutwoe$Performance.Tag.y==1),]
tag_1_dataset_original <- mergedfile_DT_withoutwoe[which(mergedfile_DT_withoutwoe$Performance.Tag.y==1),]

############################ sampling technique ####################

library(ROSE)

nrow(mergedfile_DT)

# > nrow(mergedfile_DT)
# [1] 69867

nrow(tag_1_dataset)

# > nrow(tag_1_dataset)
# [1] 2947

nrow(tag_0_dataset)

# > nrow(tag_0_dataset)
```

```
# [1] 66920

nrow(mergedfile_DT_withoutwoe)
nrow(tag_1_dataset_original)
nrow(tag_0_dataset_original)

# > nrow(mergedfile_DT_withoutwoe)
# [1] 68841
# > nrow(tag_1_dataset_original)
# [1] 2899
# > nrow(tag_0_dataset_original)
# [1] 65942



## Changing the col names, replacing ":" with "_" for woe data
library(colr)
mergedfile_DT <- csub(mergedfile_DT, "\\:", "_")

##########  Sampling with different methods of balancing for woe

### data balancing method = over
data_balanced_over <- ovun.sample(as.factor(Performance.Tag.y)  ~ ., data = mergedfile_DT , method = "over",N = 139734)$data
table(data_balanced_over$Performance.Tag.y)

### data balancing method = under
data_balanced_under <- ovun.sample(as.factor(Performance.Tag.y)  ~ ., data = mergedfile_DT , method = "under",N = 5894)$data
table(data_balanced_under$Performance.Tag.y)

### data balancing method = both
data_balanced_both <- ovun.sample(as.factor(Performance.Tag.y)  ~ ., data = mergedfile_DT , method = "both", p=0.5, N=69867, seed = 1)$data
table(data_balanced_both$Performance.Tag.y)

### data balancing method = synthesis
```

```
data_balanced_synthesis <- ovun.sample(as.factor(Performance.Tag.y)  ~ ., data = mergedfile_DT , seed = 1)$data

table(data_balanced_synthesis$Performance.Tag.y)


prop.table(table(data_balanced_both$Performance.Tag.y))


prop.table(table(data_balanced_under$Performance.Tag.y))


prop.table(table(data_balanced_over$Performance.Tag.y))


prop.table(table(data_balanced_synthesis$Performance.Tag.y))





##########  Sampling with different methods of balancing for original


### data balancing method = over

data_balanced_over_original <- ovun.sample(as.factor(Performance.Tag.y)  ~ ., data = mergedfile_DT_withoutwoe , method = "over",N = 139734)$data

table(data_balanced_over_original$Performance.Tag.y)


### data balancing method = under

data_balanced_under_original <- ovun.sample(as.factor(Performance.Tag.y)  ~ ., data = mergedfile_DT_withoutwoe , method = "under",N = 5894)$data

table(data_balanced_under_original$Performance.Tag.y)


### data balancing method = both

data_balanced_both_original <- ovun.sample(as.factor(Performance.Tag.y)  ~ ., data = mergedfile_DT_withoutwoe , method = "both", p=0.5, N=69867, seed = 1)$data

table(data_balanced_both_original$Performance.Tag.y)


### data balancing method = synthesis

data_balanced_synthesis_original <- ovun.sample(as.factor(Performance.Tag.y)  ~ ., data = mergedfile_DT_withoutwoe , seed = 1)$data

table(data_balanced_synthesis_original$Performance.Tag.y)


prop.table(table(data_balanced_both_original$Performance.Tag.y))
```

```
prop.table(table(data_balanced_under_original$Performance.Tag.y))

prop.table(table(data_balanced_over_original$Performance.Tag.y))

prop.table(table(data_balanced_synthesis_original$Performance.Tag.y))



################################### smapling technique ends here ##############3

#########################  Decision Tree starts here with Woe Data ##########################


# divide into train and test set
set.seed(123)

split.indices_under <- sample(nrow(data_balanced_under), nrow(data_balanced_under)*0.8, replace = F)

split.indices_over <- sample(nrow(data_balanced_over), nrow(data_balanced_over)*0.8, replace = F)

split.indices_both <- sample(nrow(data_balanced_both), nrow(data_balanced_both)*0.8, replace = F)

split.indices_both <- sample(nrow(data_balanced_synthesis), nrow(data_balanced_synthesis)*0.8, replace = F)

train_under <- data_balanced_under[split.indices_under, ]
test_under <- data_balanced_under[-split.indices_under, ]

train_over <- data_balanced_over[split.indices_over, ]
test_over <- data_balanced_over[-split.indices_over, ]

train_both <- data_balanced_both[split.indices_both, ]
test_both <- data_balanced_both[-split.indices_both, ]
```

```
train_synthesis <- data_balanced_synthesis[split.indices_both, ]

test_synthesis <- data_balanced_synthesis[-split.indices_both, ]


# Classification Trees

library(rpart)

library(rpart.plot)

library(ggplot2)

library(caret)


#1 build tree model- default hyperparameters


train_under$Performance.Tag.y <- as.factor(train_under$Performance.Tag.y)


train_over$Performance.Tag.y <- as.factor(train_over$Performance.Tag.y)


train_both$Performance.Tag.y <- as.factor(train_both$Performance.Tag.y)


train_synthesis$Performance.Tag.y <- as.factor(train_synthesis$Performance.Tag.y)



test_under$Performance.Tag.y <- as.factor(test_under$Performance.Tag.y)


test_over$Performance.Tag.y <- as.factor(test_over$Performance.Tag.y)


test_both$Performance.Tag.y <- as.factor(test_both$Performance.Tag.y)


test_synthesis$Performance.Tag.y <- as.factor(test_synthesis$Performance.Tag.y)





#####################  Decision tree for under smapling starts #######################


tree.model <- rpart(Performance.Tag.y ~ .,                  # formula
```

```
              data = train_under,                  # training data
              method = "class")               # classification or regression

# display decision tree

prp(tree.model)

# make predictions on the test set

tree.predict <- predict(tree.model, test_under, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_under$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction   0   1
# 0 282 142
# 1 317 438
#
# Accuracy : 0.6107
# 95% CI : (0.5822, 0.6386)
# No Information Rate : 0.5081
# P-Value [Acc > NIR] : 8.835e-13
#
# Kappa : 0.2249
# Mcnemar's Test P-Value : 4.600e-16
#
#           Sensitivity : 0.4708
#           Specificity : 0.7552
#         Pos Pred Value : 0.6651
#         Neg Pred Value : 0.5801
#            Prevalence : 0.5081
```

```
#         Detection Rate : 0.2392

#    Detection Prevalence : 0.3596

#      Balanced Accuracy : 0.6130

#

#       'Positive' Class : 0


#2 Change the algorithm to "information gain" instead of default "gini" ----------------------

tree.model <- rpart(Performance.Tag.y ~ .,                    # formula

                 data = train_under,                  # training data

                 method = "class",              # classification or regression

                 parms = list(split = "information"))


# display decision tree


prp(tree.model)


# make predictions on the test set

tree.predict <- predict(tree.model, test_under, type = "class")


# evaluate the results

confusionMatrix(tree.predict, test_under$Performance.Tag.y)


# Confusion Matrix and Statistics

#

# Reference

# Prediction   0   1

# 0 282 142

# 1 317 438

#

# Accuracy : 0.6107

# 95% CI : (0.5822, 0.6386)

# No Information Rate : 0.5081

# P-Value [Acc > NIR] : 8.835e-13

#

# Kappa : 0.2249
```

```
# Mcnemar's Test P-Value : 4.600e-16
#
#            Sensitivity : 0.4708
#            Specificity : 0.7552
#         Pos Pred Value : 0.6651
#         Neg Pred Value : 0.5801
#             Prevalence : 0.5081
#         Detection Rate : 0.2392
#   Detection Prevalence : 0.3596
#      Balanced Accuracy : 0.6130
#
#       'Positive' Class : 0

