Credit Scoring: Logistic + Tree-based Algorithms

Anh Dang

10 March 2019

By the data set from **Give Me Some Credit** (2012) https://www.kaggle.com/c/GiveMeSomeCredit (https://www.kaggle.com/c/GiveMeSomeCredit), which contains 150,000 observations with 11 features, this work is to illustrate some useful techniques in Credit Scoring Modelling, namely:

1. Data Preparation and Exploration at Univariate Level with rank plots for continuous variables, and the Information Value Analysis to do variable transformation and selection
2. Multivariate Analysis with several feature engineering along the analytics
3. **Synthetic Minority Over-sampling Techique (SMOTE)** for the unbalance data
4. **Standard Logistic Regression** with Step-wise selection
5. **Tree-based Recursive Partitioning**
6. **(Unbiased Non-parametric Model-Based (MOB)** with two-layer of recursive partitioning, and logistic regression at terminal node as the hybrid of (4) and (5)
7. **Chi-squared Automation Interaction Detection** as another (and very interesting) approach of tree-based algorithms, comparing to the popular Recursive Partitioning CART or Regression Tree. The algorithms is designed to be more analytical-oriented than predict-oriented.

Practical Meanings:

The purpose of this work is to briefly overview some practical and handy techniques which are applicable for credit scoring. Rather than the thorough calibriating and validating the model to obtain higher accuracy, this work is focusing on walk-through and discuss around the practical applications of these techniques.

- Data Exploration and Preparation would be mentioned
- `Logistic Regression` is covered as this is one of the most conventional approach in Credit Scoring
- However, the limitation of Logit is its dependencies in lineary and monotone relationship, and not efficient in capturining the non-linearity and interaction among features. This is a big limitation, especially in Credit Scoring for SMEs where we would expect the pattern is more varied, non-linear, and less straighforward than that of bigger companies
- In that sense, tree-based alogrithms might be good ideas as it conducts the classification taking into account the interactions, non-linearity and working well with categorical variables. The visual presentation of decision trees is easy to commcate, and could play the role as a framework for rule-based credit risk models. We would walk-through the `Recursive Partitioning` and `Chi-squared Automatic Detection (CHAID)`.
- As a hybrid approach, we discuss about `Model-Based Recursive Partitioning (MOB)` based on Logit model, to combine the advantages of both trees and logit regression. This method is also potential to be applied for segmentation in Risk modelling, and to validating the stability of model parameters among different subsets of population.

```
knitr::opts_chunk$set(echo = TRUE)
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.5.2
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(knitr)
library(kableExtra)
library(magrittr)
library(DT)
```

```
## Warning: package 'DT' was built under R version 3.5.2
```

```
library(mltools)
```

```
## Warning: package 'mltools' was built under R version 3.5.2
```

```r
library(ape)
```

```
## Warning: package 'ape' was built under R version 3.5.2
```

```r
library(caret)
```

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

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

```r
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(DMwR)
```

```
## Warning: package 'DMwR' was built under R version 3.5.2
```

```
## Loading required package: grid
```

```r
library(rpart)
library(ggplot2)
library(ggthemes)
```

```
## Warning: package 'ggthemes' was built under R version 3.5.2
```

```r
library(RColorBrewer)
```

```
## Warning: package 'RColorBrewer' was built under R version 3.5.2
```

```r
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.5.2
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(vcd)
library(party)
```

```
## Warning: package 'party' was built under R version 3.5.2
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Warning: package 'modeltools' was built under R version 3.5.2
```

```
## Loading required package: stats4
## Loading required package: strucchange
```

```
## Warning: package 'strucchange' was built under R version 3.5.2
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
##
## Attaching package: 'party'
```

```
## The following object is masked from 'package:ape':
##
##     where
```

```
library(partykit)
```

```
## Warning: package 'partykit' was built under R version 3.5.2
```

```
## Loading required package: libcoin
```

```
## Warning: package 'libcoin' was built under R version 3.5.2
```

```
##
## Attaching package: 'partykit'
```

```
## The following objects are masked from 'package:party':
##
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner,
##     node_surv, node_terminal, varimp
```

```
library(plotROC)
```

```
## Warning: package 'plotROC' was built under R version 3.5.2
```

```
library(CHAID)

source("./Handy_functions/summary_handy.R")
source("./Handy_functions/rank_plot.R")
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

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

```
##
## Attaching package: 'pROC'
## The following object is masked from 'package:plotROC':
##
##     ggroc
```

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

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
source("./Handy_functions/var_treat.R")
source("./Handy_functions/gb_integer_cat.R")
source("./Handy_functions/split_sample.R")
source("./Handy_functions/univariate_cont_plot.R")
```

1. Data Set

The data set used as the illustration in this document is the public dataset in Kaggle, under the competition named **Give Me Some Credit** (2012) https://www.kaggle.com/c/GiveMeSomeCredit (https://www.kaggle.com/c/GiveMeSomeCredit), which contains 150,000 observations with 11 features.