#3 Tune the hyperparameters ----------------------------------------------------------
tree.model <- rpart(Performance.Tag.y ~ .,                              # formula
                 data = train_under,                          # training data
                 method = "class",                        # classification or regression
                 control = rpart.control(minsplit = 10,  # min observations for node
                                         minbucket = 10, # min observations for leaf node
                                         cp = 0.005))       # complexity parameter

# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_under, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_under$Performance.Tag.y)

# Confusion Matrix and Statistics
#
# Reference
# Prediction   0   1
# 0 303 150
```

```
# 1 296 430

#

# Accuracy : 0.6217

# 95% CI : (0.5933, 0.6495)

# No Information Rate : 0.5081

# P-Value [Acc > NIR] : 2.644e-15

#

# Kappa : 0.2462

# Mcnemar's Test P-Value : 6.605e-12

#

# Sensitivity : 0.5058

# Specificity : 0.7414

# Pos Pred Value : 0.6689

# Neg Pred Value : 0.5923

# Prevalence : 0.5081

# Detection Rate : 0.2570

# Detection Prevalence : 0.3842

# Balanced Accuracy : 0.6236

#

# 'Positive' Class : 0




#4 A more complex tree ----------------------------------------------------------------

tree.model <- rpart(Performance.Tag.y ~ .,                    # formula

            data = train_under,                   # training data

            method = "class",                # classification or regression

            control = rpart.control(minsplit = 10,    # min observations for node

                        minbucket = 10,   # min observations for leaf node

                        cp = 0.0025))     # complexity parameter


# display decision tree

prp(tree.model)


# make predictions on the test set
```

```
tree.predict <- predict(tree.model, test_under, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_under$Performance.Tag.y)

# Confusion Matrix and Statistics
#
# Reference
# Prediction   0   1
# 0 327 165
# 1 272 415
#
# Accuracy : 0.6293
# 95% CI : (0.6011, 0.657)
# No Information Rate : 0.5081
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2607
# Mcnemar's Test P-Value : 3.964e-07
#
#            Sensitivity : 0.5459
#            Specificity : 0.7155
#         Pos Pred Value : 0.6646
#         Neg Pred Value : 0.6041
#             Prevalence : 0.5081
#         Detection Rate : 0.2774
#   Detection Prevalence : 0.4173
#      Balanced Accuracy : 0.6307
#
#       'Positive' Class : 0
```

```
# Cross test to choose CP ------------------------------------------------------------

library(caret)


# set the number of folds in cross test to 5

tree.control = trainControl(method = "cv", number = 25)


# set the search space for CP

tree.grid = expand.grid(cp = seq(0.001, 0.005, 0.00016))


# train model

tree.model <- train(Performance.Tag.y ~ .,

                    data = train_under,

                    method = "rpart",

                    metric = "Accuracy",

                    trControl = tree.control,

                    tuneGrid = tree.grid,

                    control = rpart.control(minsplit = 10,

                                            minbucket = 10))


# look at cross validated model results

tree.model


# look at best value of hyperparameter

tree.model$bestTune


# make predictions on test set

tree.predict <- predict.train(tree.model, test_under)


# accuracy

confusionMatrix(tree.predict, test_under$Performance.Tag.y)



# Confusion Matrix and Statistics

#

# Reference
```

```
# Prediction   0   1
# 0 292 139
# 1 307 441
#
# Accuracy : 0.6217
# 95% CI : (0.5933, 0.6495)
# No Information Rate : 0.5081
# P-Value [Acc > NIR] : 2.644e-15
#
# Kappa : 0.2467
# Mcnemar's Test P-Value : 2.622e-15
#
# Sensitivity : 0.4875
# Specificity : 0.7603
# Pos Pred Value : 0.6775
# Neg Pred Value : 0.5896
# Prevalence : 0.5081
# Detection Rate : 0.2477
# Detection Prevalence : 0.3656
# Balanced Accuracy : 0.6239
#
# 'Positive' Class : 0

# plot CP vs Accuracy
library(ggplot2)
accuracy_graph <- data.frame(tree.model$results)
ggplot(data = accuracy_graph, aes(x = cp, y = Accuracy*100)) +
  geom_line() +
  geom_point() +
  labs(x = "Complexity Parameter (CP)", y = "Accuracy", title = "CP vs Accuracy")


####################  Decision tree for under smapling ends ######################
```

```
####################  Decision tree for over smapling starts #######################

tree.model <- rpart(Performance.Tag.y ~ .,                    # formula
                data = train_over,                  # training data
                method = "class")              # classification or regression

# display decision tree

prp(tree.model)

# make predictions on the test set

tree.predict <- predict(tree.model, test_over, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_over$Performance.Tag.y)

# Confusion Matrix and Statistics
#
# Reference
# Prediction     0     1
# 0  7440  4195
# 1  5801 10511
#
# Accuracy : 0.6423
# 95% CI : (0.6367, 0.6479)
# No Information Rate : 0.5262
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2783
# Mcnemar's Test P-Value : < 2.2e-16
```

```
#
# Sensitivity : 0.5619
# Specificity : 0.7147
# Pos Pred Value : 0.6394
# Neg Pred Value : 0.6444
# Prevalence : 0.4738
# Detection Rate : 0.2662
# Detection Prevalence : 0.4163
# Balanced Accuracy : 0.6383
#
# 'Positive' Class : 0



# Change the algorithm to "information gain" instead of default "gini" ----------------------
tree.model <- rpart(Performance.Tag.y ~ .,                    # formula
            data = train_over,                  # training data
            method = "class",               # classification or regression
            parms = list(split = "information"))

# display decision tree

prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_over, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_over$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
```

```
# 0  7440  4195

# 1  5801 10511

#

# Accuracy : 0.6423

# 95% CI : (0.6367, 0.6479)

# No Information Rate : 0.5262

# P-Value [Acc > NIR] : < 2.2e-16

#

# Kappa : 0.2783

# Mcnemar's Test P-Value : < 2.2e-16

#

# Sensitivity : 0.5619

# Specificity : 0.7147

# Pos Pred Value : 0.6394

# Neg Pred Value : 0.6444

# Prevalence : 0.4738

# Detection Rate : 0.2662

# Detection Prevalence : 0.4163

# Balanced Accuracy : 0.6383

#

# 'Positive' Class : 0




#3 Tune the hyperparameters ----------------------------------------------------------

tree.model <- rpart(Performance.Tag.y ~ .,                          # formula

             data = train_over,                        # training data

             method = "class",                     # classification or regression

             control = rpart.control(minsplit = 50,  # min observations for node

                          minbucket = 50, # min observations for leaf node

                          cp = 0.005))      # complexity parameter


# display decision tree

prp(tree.model)
```

```
# make predictions on the test set
tree.predict <- predict(tree.model, test_over, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_over$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction     0     1
# 0  7440  4195
# 1  5801 10511
#
# Accuracy : 0.6423
# 95% CI : (0.6367, 0.6479)
# No Information Rate : 0.5262
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2783
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.5619
#            Specificity : 0.7147
#         Pos Pred Value : 0.6394
#         Neg Pred Value : 0.6444
#             Prevalence : 0.4738
#         Detection Rate : 0.2662
#   Detection Prevalence : 0.4163
#      Balanced Accuracy : 0.6383
#
#       'Positive' Class : 0

#4 A more complex tree ------------------------------------------------------------------
tree.model <- rpart(Performance.Tag.y ~ .,                              # formula
```

```
            data = train_over,                  # training data
            method = "class",                   # classification or regression
            control = rpart.control(minsplit = 10,    # min observations for node
                        minbucket = 10,    # min observations for leaf node
                        cp = 0.0025))      # complexity parameter

# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_over, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_over$Performance.Tag.y)

# Confusion Matrix and Statistics
#
# Reference
# Prediction     0     1
# 0  7440  4195
# 1  5801 10511
#
# Accuracy : 0.6423
# 95% CI : (0.6367, 0.6479)
# No Information Rate : 0.5262
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2783
# Mcnemar's Test P-Value : < 2.2e-16
#
#          Sensitivity : 0.5619
#          Specificity : 0.7147
#       Pos Pred Value : 0.6394
#       Neg Pred Value : 0.6444
#           Prevalence : 0.4738
```

```
#         Detection Rate : 0.2662
#   Detection Prevalence : 0.4163
#      Balanced Accuracy : 0.6383
#
#       'Positive' Class : 0



# Cross test to choose CP ------------------------------------------------------------

# set the number of folds in cross test to 5
tree.control = trainControl(method = "cv", number = 25)

# set the search space for CP
tree.grid = expand.grid(cp = seq(0.001, 0.0035, 0.0001))

# train model
tree.model <- train(Performance.Tag.y ~ .,
                    data = train_over,
                    method = "rpart",
                    metric = "Accuracy",
                    trControl = tree.control,
                    tuneGrid = tree.grid,
                    control = rpart.control(minsplit = 10,
                                            minbucket = 10))

# look at cross validated model results
tree.model

# look at best value of hyperparameter
tree.model$bestTune

# make predictions on test set
tree.predict <- predict.train(tree.model, test_over)
```

```
# accuracy
confusionMatrix(tree.predict, test_over$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction     0     1
# 0  6524  2898
# 1  6717 11808
#
# Accuracy : 0.656
# 95% CI : (0.6504, 0.6615)
# No Information Rate : 0.5262
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.3
# Mcnemar's Test P-Value : < 2.2e-16
#
# Sensitivity : 0.4927
# Specificity : 0.8029
# Pos Pred Value : 0.6924
# Neg Pred Value : 0.6374
# Prevalence : 0.4738
# Detection Rate : 0.2334
# Detection Prevalence : 0.3371
# Balanced Accuracy : 0.6478
#
# 'Positive' Class : 0


# plot CP vs Accuracy
library(ggplot2)
accuracy_graph <- data.frame(tree.model$results)
ggplot(data = accuracy_graph, aes(x = cp, y = Accuracy*100)) +
  geom_line() +
```

```
  geom_point() +
  labs(x = "Complexity Parameter (CP)", y = "Accuracy", title = "CP vs Accuracy")


#####################  Decision tree for over smapling ends #######################


#####################  Decision tree for both smapling starts #######################


tree.model <- rpart(Performance.Tag.y ~ .,                   # formula
             data = train_both,                  # training data
             method = "class")              # classification or regression

# display decision tree

prp(tree.model)

# make predictions on the test set

tree.predict <- predict(tree.model, test_both, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_both$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 3454 1772
# 1 3535 5213
#
# Accuracy : 0.6202
```

```
# 95% CI : (0.6121, 0.6283)
# No Information Rate : 0.5001
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2405
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.4942
#            Specificity : 0.7463
#         Pos Pred Value : 0.6609
#         Neg Pred Value : 0.5959
#             Prevalence : 0.5001
#         Detection Rate : 0.2472
#   Detection Prevalence : 0.3740
#      Balanced Accuracy : 0.6203
#
#       'Positive' Class : 0


#2 Change the algorithm to "information gain" instead of default "gini" ----------------------
tree.model <- rpart(Performance.Tag.y ~ .,                  # formula
                    data = train_both,                 # training data
                    method = "class",              # classification or regression
                    parms = list(split = "information"))

# display decision tree

prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_both, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_both$Performance.Tag.y)
```

```
# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 3454 1772
# 1 3535 5213
#
# Accuracy : 0.6202
# 95% CI : (0.6121, 0.6283)
# No Information Rate : 0.5001
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2405
# Mcnemar's Test P-Value : < 2.2e-16
#
# Sensitivity : 0.4942
# Specificity : 0.7463
# Pos Pred Value : 0.6609
# Neg Pred Value : 0.5959
# Prevalence : 0.5001
# Detection Rate : 0.2472
# Detection Prevalence : 0.3740
# Balanced Accuracy : 0.6203
#
# 'Positive' Class : 0


#3 Tune the hyperparameters ---------------------------------------------------------
tree.model <- rpart(Performance.Tag.y ~ .,                             # formula
            data = train_both,                          # training data
            method = "class",                      # classification or regression
            control = rpart.control(minsplit = 50,  # min observations for node
                        minbucket = 50, # min observations for leaf node
                        cp = 0.005))       # complexity parameter
```

```
# display decision tree

prp(tree.model)


# make predictions on the test set

tree.predict <- predict(tree.model, test_both, type = "class")


# evaluate the results

confusionMatrix(tree.predict, test_both$Performance.Tag.y)



# Confusion Matrix and Statistics

#

# Reference

# Prediction    0    1

# 0 3902 1978

# 1 3087 5007

#

# Accuracy : 0.6375

# 95% CI : (0.6295, 0.6455)

# No Information Rate : 0.5001

# P-Value [Acc > NIR] : < 2.2e-16

#

# Kappa : 0.2751

# Mcnemar's Test P-Value : < 2.2e-16

#

# Sensitivity : 0.5583

# Specificity : 0.7168

# Pos Pred Value : 0.6636

# Neg Pred Value : 0.6186

# Prevalence : 0.5001

# Detection Rate : 0.2792

# Detection Prevalence : 0.4208

# Balanced Accuracy : 0.6376

#
```

```
# 'Positive' Class : 0



#4 A more complex tree ------------------------------------------------------------------
tree.model <- rpart(Performance.Tag.y ~ .,                              # formula
                data = train_both,                          # training data
                method = "class",                       # classification or regression
                control = rpart.control(minsplit = 10,     # min observations for node
                                minbucket = 10,    # min observations for leaf node
                                cp = 0.0025))      # complexity parameter

# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_both, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_both$Performance.Tag.y)



# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 3902 1978
# 1 3087 5007
#
# Accuracy : 0.6375
# 95% CI : (0.6295, 0.6455)
# No Information Rate : 0.5001
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2751
```

```
# Mcnemar's Test P-Value : < 2.2e-16

#

# Sensitivity : 0.5583

# Specificity : 0.7168

# Pos Pred Value : 0.6636

# Neg Pred Value : 0.6186

# Prevalence : 0.5001

# Detection Rate : 0.2792

# Detection Prevalence : 0.4208

# Balanced Accuracy : 0.6376

#

# 'Positive' Class : 0




#5 Cross test to choose CP ------------------------------------------------------------


# set the number of folds in cross test to 5

tree.control = trainControl(method = "cv", number = 25)


# set the search space for CP

tree.grid = expand.grid(cp = seq(0.001, 0.0035, 0.0001))


# train model

tree.model <- train(Performance.Tag.y ~ .,

            data = train_both,

            method = "rpart",

            metric = "Accuracy",

            trControl = tree.control,

            tuneGrid = tree.grid,

            control = rpart.control(minsplit = 5,

                        minbucket = 5))


# look at cross validated model results

tree.model
```

```
# look at best value of hyperparameter

tree.model$bestTune


# make predictions on test set

tree.predict <- predict.train(tree.model, test_both)


# accuracy

confusionMatrix(tree.predict, test_both$Performance.Tag.y)