> Based on this Data, Credit scoring algorithms is built, which make a guess at the probability of default, are the method banks use to determine whether or not a loan should be granted. This competition requires participants to improve on the state of the art in credit scoring, by predicting the probability that somebody will experience financial distress in the next two years.

The structure and summary of data as below:

- **Good/Bad Target Variable**: `SeriousDlqnin2yrs` , Person experienced 90 days past due delinquency or worse (Technical Default), take value 0 (good) and 1 (bad)
- All preditors are numeric/integer (which make our life a little bit easier, but we can try another data sets with more categorical variables)

```
cs_training <- read.csv("./Data/cs-training.csv") %>% select(-X)
train_set = cs_training
str(train_set)
```

```
## 'data.frame':    150000 obs. of  11 variables:
##  $ SeriousDlqin2yrs                    : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ RevolvingUtilizationOfUnsecuredLines: num  0.766 0.957 0.658 0.234 0.907 ...
##  $ age                                 : int  45 40 38 30 49 74 57 39 27 57 ...
##  $ NumberOfTime30.59DaysPastDueNotWorse: int  2 0 1 0 1 0 0 0 0 0 ...
##  $ DebtRatio                           : num  0.803 0.1219 0.0851 0.036 0.0249 ...
##  $ MonthlyIncome                       : int  9120 2600 3042 3300 63588 3500 NA 3500 NA 23684 ...
##  $ NumberOfOpenCreditLinesAndLoans     : int  13 4 2 5 7 3 8 8 2 9 ...
##  $ NumberOfTimes90DaysLate             : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ NumberRealEstateLoansOrLines        : int  6 0 0 0 1 1 3 0 0 4 ...
##  $ NumberOfTime60.89DaysPastDueNotWorse: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NumberOfDependents                  : int  2 1 0 0 0 1 0 0 NA 2 ...
```

```
train_set %>% summary_handy(quantile = T) %>%
  select(Variable, Type, Unique, NA_rate, Mean, Min, Max, Q0.25, Q0.50, Q0.75, Q0.95) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  kable %>% kable_styling()
```

Variable	Type	Unique	NA_rate	Mean	Min	Max	Q0.25	Q0.50	Q0.75	Q0.95
SeriousDlqin2yrs	integer	2	0.00	0.07	0	1	0.00	0.00	0.00	1.0
RevolvingUtilizationOfUnsecuredLines	numeric	125728	0.00	6.05	0	50708	0.03	0.15	0.56	1.0
age	integer	86	0.00	52.30	0	109	41.00	52.00	63.00	78.0
NumberOfTime30.59DaysPastDueNotWorse	integer	16	0.00	0.42	0	98	0.00	0.00	0.00	2.0

Variable	Type	Unique	NA_rate	Mean	Min	Max	Q0.25	Q0.50	Q0.75	Q0.95
DebtRatio	numeric	114194	0.00	353.01	0	329664	0.18	0.37	0.87	2449.0
MonthlyIncome	integer	13595	0.20	6670.22	0	3008750	3400.00	5400.00	8249.00	14587.6
NumberOfOpenCreditLinesAndLoans	integer	58	0.00	8.45	0	58	5.00	8.00	11.00	18.0
NumberOfTimes90DaysLate	integer	19	0.00	0.27	0	98	0.00	0.00	0.00	1.0
NumberRealEstateLoansOrLines	integer	28	0.00	1.02	0	54	0.00	1.00	2.00	3.0
NumberOfTime60.89DaysPastDueNotWorse	integer	13	0.00	0.24	0	98	0.00	0.00	0.00	1.0
NumberOfDependents	integer	14	0.03	0.76	0	20	0.00	0.00	1.00	3.0

2. Initial Data Cleaning

- `MontlyIncome` and `NumerOfDependents` has the missing rate of 20% and 3%, respectively. We would need to do imputation for these missing values by median values
- The `Min`, `Max` and Distribution shows that some cap/floor should be applied for the numeric variable, considering the table of distribution, we cap/floor at 5% and 95% quantile

```
train_set %<>%
  mutate(
    RevolvingUtilizationOfUnsecuredLines = var_treat(RevolvingUtilizationOfUnsecuredLines,
                                                     q_low = 0.05, q_up = 0.95,
                                                     impute_val = median(train_set$RevolvingUtilizationOfUnsecuredLines,
                                                                         na.rm=T)),
    MonthlyIncome = var_treat(MonthlyIncome, q_low = 0.05, q_up = 0.95,
                              impute_val = median(train_set$MonthlyIncome, na.rm = T)),
    NumberOfDependents = ifelse(is.na(NumberOfDependents),
                                median(NumberOfDependents, na.rm = T),
                                NumberOfDependents)
  )
```