# Confusion Matrix and Statistics

#

# Reference

# Prediction    0    1

# 0 3890 1853

# 1 3099 5132

#

# Accuracy : 0.6456

# 95% CI : (0.6376, 0.6536)

# No Information Rate : 0.5001

# P-Value [Acc > NIR] : < 2.2e-16

#

# Kappa : 0.2913

# Mcnemar's Test P-Value : < 2.2e-16

#

#            Sensitivity : 0.5566

#            Specificity : 0.7347

#         Pos Pred Value : 0.6773

#         Neg Pred Value : 0.6235

#             Prevalence : 0.5001

#         Detection Rate : 0.2784

#   Detection Prevalence : 0.4110

#      Balanced Accuracy : 0.6457

#
```

```
#       'Positive' Class : 0


# plot CP vs Accuracy

library(ggplot2)

accuracy_graph <- data.frame(tree.model$results)

ggplot(data = accuracy_graph, aes(x = cp, y = Accuracy*100)) +

  geom_line() +

  geom_point() +

  labs(x = "Complexity Parameter (CP)", y = "Accuracy", title = "CP vs Accuracy")



#####################  Decision tree for both smapling ends #######################



#####################  Decision tree for Synthesys smapling starts #######################



tree.model <- rpart(Performance.Tag.y ~ .,                    # formula

                  data = train_synthesis,                  # training data

                  method = "class")             # classification or regression


# display decision tree


prp(tree.model)


# make predictions on the test set


tree.predict <- predict(tree.model, test_synthesis, type = "class")


# evaluate the results

confusionMatrix(tree.predict, test_synthesis$Performance.Tag.y)



# Confusion Matrix and Statistics

#
```

```
# Reference
# Prediction    0    1
# 0 3454 1772
# 1 3535 5213
#
# Accuracy : 0.6202
# 95% CI : (0.6121, 0.6283)
# No Information Rate : 0.5001
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2405
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.4942
#            Specificity : 0.7463
#         Pos Pred Value : 0.6609
#         Neg Pred Value : 0.5959
#             Prevalence : 0.5001
#         Detection Rate : 0.2472
#   Detection Prevalence : 0.3740
#      Balanced Accuracy : 0.6203
#
#       'Positive' Class : 0

#2 Change the algorithm to "information gain" instead of default "gini" ----------------------
tree.model <- rpart(Performance.Tag.y ~ .,                    # formula
                 data = train_synthesis,                  # training data
                 method = "class",              # classification or regression
                 parms = list(split = "information"))

# display decision tree

prp(tree.model)

# make predictions on the test set
```

```
tree.predict <- predict(tree.model, test_synthesis, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_synthesis$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 3454 1772
# 1 3535 5213
#
# Accuracy : 0.6202
# 95% CI : (0.6121, 0.6283)
# No Information Rate : 0.5001
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2405
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.4942
#            Specificity : 0.7463
#         Pos Pred Value : 0.6609
#         Neg Pred Value : 0.5959
#             Prevalence : 0.5001
#         Detection Rate : 0.2472
#   Detection Prevalence : 0.3740
#      Balanced Accuracy : 0.6203
#
#       'Positive' Class : 0


#3 Tune the hyperparameters ---------------------------------------------------------
tree.model <- rpart(Performance.Tag.y ~ .,                          # formula
```

```
                data = train_synthesis,                          # training data
                method = "class",                      # classification or regression
                control = rpart.control(minsplit = 50,  # min observations for node
                                    minbucket = 50, # min observations for leaf node
                                    cp = 0.005))      # complexity parameter

# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_synthesis, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_synthesis$Performance.Tag.y)



# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 3902 1978
# 1 3087 5007
#
# Accuracy : 0.6375
# 95% CI : (0.6295, 0.6455)
# No Information Rate : 0.5001
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2751
# Mcnemar's Test P-Value : < 2.2e-16
#
# Sensitivity : 0.5583
# Specificity : 0.7168
# Pos Pred Value : 0.6636
# Neg Pred Value : 0.6186
```

```
# Prevalence : 0.5001

# Detection Rate : 0.2792

# Detection Prevalence : 0.4208

# Balanced Accuracy : 0.6376

#

# 'Positive' Class : 0



#4 A more complex tree ----------------------------------------------------------------

tree.model <- rpart(Performance.Tag.y ~ .,                             # formula

                data = train_synthesis,                         # training data

                method = "class",                       # classification or regression

                control = rpart.control(minsplit = 10,     # min observations for node

                                minbucket = 10,    # min observations for leaf node

                                cp = 0.001))      # complexity parameter


# display decision tree

prp(tree.model)


# make predictions on the test set

tree.predict <- predict(tree.model, test_synthesis, type = "class")


# evaluate the results

confusionMatrix(tree.predict, test_synthesis$Performance.Tag.y)



# Confusion Matrix and Statistics

#

# Reference

# Prediction    0    1

# 0 3890 1853

# 1 3099 5132

#

# Accuracy : 0.6456

# 95% CI : (0.6376, 0.6536)
```

```
# No Information Rate : 0.5001

# P-Value [Acc > NIR] : < 2.2e-16

#

# Kappa : 0.2913

# Mcnemar's Test P-Value : < 2.2e-16

#

# Sensitivity : 0.5566

# Specificity : 0.7347

# Pos Pred Value : 0.6773

# Neg Pred Value : 0.6235

# Prevalence : 0.5001

# Detection Rate : 0.2784

# Detection Prevalence : 0.4110

# Balanced Accuracy : 0.6457

#

# 'Positive' Class : 0




#5 Cross test to choose CP ------------------------------------------------------------


# set the number of folds in cross test to 5

tree.control = trainControl(method = "cv", number = 25)


# set the search space for CP

tree.grid = expand.grid(cp = seq(0.001, 0.0035, 0.0001))


# train model

tree.model <- train(Performance.Tag.y ~ .,

                data = train_synthesis,

                method = "rpart",

                metric = "Accuracy",

                trControl = tree.control,

                tuneGrid = tree.grid,
```

```
                    control = rpart.control(minsplit = 5,
                                            minbucket = 5))

# look at cross validated model results
tree.model

# look at best value of hyperparameter
tree.model$bestTune

# make predictions on test set
tree.predict <- predict.train(tree.model, test_synthesis)

# accuracy
confusionMatrix(tree.predict, test_synthesis$Performance.Tag.y)



# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 3890 1853
# 1 3099 5132
#
# Accuracy : 0.6456
# 95% CI : (0.6376, 0.6536)
# No Information Rate : 0.5001
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2913
# Mcnemar's Test P-Value : < 2.2e-16
#
# Sensitivity : 0.5566
# Specificity : 0.7347
# Pos Pred Value : 0.6773
# Neg Pred Value : 0.6235
```

```
# Prevalence : 0.5001

# Detection Rate : 0.2784

# Detection Prevalence : 0.4110

# Balanced Accuracy : 0.6457

#

# 'Positive' Class : 0



# plot CP vs Accuracy

library(ggplot2)

accuracy_graph <- data.frame(tree.model$results)

ggplot(data = accuracy_graph, aes(x = cp, y = Accuracy*100)) +

  geom_line() +

  geom_point() +

  labs(x = "Complexity Parameter (CP)", y = "Accuracy", title = "CP vs Accuracy")



####################  Decision tree for Synthesys smapling ends #######################



#######################################################################################
#########################################################################################
#######################################################################################

########################  Original Data Decision Tree ##################################



#######################################################################################



########################  Decision Tree starts here with Woe Data #########################
```

```
# divide into train and test set

set.seed(123)

split.indices_under_original <- sample(nrow(data_balanced_under_original), nrow(data_balanced_under_original)*0.8, replace = F)

split.indices_over_original <- sample(nrow(data_balanced_over_original), nrow(data_balanced_over_original)*0.8, replace = F)

split.indices_both_original <- sample(nrow(data_balanced_both_original), nrow(data_balanced_both_original)*0.8, replace = F)

split.indices_sysnthesis_original <- sample(nrow(data_balanced_synthesis_original), nrow(data_balanced_synthesis_original)*0.8, replace = F)


train_under_original <- data_balanced_under_original[split.indices_under_original, ]

test_under_original <- data_balanced_under_original[-split.indices_under_original, ]

train_over_original <- data_balanced_over_original[split.indices_over_original, ]

test_over_original <- data_balanced_over_original[-split.indices_over_original, ]

train_both_original <- data_balanced_both_original[split.indices_both_original, ]

test_both_original <- data_balanced_both_original[-split.indices_both_original, ]

train_synthesis_original <- data_balanced_synthesis[split.indices_sysnthesis_original, ]

test_synthesis_original <- data_balanced_synthesis[-split.indices_sysnthesis_original, ]


# Classification Trees

library(rpart)

library(rpart.plot)

library(ggplot2)

library(caret)


#1 build tree model- default hyperparameters


train_under_original$Performance.Tag.y <- as.factor(train_under_original$Performance.Tag.y)

train_over_original$Performance.Tag.y <- as.factor(train_over_original$Performance.Tag.y)

train_both_original$Performance.Tag.y <- as.factor(train_both_original$Performance.Tag.y)

train_synthesis_original$Performance.Tag.y <- as.factor(train_synthesis_original$Performance.Tag.y)

test_under_original$Performance.Tag.y <- as.factor(test_under_original$Performance.Tag.y)

test_over_original$Performance.Tag.y <- as.factor(test_over_original$Performance.Tag.y)

test_both_original$Performance.Tag.y <- as.factor(test_both_original$Performance.Tag.y)

test_synthesis_original$Performance.Tag.y <- as.factor(test_synthesis_original$Performance.Tag.y)
```

```
####################  Decision tree for under smapling starts ######################

tree.model <- rpart(Performance.Tag.y ~ .,                    # formula

                data = train_under_original,                  # training data

                method = "class")              # classification or regression

# display decision tree

prp(tree.model)

# make predictions on the test set

tree.predict <- predict(tree.model, test_under_original, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_under_original$Performance.Tag.y)

# Confusion Matrix and Statistics
#
# Reference
# Prediction   0   1
# 0 345 143
# 1 270 421
#
# Accuracy : 0.6497
# 95% CI : (0.6217, 0.677)
# No Information Rate : 0.5216
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.3046
# Mcnemar's Test P-Value : 5.644e-10
#
# Sensitivity : 0.5610
# Specificity : 0.7465
# Pos Pred Value : 0.7070
```

```
# Neg Pred Value : 0.6093

# Prevalence : 0.5216

# Detection Rate : 0.2926

# Detection Prevalence : 0.4139

# Balanced Accuracy : 0.6537

#

# 'Positive' Class : 0


# Change the algorithm to "information gain" instead of default "gini" ----------------------

tree.model <- rpart(Performance.Tag.y ~ .,                  # formula

                data = train_under_original,                # training data

                method = "class",             # classification or regression

                parms = list(split = "information"))


# display decision tree


prp(tree.model)


# make predictions on the test set

tree.predict <- predict(tree.model, test_under_original, type = "class")


# evaluate the results

confusionMatrix(tree.predict, test_under_original$Performance.Tag.y)


########## No Change in results




# Tune the hyperparameters ---------------------------------------------------------

tree.model <- rpart(Performance.Tag.y ~ .,                          # formula

                data = train_under_original,                        # training data

                method = "class",                       # classification or regression

                control = rpart.control(minsplit = 10,  # min observations for node

                               minbucket = 10, # min observations for leaf node

                               cp = 0.003))      # complexity parameter
```

```
# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_under_original, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_under_original$Performance.Tag.y)

# Confusion Matrix and Statistics
#
# Reference
# Prediction   0   1
# 0 322 151
# 1 293 413
#
# Accuracy : 0.6234
# 95% CI : (0.595, 0.6512)
# No Information Rate : 0.5216
# P-Value [Acc > NIR] : 1.180e-12
#
# Kappa : 0.2532
# Mcnemar's Test P-Value : 2.208e-11
#
# Sensitivity : 0.5236
# Specificity : 0.7323
# Pos Pred Value : 0.6808
# Neg Pred Value : 0.5850
# Prevalence : 0.5216
# Detection Rate : 0.2731
# Detection Prevalence : 0.4012
# Balanced Accuracy : 0.6279
#
# 'Positive' Class : 0
```

```
#4 A more complex tree -----------------------------------------------------------------

tree.model <- rpart(Performance.Tag.y ~ .,                    # formula
            data = train_under_original,                # training data
            method = "class",                # classification or regression
            control = rpart.control(minsplit = 10,    # min observations for node
                        minbucket = 10,   # min observations for leaf node
                        cp = 0.002))     # complexity parameter

# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_under_original, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_under_original$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction   0   1
# 0 343 162
# 1 272 402
#
# Accuracy : 0.6319
# 95% CI : (0.6036, 0.6595)
# No Information Rate : 0.5216
# P-Value [Acc > NIR] : 1.427e-14
#
# Kappa : 0.2683
# Mcnemar's Test P-Value : 1.675e-07
#
# Sensitivity : 0.5577
# Specificity : 0.7128
```

```
# Pos Pred Value : 0.6792

# Neg Pred Value : 0.5964

# Prevalence : 0.5216

# Detection Rate : 0.2909

# Detection Prevalence : 0.4283

# Balanced Accuracy : 0.6352

#

# 'Positive' Class : 0




# Cross test to choose CP ------------------------------------------------------------

library(caret)


# set the number of folds in cross test to 5

tree.control = trainControl(method = "cv", number = 10)


# set the search space for CP

tree.grid = expand.grid(cp = seq(0.002, 0.003, 0.0001))


# train model

tree.model <- train(Performance.Tag.y ~ .,

            data = train_under_original,

            method = "rpart",

            metric = "Accuracy",

            trControl = tree.control,

            tuneGrid = tree.grid,

            control = rpart.control(minsplit = 10,

                        minbucket = 10))


# look at cross validated model results

tree.model


# look at best value of hyperparameter

tree.model$bestTune
```

```
# make predictions on test set

tree.predict <- predict.train(tree.model, test_under_original)


# accuracy

confusionMatrix(tree.predict, test_under_original$Performance.Tag.y)



# Confusion Matrix and Statistics
#
# Reference
# Prediction   0   1
# 0 337 159
# 1 278 405
#
# Accuracy : 0.6293
# 95% CI : (0.6011, 0.657)
# No Information Rate : 0.5216
# P-Value [Acc > NIR] : 5.570e-14
#
# Kappa : 0.2637
# Mcnemar's Test P-Value : 1.655e-08
#
# Sensitivity : 0.5480
# Specificity : 0.7181
# Pos Pred Value : 0.6794
# Neg Pred Value : 0.5930
# Prevalence : 0.5216
# Detection Rate : 0.2858
# Detection Prevalence : 0.4207
# Balanced Accuracy : 0.6330
#
# 'Positive' Class : 0
```

```
# plot CP vs Accuracy
library(ggplot2)
accuracy_graph <- data.frame(tree.model$results)
ggplot(data = accuracy_graph, aes(x = cp, y = Accuracy*100)) +
  geom_line() +
  geom_point() +
  labs(x = "Complexity Parameter (CP)", y = "Accuracy", title = "CP vs Accuracy")


#####################  Decision tree for under smapling ends #######################



#####################  Decision tree for over smapling starts #######################



tree.model <- rpart(Performance.Tag.y ~ .,                    # formula
                data = train_over_original,                  # training data
                method = "class")              # classification or regression

# display decision tree

prp(tree.model)

# make predictions on the test set

tree.predict <- predict(tree.model, test_over_original, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_over_original$Performance.Tag.y)