2. Univariate Analysis

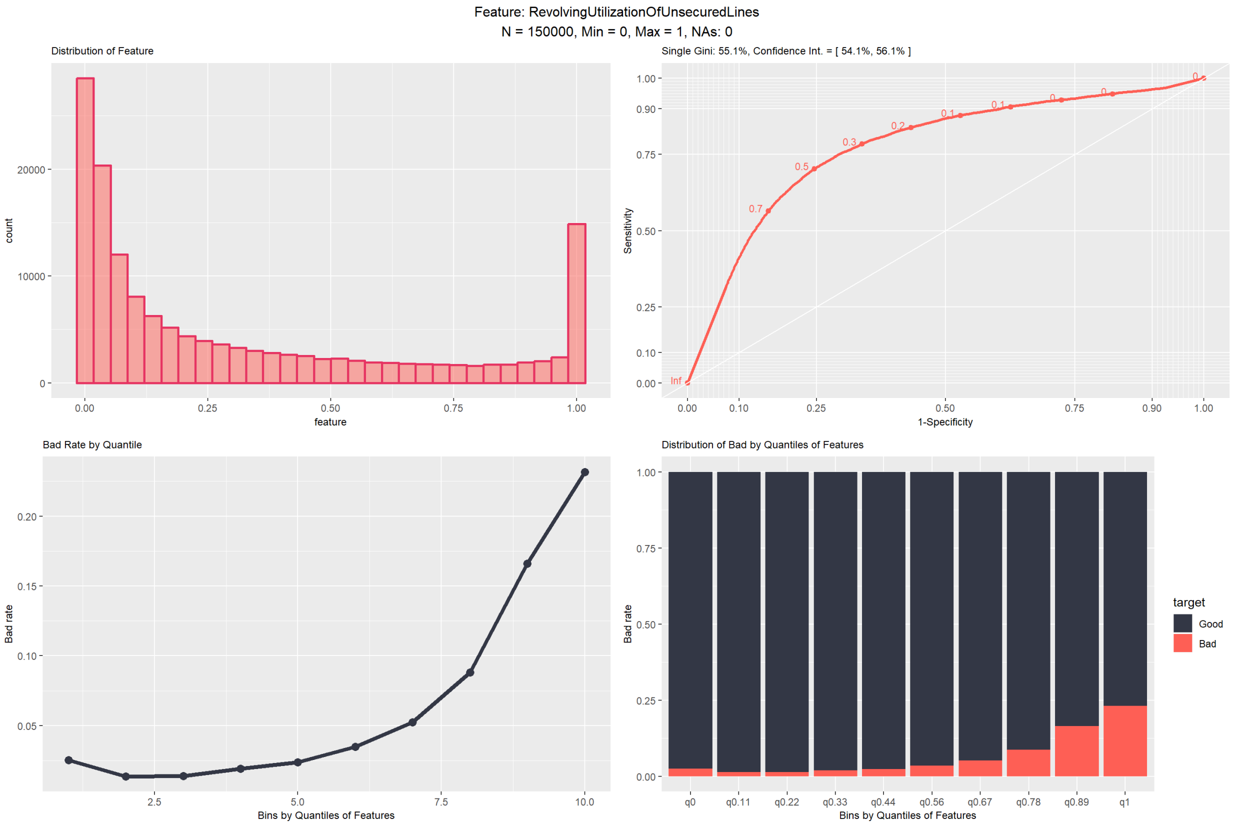
For numeric variables, we conduct the rank plots to compare the association between the predictor and Good/Bad flag. We see that:

- `RevolvingUtilizationOfUnsecuredLines` : very predictive, with single gini up to 54.1%, this predictor represent the utilization of credit limits in credit card after adjusting several factors (See: Data Dictionary). Thus, the correlation is positive, which is intuitive.
- `MonthlyIncome` : moderately predictive with Single Gini of 14.3%, with the negative correlation with Good/Bad. It means that individuals with higher monthly income is less likely to have the financial distress.
- `Debt Ratio` : gives us a U-curve relationship. That is reasonable what a ‘good’ borrower is likely to be granted some ‘healthy’ debt, yet when the Debt Ratio increases to over 0.287 (as in graph after capping), the credit risk increase.