# Confusion Matrix and Statistics
#
```

```
# Reference
# Prediction     0     1
# 0  7103  4179
# 1  5960 10705
#
# Accuracy : 0.6372
# 95% CI : (0.6315, 0.6428)
# No Information Rate : 0.5326
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2652
# Mcnemar's Test P-Value : < 2.2e-16
#
#           Sensitivity : 0.5437
#           Specificity : 0.7192
#        Pos Pred Value : 0.6296
#        Neg Pred Value : 0.6424
#            Prevalence : 0.4674
#        Detection Rate : 0.2542
#   Detection Prevalence : 0.4037
#      Balanced Accuracy : 0.6315
#
#       'Positive' Class : 0


# Change the algorithm to "information gain" instead of default "gini" ----------------------
tree.model <- rpart(Performance.Tag.y ~ .,                    # formula
                 data = train_over_original,                  # training data
                 method = "class",              # classification or regression
                 parms = list(split = "information"))

# display decision tree

prp(tree.model)
```

```
# make predictions on the test set
tree.predict <- predict(tree.model, test_over_original, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_over_original$Performance.Tag.y)


### No Change in results


# Tune the hyperparameters ----------------------------------------------------------
tree.model <- rpart(Performance.Tag.y ~ .,                              # formula
                data = train_over_original,                          # training data
                method = "class",                       # classification or regression
                control = rpart.control(minsplit = 10,  # min observations for node
                                    minbucket = 10, # min observations for leaf node
                                    cp = 0.002))      # complexity parameter

# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_over_original, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_over_original$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction     0     1
# 0  6638  3451
# 1  6425 11433
#
```

```
# Accuracy : 0.6466

# 95% CI : (0.641, 0.6522)

# No Information Rate : 0.5326

# P-Value [Acc > NIR] : < 2.2e-16

#

# Kappa : 0.2802

# Mcnemar's Test P-Value : < 2.2e-16

#

# Sensitivity : 0.5082

# Specificity : 0.7681

# Pos Pred Value : 0.6579

# Neg Pred Value : 0.6402

# Prevalence : 0.4674

# Detection Rate : 0.2375

# Detection Prevalence : 0.3610

# Balanced Accuracy : 0.6381

#

# 'Positive' Class : 0


#4 A more complex tree -----------------------------------------------------------------

tree.model <- rpart(Performance.Tag.y ~ .,                    # formula

        data = train_over_original,                   # training data

        method = "class",                 # classification or regression

        control = rpart.control(minsplit = 10,    # min observations for node

                  minbucket = 10,   # min observations for leaf node

                  cp = 0.001))     # complexity parameter


# display decision tree

prp(tree.model)


# make predictions on the test set

tree.predict <- predict(tree.model, test_over_original, type = "class")


# evaluate the results

confusionMatrix(tree.predict, test_over_original$Performance.Tag.y)
```

```
# Confusion Matrix and Statistics

#

# Reference

# Prediction     0     1

# 0  6706  3322

# 1  6357 11562

#

# Accuracy : 0.6537

# 95% CI : (0.6481, 0.6592)

# No Information Rate : 0.5326

# P-Value [Acc > NIR] : < 2.2e-16

#

# Kappa : 0.2943

# Mcnemar's Test P-Value : < 2.2e-16

#

# Sensitivity : 0.5134

# Specificity : 0.7768

# Pos Pred Value : 0.6687

# Neg Pred Value : 0.6452

# Prevalence : 0.4674

# Detection Rate : 0.2400

# Detection Prevalence : 0.3588

# Balanced Accuracy : 0.6451

#

# 'Positive' Class : 0



# Cross test to choose CP ------------------------------------------------------------


# set the number of folds in cross test to 5

tree.control = trainControl(method = "cv", number = 10)


# set the search space for CP

tree.grid = expand.grid(cp = seq(0.001, 0.002, 0.0001))
```

```
# train model

tree.model <- train(Performance.Tag.y ~ .,

                    data = train_over_original,

                    method = "rpart",

                    metric = "Accuracy",

                    trControl = tree.control,

                    tuneGrid = tree.grid,

                    control = rpart.control(minsplit = 10,

                                            minbucket = 10))


# look at cross validated model results

tree.model


# look at best value of hyperparameter

tree.model$bestTune


# make predictions on test set

tree.predict <- predict.train(tree.model, test_over_original)


# accuracy

confusionMatrix(tree.predict, test_over_original$Performance.Tag.y)



# Confusion Matrix and Statistics

#

# Reference

# Prediction     0     1

# 0  6706  3322

# 1  6357 11562

#

# Accuracy : 0.6537

# 95% CI : (0.6481, 0.6592)

# No Information Rate : 0.5326

# P-Value [Acc > NIR] : < 2.2e-16
```

```
#
# Kappa : 0.2943
# Mcnemar's Test P-Value : < 2.2e-16
#
# Sensitivity : 0.5134
# Specificity : 0.7768
# Pos Pred Value : 0.6687
# Neg Pred Value : 0.6452
# Prevalence : 0.4674
# Detection Rate : 0.2400
# Detection Prevalence : 0.3588
# Balanced Accuracy : 0.6451
#
# 'Positive' Class : 0

# plot CP vs Accuracy
library(ggplot2)
accuracy_graph <- data.frame(tree.model$results)
ggplot(data = accuracy_graph, aes(x = cp, y = Accuracy*100)) +
  geom_line() +
  geom_point() +
  labs(x = "Complexity Parameter (CP)", y = "Accuracy", title = "CP vs Accuracy")


####################  Decision tree for over smapling ends ######################


####################  Decision tree for both smapling starts ######################



tree.model <- rpart(Performance.Tag.y ~ .,                    # formula
             data = train_both_original,                 # training data
             method = "class")             # classification or regression
```

```
# display decision tree

prp(tree.model)

# make predictions on the test set

tree.predict <- predict(tree.model, test_both_original, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_both_original$Performance.Tag.y)



# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 3805 1908
# 1 3206 5055
#
# Accuracy : 0.634
# 95% CI : (0.626, 0.642)
# No Information Rate : 0.5017
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2685
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.5427
#            Specificity : 0.7260
#         Pos Pred Value : 0.6660
#         Neg Pred Value : 0.6119
#             Prevalence : 0.5017
#         Detection Rate : 0.2723
#    Detection Prevalence : 0.4088
#       Balanced Accuracy : 0.6343
```

```
#
#       'Positive' Class : 0



# Change the algorithm to "information gain" instead of default "gini" ----------------------
tree.model <- rpart(Performance.Tag.y ~ .,                    # formula
                data = train_both_original,                  # training data
                method = "class",              # classification or regression
                parms = list(split = "information"))

# display decision tree

prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_both_original, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_both_original$Performance.Tag.y)

# No Change in results

#3 Tune the hyperparameters ---------------------------------------------------------
tree.model <- rpart(Performance.Tag.y ~ .,                            # formula
                data = train_both_original,                          # training data
                method = "class",                        # classification or regression
                control = rpart.control(minsplit = 10,  # min observations for node
                               minbucket = 10, # min observations for leaf node
                               cp = 0.003))       # complexity parameter

# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_both_original, type = "class")
```

```
# evaluate the results

confusionMatrix(tree.predict, test_both_original$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 3732 1765
# 1 3279 5198
#
# Accuracy : 0.639
# 95% CI : (0.631, 0.647)
# No Information Rate : 0.5017
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.2786
# Mcnemar's Test P-Value : < 2.2e-16
#
#           Sensitivity : 0.5323
#           Specificity : 0.7465
#         Pos Pred Value : 0.6789
#         Neg Pred Value : 0.6132
#            Prevalence : 0.5017
#         Detection Rate : 0.2671
#   Detection Prevalence : 0.3934
#      Balanced Accuracy : 0.6394
#
#       'Positive' Class : 0

# A more complex tree -----------------------------------------------------------------
tree.model <- rpart(Performance.Tag.y ~ .,                          # formula
             data = train_both_original,                      # training data
             method = "class",                    # classification or regression
```

```r
                    control = rpart.control(minsplit = 10,     # min observations for node
                                            minbucket = 10,    # min observations for leaf node
                                            cp = 0.001))       # complexity parameter

# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_both_original, type = "class")

# evaluate the results
conf_final_dt <- confusionMatrix(tree.predict, test_both_original$Performance.Tag.y)



# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 4439 2212
# 1 2572 4751
#
# Accuracy : 0.6576
# 95% CI : (0.6497, 0.6655)
# No Information Rate : 0.5017
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.3154
# Mcnemar's Test P-Value : 2.099e-07
#
#            Sensitivity : 0.6331
#            Specificity : 0.6823
#         Pos Pred Value : 0.6674
#         Neg Pred Value : 0.6488
#             Prevalence : 0.5017
#         Detection Rate : 0.3177
```

```
#    Detection Prevalence : 0.4760
#       Balanced Accuracy : 0.6577
#
#        'Positive' Class : 0


#################################  the above is One of the Best Model in Decision tree #########################

###############################################################################################################



#5 Cross test to choose CP ------------------------------------------------------------

# set the number of folds in cross test to 5
tree.control = trainControl(method = "cv", number = 20)

# set the search space for CP
tree.grid = expand.grid(cp = seq(0.001, 0.003, 0.0001))

# train model
tree.model <- train(Performance.Tag.y ~ .,
                    data = train_both_original,
                    method = "rpart",
                    metric = "Accuracy",
                    trControl = tree.control,
                    tuneGrid = tree.grid,
                    control = rpart.control(minsplit = 10,
                                            minbucket = 10))

# look at cross validated model results
tree.model

# look at best value of hyperparameter
```

```
tree.model$bestTune

# make predictions on test set
tree.predict <- predict.train(tree.model, test_both_original)

# accuracy
confusionMatrix(tree.predict, test_both_original$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 4439 2212
# 1 2572 4751
#
# Accuracy : 0.6576
# 95% CI : (0.6497, 0.6655)
# No Information Rate : 0.5017
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.3154
# Mcnemar's Test P-Value : 2.099e-07
#
#            Sensitivity : 0.6331
#            Specificity : 0.6823
#         Pos Pred Value : 0.6674
#         Neg Pred Value : 0.6488
#             Prevalence : 0.5017
#         Detection Rate : 0.3177
#   Detection Prevalence : 0.4760
#      Balanced Accuracy : 0.6577
#
#       'Positive' Class : 0
```

```
# plot CP vs Accuracy

library(ggplot2)

accuracy_graph <- data.frame(tree.model$results)

ggplot(data = accuracy_graph, aes(x = cp, y = Accuracy*100)) +

  geom_line() +

  geom_point() +

  labs(x = "Complexity Parameter (CP)", y = "Accuracy", title = "CP vs Accuracy")



####################  Decision tree for both smapling ends ######################



####################  Decision tree for Synthesys smapling starts ######################



tree.model <- rpart(Performance.Tag.y ~ .,                    # formula

                    data = train_synthesis_original,                  # training data

                    method = "class")              # classification or regression


# display decision tree


prp(tree.model)


# make predictions on the test set


tree.predict <- predict(tree.model, test_synthesis_original, type = "class")


# evaluate the results

confusionMatrix(tree.predict, test_synthesis_original$Performance.Tag.y)



# Confusion Matrix and Statistics

#

# Reference

# Prediction    0    1
```

```
# 0 3511 2026

# 1 3463 5795

#

# Accuracy : 0.629

# 95% CI : (0.6212, 0.6368)

# No Information Rate : 0.5286

# P-Value [Acc > NIR] : < 2.2e-16

#

# Kappa : 0.2472

# Mcnemar's Test P-Value : < 2.2e-16

#

# Sensitivity : 0.5034

# Specificity : 0.7410

# Pos Pred Value : 0.6341

# Neg Pred Value : 0.6259

# Prevalence : 0.4714

# Detection Rate : 0.2373

# Detection Prevalence : 0.3742

# Balanced Accuracy : 0.6222

#

# 'Positive' Class : 0


# Change the algorithm to "information gain" instead of default "gini" ----------------------

tree.model <- rpart(Performance.Tag.y ~ .,                    # formula

                   data = train_synthesis_original,                  # training data

                   method = "class",              # classification or regression

                   parms = list(split = "information"))


# display decision tree


prp(tree.model)


# make predictions on the test set

tree.predict <- predict(tree.model, test_synthesis_original, type = "class")
```

```
# evaluate the results
confusionMatrix(tree.predict, test_synthesis_original$Performance.Tag.y)


# ### No Change in results

#3 Tune the hyperparameters ---------------------------------------------------------
tree.model <- rpart(Performance.Tag.y ~ .,                          # formula
              data = train_synthesis_original,                      # training data
              method = "class",                    # classification or regression
              control = rpart.control(minsplit = 10,  # min observations for node
                          minbucket = 10, # min observations for leaf node
                          cp = 0.003))      # complexity parameter

# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_synthesis_original, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_synthesis_original$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 3931 2223
# 1 3043 5598
#
# Accuracy : 0.6441
# 95% CI : (0.6363, 0.6518)
# No Information Rate : 0.5286
# P-Value [Acc > NIR] : < 2.2e-16
```

```
#
# Kappa : 0.2812
# Mcnemar's Test P-Value : < 2.2e-16
#
# Sensitivity : 0.5637
# Specificity : 0.7158
# Pos Pred Value : 0.6388
# Neg Pred Value : 0.6478
# Prevalence : 0.4714
# Detection Rate : 0.2657
# Detection Prevalence : 0.4160
# Balanced Accuracy : 0.6397
#
# 'Positive' Class : 0


#4 A more complex tree ------------------------------------------------------------------
tree.model <- rpart(Performance.Tag.y ~ .,                     # formula
         data = train_synthesis_original,                   # training data
         method = "class",                   # classification or regression
         control = rpart.control(minsplit = 10,     # min observations for node
                  minbucket = 10,    # min observations for leaf node
                  cp = 0.001))      # complexity parameter

# display decision tree
prp(tree.model)

# make predictions on the test set
tree.predict <- predict(tree.model, test_synthesis_original, type = "class")

# evaluate the results
confusionMatrix(tree.predict, test_synthesis_original$Performance.Tag.y)


# Confusion Matrix and Statistics
```

```
#
# Reference
# Prediction    0    1
# 0 4367 2577
# 1 2607 5244
#
# Accuracy : 0.6496
# 95% CI : (0.6419, 0.6573)
# No Information Rate : 0.5286
# P-Value [Acc > NIR] : <2e-16
#
# Kappa : 0.2968
# Mcnemar's Test P-Value : 0.6871
#
# Sensitivity : 0.6262
# Specificity : 0.6705
# Pos Pred Value : 0.6289
# Neg Pred Value : 0.6679
# Prevalence : 0.4714
# Detection Rate : 0.2952
# Detection Prevalence : 0.4693
# Balanced Accuracy : 0.6483
#
# 'Positive' Class : 0



# Cross test to choose CP ------------------------------------------------------------

# set the number of folds in cross test to 5
tree.control = trainControl(method = "cv", number = 20)

# set the search space for CP
tree.grid = expand.grid(cp = seq(0.001, 0.003, 0.0001))
```

```
# train model
tree.model <- train(Performance.Tag.y ~ .,
            data = train_synthesis_original,
            method = "rpart",
            metric = "Accuracy",
            trControl = tree.control,
            tuneGrid = tree.grid,
            control = rpart.control(minsplit = 5,
                        minbucket = 5))

# look at cross validated model results
tree.model

# look at best value of hyperparameter
tree.model$bestTune

# make predictions on test set
tree.predict <- predict.train(tree.model, test_synthesis_original)

# accuracy
confusionMatrix(tree.predict, test_synthesis_original$Performance.Tag.y)


# Confusion Matrix and Statistics
#
# Reference
# Prediction    0    1
# 0 4367 2577
# 1 2607 5244
#
# Accuracy : 0.6496
# 95% CI : (0.6419, 0.6573)
# No Information Rate : 0.5286
# P-Value [Acc > NIR] : <2e-16
```