2.1. Continuous Variables

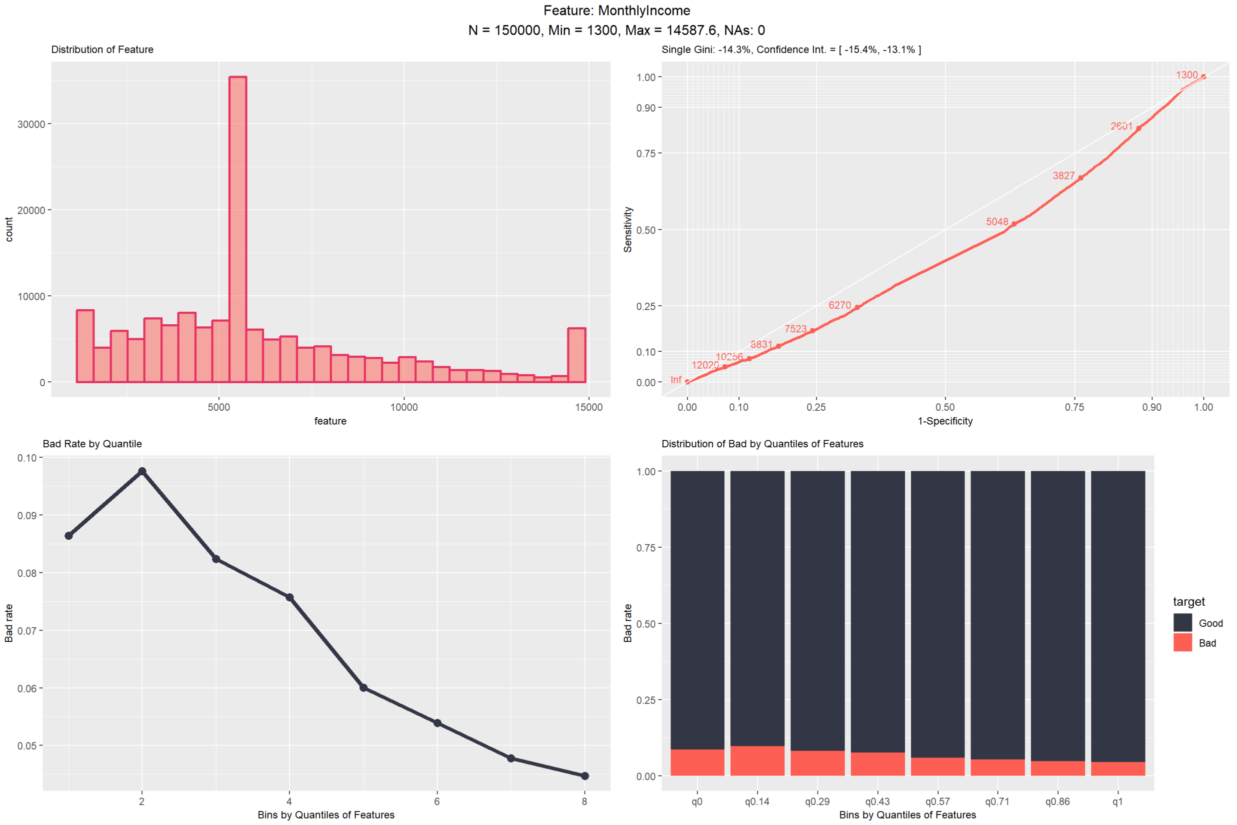
```
univariate_cont_plot(x = train_set$RevolvingUtilizationOfUnsecuredLines,
                     y = train_set$SeriousDlqin2yrs,
                     varname = 'RevolvingUtilizationOfUnsecuredLines')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



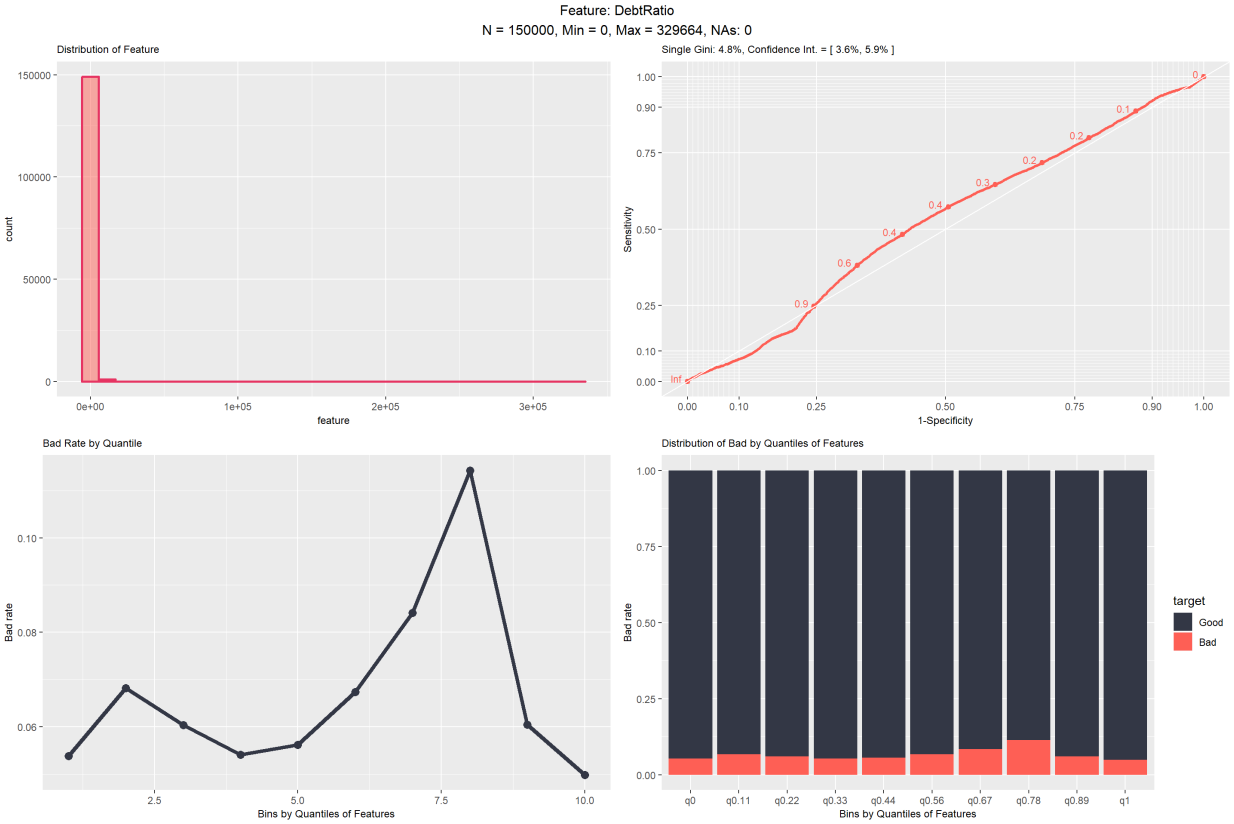
```
univariate_cont_plot(x = train_set$MonthlyIncome,
                     y = train_set$SeriousDlqin2yrs,
                     varname = 'MonthlyIncome')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
univariate_cont_plot(x = train_set$DebtRatio,
                     y = train_set$SeriousDlqin2yrs,
                     varname = 'DebtRatio')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