```
#

# Kappa : 0.2968

# Mcnemar's Test P-Value : 0.6871

#

# Sensitivity : 0.6262

# Specificity : 0.6705

# Pos Pred Value : 0.6289

# Neg Pred Value : 0.6679

# Prevalence : 0.4714

# Detection Rate : 0.2952

# Detection Prevalence : 0.4693

# Balanced Accuracy : 0.6483

#

# 'Positive' Class : 0



# plot CP vs Accuracy

library(ggplot2)

accuracy_graph <- data.frame(tree.model$results)

ggplot(data = accuracy_graph, aes(x = cp, y = Accuracy*100)) +

  geom_line() +

  geom_point() +

  labs(x = "Complexity Parameter (CP)", y = "Accuracy", title = "CP vs Accuracy")




######################  Decision tree for Synthesys smapling ends #########################




##################################################################################################

######################################################################################################

################## Original Data Decision tree ends
```

```
#################################################################################################


######################################################################################################################
##################################### Decision Tree Ends #######################################################
######################################################################################################################


######################################################################################################################
#######################  Random Forest for synthesis sampling starts ##########################################
######################################################################################################################
library(randomForest)

model1 <- randomForest(Performance.Tag.y ~ ., data = train_synthesis, importance = TRUE)
model1

model2 <- randomForest(Performance.Tag.y ~ ., data = train_synthesis, ntree = 500, mtry = 6, importance = TRUE)
model2



# > model1 <- randomForest(Performance.Tag.y ~ ., data = train_synthesis, importance = TRUE)
# > model1
#
# Call:
#   randomForest(formula = Performance.Tag.y ~ ., data = train_synthesis,      importance = TRUE)
# Type of random forest: classification
# Number of trees: 500
# No. of variables tried at each split: 5
#
# OOB estimate of  error rate: 0.09%
# Confusion matrix:
#   0     1  class.error
# 0 27823    37 0.0013280689
# 1    14 28019 0.0004994114
```

```
# >
#   > model2 <- randomForest(Performance.Tag.y ~ ., data = train_synthesis, ntree = 500, mtry = 6, importance = TRUE)
# > model2
#
# Call:
#   randomForest(formula = Performance.Tag.y ~ ., data = train_synthesis,      ntree = 500, mtry = 6, importance = TRUE)
# Type of random forest: classification
# Number of trees: 500
# No. of variables tried at each split: 6
#
# OOB estimate of  error rate: 0.08%
# Confusion matrix:
#   0     1 class.error
# 0 27819    41 0.001471644
# 1     4 28029 0.000142689


predTrain <- predict(model2, train_synthesis, type = "class")
# Checking classification accuracy
table(predTrain, train_synthesis$Performance.Tag.y)


# > table(predTrain, train_synthesis$Performance.Tag.y)
#
# predTrain     0     1
# 0 27853     0
# 1     7 28033


# Predicting on Validation set
predValid <- predict(model2, test_synthesis, type = "class")
# Checking classification accuracy
mean(predValid == test_synthesis$Performance.Tag.y)
table(predValid,test_synthesis$Performance.Tag.y)
```

```
importance(model2)
varImpPlot(model2)


# Using For loop to identify the right mtry for model
a=c()
i=5
for (i in 3:8) {
  model3 <- randomForest(Performance.Tag.y ~ ., data = train_synthesis, ntree = 500, mtry = i, importance = TRUE)
  predValid <- predict(model3, test_synthesis, type = "class")
  a[i-2] = mean(predValid == test_synthesis$Performance.Tag.y)
}

a

plot(3:8,a)

# 4th model looks the best
model4 <- randomForest(Performance.Tag.y ~ ., data = train_synthesis, ntree = 500, mtry = 4, importance = TRUE)
# Predict the parformance.Tag for the test data
predValid <- predict(model4, test_synthesis , type = "class")
conf_final_rf <- confusionMatrix(predValid, test_synthesis$Performance.Tag.y)
# Accuracy, sensitivity and specificity is very high - looks like model is overfitted

# Predict the parformance.Tag for the rejected data
predValid <- predict(model4, rejected_woe , type = "class")
View(predValid)
prop.table(table(predValid))
# As per this, 99% rejected applicants would have been accepted by this model

############################################################################################
##################################Model COmaprison#########################################
############################################################################################
```

```
# conf_final - confusion matrix of final logistic regression model

# conf

model_comparison <- as.data.frame(matrix(0,4,10))

colnames(model_comparison) <- c("Model","Accuracy","Kappa","AccuracyLower","AccuracyUpper","Sensitivity",
"Specificity","Precision","Recall","F1")

model_comparison[1] <- c("LR Merged","LR Demographic","Decision Tree","Random Forest")

model_comparison[2] <- c(conf_final$overall[[1]],conf_final_demo$overall[[1]],conf_final_dt$overall[[1]],conf_final_rf$overall[[1]])

model_comparison[3] <- c(conf_final$overall[[2]],conf_final_demo$overall[[2]],conf_final_dt$overall[[2]],conf_final_rf$overall[[2]])

model_comparison[4] <- c(conf_final$overall[[3]],conf_final_demo$overall[[3]],conf_final_dt$overall[[3]],conf_final_rf$overall[[3]])

model_comparison[5] <- c(conf_final$overall[[4]],conf_final_demo$overall[[4]],conf_final_dt$overall[[4]],conf_final_rf$overall[[4]])

model_comparison[6] <- c(conf_final$byClass[[1]],conf_final_demo$byClass[[1]],conf_final_dt$byClass[[1]],conf_final_rf$byClass[[1]])

model_comparison[7] <- c(conf_final$byClass[[2]],conf_final_demo$byClass[[2]],conf_final_dt$byClass[[2]],conf_final_rf$byClass[[2]])

model_comparison[8] <- c(conf_final$byClass[[5]],conf_final_demo$byClass[[5]],conf_final_dt$byClass[[5]],conf_final_rf$byClass[[5]])

model_comparison[9] <- c(conf_final$byClass[[6]],conf_final_demo$byClass[[6]],conf_final_dt$byClass[[6]],conf_final_rf$byClass[[6]])

model_comparison[10] <- c(conf_final$byClass[[7]],conf_final_demo$byClass[[7]],conf_final_dt$byClass[[7]],conf_final_rf$byClass[[7]])

View(model_comparison)


##############################################################################################

##########################Loss and Gain Selected Model###################################

##############################################################################################

conf_final$table

#          Reference

#Prediction    No   Yes

#No        12803   328

#Yes        7273   556


Onboarding_Probable_Wrong_Customer <- conf_final$table[3][1]

Rejecting_Probable_Good_Customer <- conf_final$table[2][1]


Onboarding_Probable_Wrong_Customer_Percentage <- (Onboarding_Probable_Wrong_Customer/sum(conf_final$table))*100  #1

Rejecting_Probable_Good_Customer_Percentage <- (Rejecting_Probable_Good_Customer/sum(conf_final$table))*100



# > Onboarding_Probable_Wrong_Customer_Percentage
```

[1] 1.564885

> Rejecting_Probable_Good_Customer_Percentage

[1] 34.69943

Assuming the average monthly profit-loss for rejecting 1 good customer is Rs. 2000

Assuming the average monthly loss for onboarding a defaulter is Rs. 100000

Let's calculate the loss-gain for bank in a 100 applications

#Without model Bank will accept all 100 customer

Bank will lose

```
Bank_lose_per_100_application <- Onboarding_Probable_Wrong_Customer_Percentage*100000 -
Rejecting_Probable_Good_Customer_Percentage*2000
```

> Bank_lose_per_100_application

[1] 87089.69