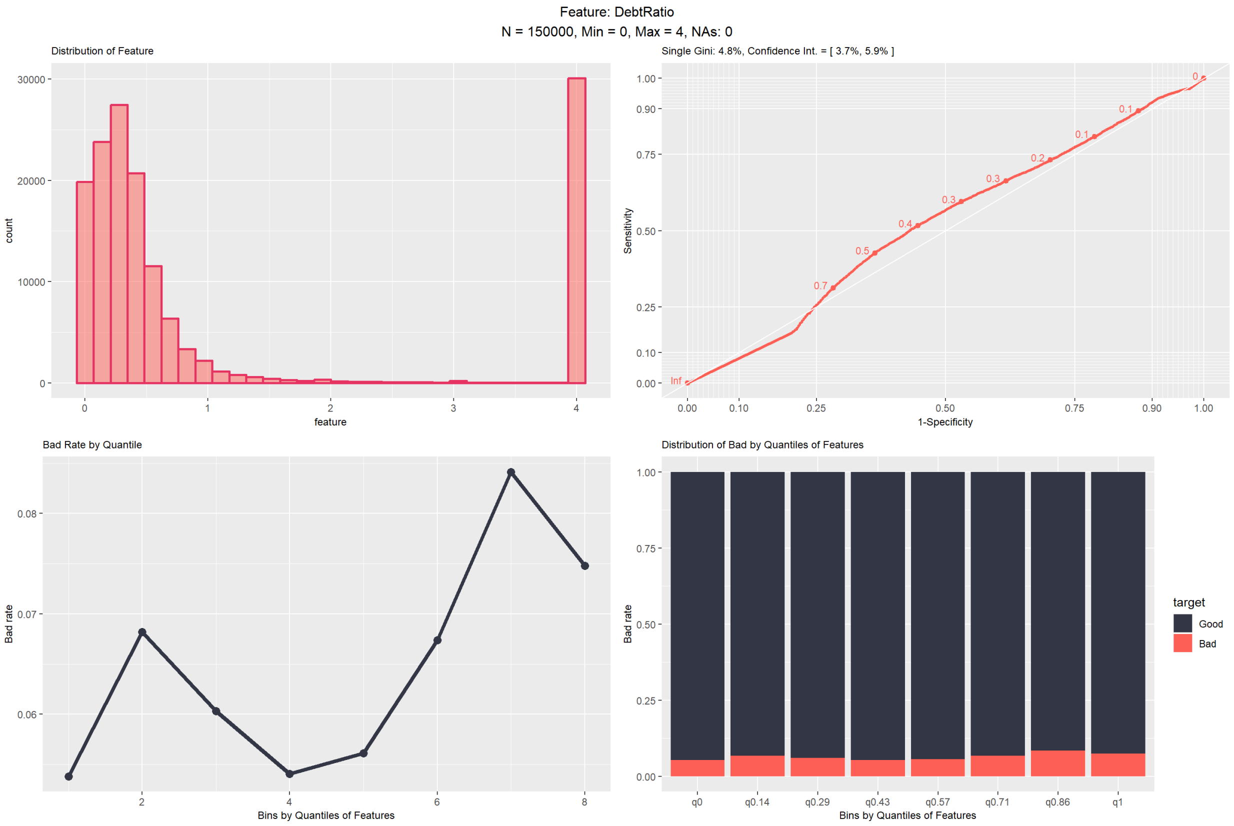


Variable Treatments

Considering the rank plots, we cap `DebtRatio` at 4

```
train_set %<>%
  mutate(DebtRatio = ifelse(DebtRatio > 4, 4, DebtRatio))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



### 2.2. Discrete Variables

For Discrete Variables, we consider the Univariate Analysis by below metrics:

- **Weight of Evidence (WOE):** shows predictive power of predictors. It increases with credit scoring for the power of seperating good/bad.
- **Information Value (IV):** based on WOE, this helps to select variables by the IV (higher IV, more predictive)
- **Efficiency**: Abs(%Good - %Bad)/2

For `Age` and `NumberOfOpenCreditLinesAndLoans` , we cut them into 5 bins by quantiles. For the others, we consider the table of IV to cut at the value of IV below 2.

```
train_set %<>%
  mutate_at(vars(-RevolvingUtilizationOfUnsecuredLines,
                 -MonthlyIncome,
                 -DebtRatio,
                 -SeriousDlqin2yrs,
                 -NumberOfTime30.59DaysPastDueNotWorse,
                 -NumberOfTime60.89DaysPastDueNotWorse,
                 -NumberOfTimes90DaysLate,
                 -NumberOfDependents
                 ),
            funs(bin_data(., bins = 3, binType = 'quantile')))
```

```
gb_integer_cat(train_set$age, train_set$SeriousDlqin2yrs) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  datatable()
```

Show 10 **entries**

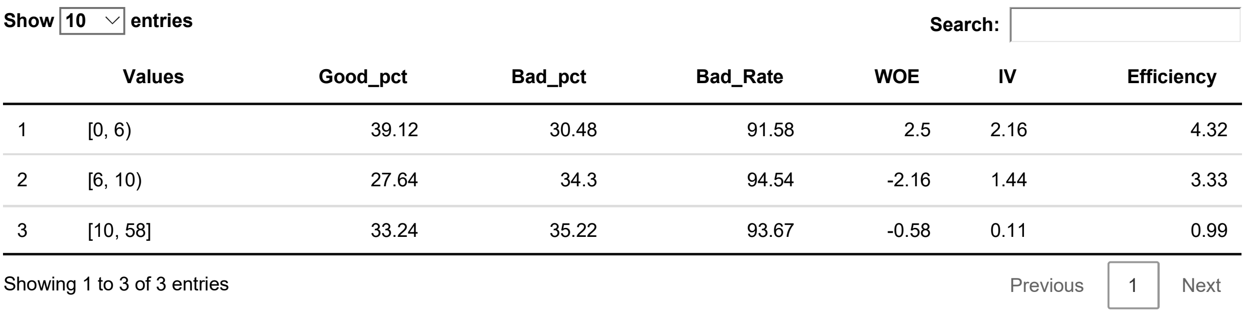
Search:

	Values	Good_pct	Bad_pct	Bad_Rate	WOE	IV	Efficiency
1	[0, 45)	47.23	30.77	90.09	4.28	7.04	8.23
2	[45, 59)	36.15	33.56	92.84	0.74	0.19	1.29
3	[59, 109]	16.63	35.67	96.77	-7.63	14.53	9.52

Showing 1 to 3 of 3 entries

Previous 1 Next

```
gb_integer_cat(train_set$NumberOfOpenCreditLinesAndLoans, train_set$SeriousDlqin2yrs) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  datatable()
```



Show 10 entries

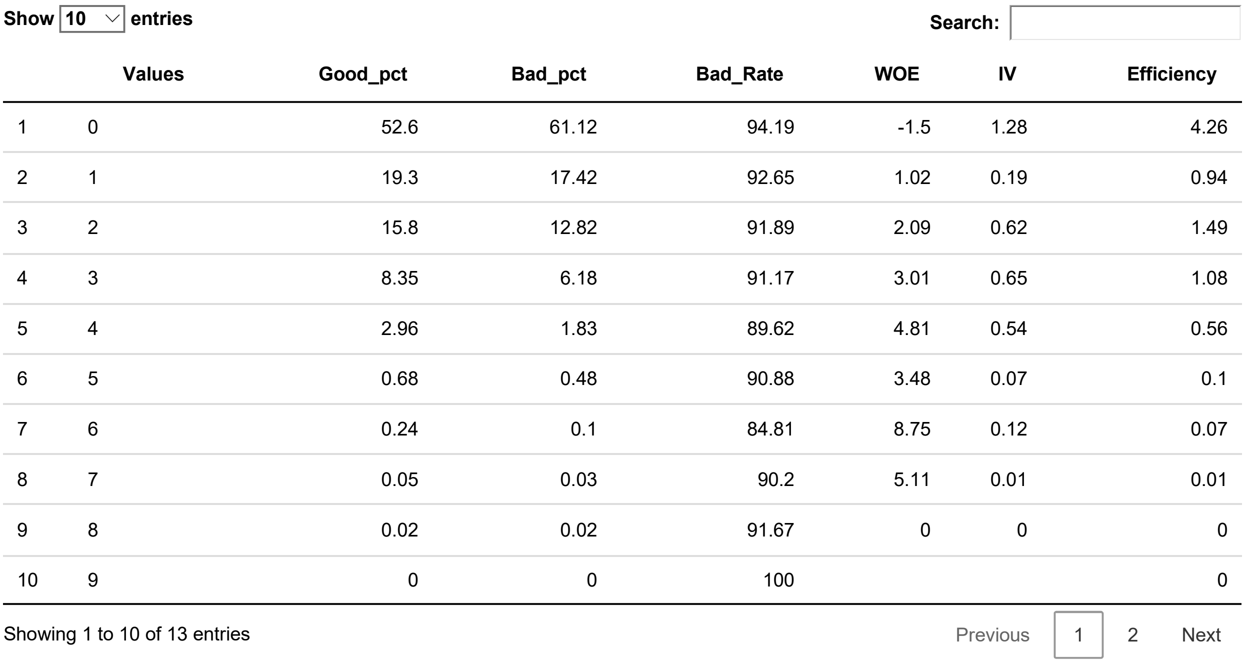
Search:

	Values	Good_pct	Bad_pct	Bad_Rate	WOE	IV	Efficiency
1	[0, 6)	39.12	30.48	91.58	2.5	2.16	4.32
2	[6, 10)	27.64	34.3	94.54	-2.16	1.44	3.33
3	[10, 58]	33.24	35.22	93.67	-0.58	0.11	0.99

Showing 1 to 3 of 3 entries

Previous 1 Next

```
gb_integer_cat(train_set$NumberOfDependents, train_set$SeriousDlqin2yrs) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  datatable()
```



Show 10 entries

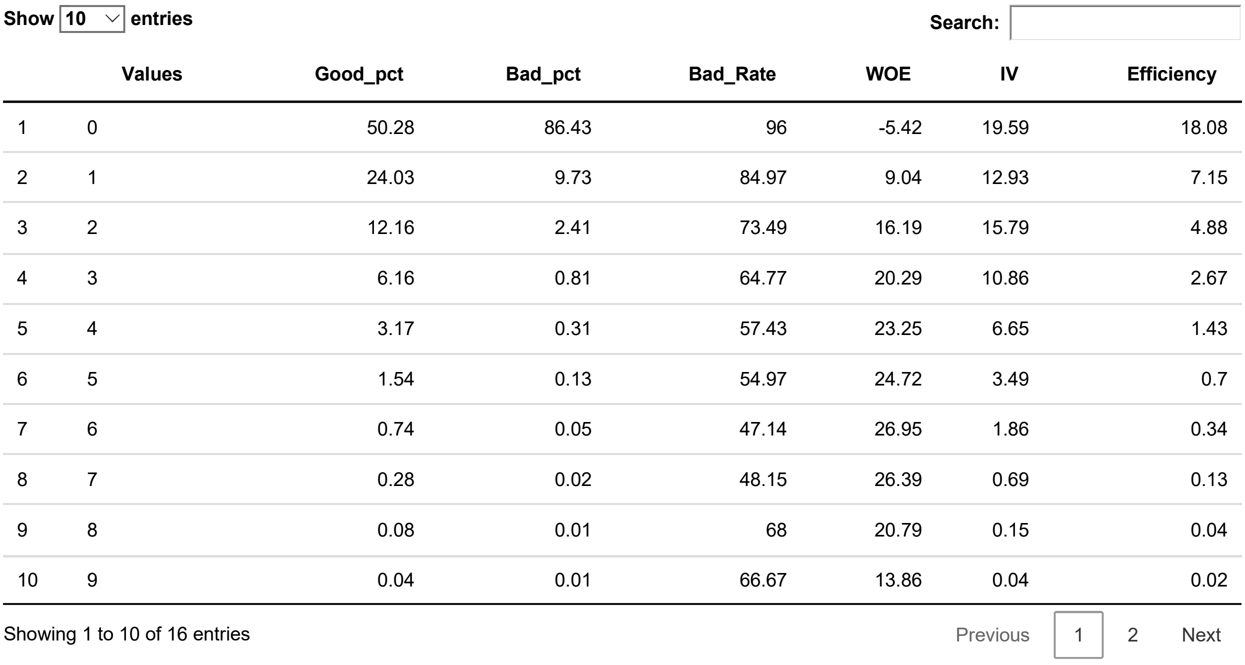
Search:

	Values	Good_pct	Bad_pct	Bad_Rate	WOE	IV	Efficiency
1	0	52.6	61.12	94.19	-1.5	1.28	4.26
2	1	19.3	17.42	92.65	1.02	0.19	0.94
3	2	15.8	12.82	91.89	2.09	0.62	1.49
4	3	8.35	6.18	91.17	3.01	0.65	1.08
5	4	2.96	1.83	89.62	4.81	0.54	0.56
6	5	0.68	0.48	90.88	3.48	0.07	0.1
7	6	0.24	0.1	84.81	8.75	0.12	0.07
8	7	0.05	0.03	90.2	5.11	0.01	0.01
9	8	0.02	0.02	91.67	0	0	0
10	9	0	0	100			0

Showing 1 to 10 of 13 entries

Previous 1 2 Next

```
gb_integer_cat(train_set$NumberOfTime30.59DaysPastDueNotWorse, train_set$SeriousDlqin2yrs) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  datatable()
```



Show 10 entries

Search:

	Values	Good_pct	Bad_pct	Bad_Rate	WOE	IV	Efficiency
1	0	50.28	86.43	96	-5.42	19.59	18.08
2	1	24.03	9.73	84.97	9.04	12.93	7.15
3	2	12.16	2.41	73.49	16.19	15.79	4.88
4	3	6.16	0.81	64.77	20.29	10.86	2.67
5	4	3.17	0.31	57.43	23.25	6.65	1.43
6	5	1.54	0.13	54.97	24.72	3.49	0.7
7	6	0.74	0.05	47.14	26.95	1.86	0.34
8	7	0.28	0.02	48.15	26.39	0.69	0.13
9	8	0.08	0.01	68	20.79	0.15	0.04
10	9	0.04	0.01	66.67	13.86	0.04	0.02

Showing 1 to 10 of 16 entries

Previous 1 2 Next

```
gb_integer_cat(train_set$NumberOfTime60.89DaysPastDueNotWorse, train_set$SeriousDlqin2yrs) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  datatable()
```

Show 10 **entries** **Search:**

	Values	Good_pct	Bad_pct	Bad_Rate	WOE	IV	Efficiency
1	0	72.37	96.55	94.9	-2.88	6.96	12.09
2	1	17.72	2.82	68.99	18.38	27.39	7.45
3	2	5.6	0.4	49.82	26.39	13.72	2.6
4	3	1.8	0.1	43.4	28.9	4.91	0.85
5	4	0.65	0.03	38.1	30.76	1.91	0.31
6	5	0.21	0.01	38.24	30.45	0.61	0.1
7	6	0.12	0	25			0.06
8	7	0.05	0	44.44			0.02
9	8	0.01	0	50		0	0
10	9	0	0	100			0

Showing 1 to 10 of 13 entries Previous 1 2 Next

```
gb_integer_cat(train_set$NumberOfTimes90DaysLate, train_set$SeriousDlqin2yrs) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  datatable()
```

Show 10 **entries** **Search:**

	Values	Good_pct	Bad_pct	Bad_Rate	WOE	IV	Efficiency
1	0	65.37	96.52	95.37	-3.9	12.15	15.57
2	1	17.6	2.48	66.34	19.6	29.64	7.56
3	2	7.74	0.56	50.1	26.26	18.85	3.59
4	3	3.84	0.2	42.28	29.55	10.76	1.82
5	4	1.94	0.07	32.99	33.22	6.21	0.94
6	5	0.83	0.03	36.64	33.2	2.66	0.4
7	6	0.48	0.02	40	31.78	1.46	0.23
8	7	0.31	0.01	18.42	34.34	1.03	0.15
9	8	0.15	0	28.57			0.08
10	9	0.14	0	26.32			0.07

Showing 1 to 10 of 19 entries Previous 1 2 Next

Variable Treatments

Based on the table of IV, we would group `NumberOfDependents`, `NumberOfTime30.59DaysPastDueNotWorse`, `NumberOfTime60.89DaysPastDueNotWorse`, `NumberOfTimes90DaysLate` at the cut-off that the IV declines to below 2.

```
train_set %<>%
  mutate(NumberOfDependents = ifelse(NumberOfDependents > 0, 1, NumberOfDependents),
         NumberOfTime30.59DaysPastDueNotWorse = ifelse(NumberOfTime30.59DaysPastDueNotWorse > 4,
                                                      4, NumberOfTime30.59DaysPastDueNotWorse),
         NumberOfTime60.89DaysPastDueNotWorse = ifelse(NumberOfTime60.89DaysPastDueNotWorse > 3,
                                                      3, NumberOfTime60.89DaysPastDueNotWorse),
         NumberOfTimes90DaysLate = ifelse(NumberOfTimes90DaysLate > 3,
                                         3, NumberOfTimes90DaysLate)
         )
```

We can see that after treating, the IV generally shows up better among levels of variables.

```
gb_integer_cat(train_set$NumberOfDependents, train_set$SeriousDlqin2yrs) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  datatable()
```

Show 10 **entries** **Search:**

	Values	Good_pct	Bad_pct	Bad_Rate	WOE	IV	Efficiency
1	0	52.6	61.12	94.19	-1.5	1.28	4.26
2	1	47.4	38.88	91.97	1.98	1.69	4.26

Showing 1 to 2 of 2 entries Previous 1 Next

```
gb_integer_cat(train_set$NumberOfTime30.59DaysPastDueNotWorse, train_set$SeriousDlqin2yrs) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  datatable()
```

Show 10 **entries** **Search:**

	Values	Good_pct	Bad_pct	Bad_Rate	WOE	IV	Efficiency
1	0	50.28	86.43	96	-5.42	19.59	18.08
2	1	24.03	9.73	84.97	9.04	12.93	7.15
3	2	12.16	2.41	73.49	16.19	15.79	4.88
4	3	6.16	0.81	64.77	20.29	10.86	2.67
5	4	7.37	0.61	53.73	24.92	16.85	3.38

Showing 1 to 5 of 5 entries Previous 1 Next

```
gb_integer_cat(train_set$NumberOfTime60.89DaysPastDueNotWorse, train_set$SeriousDlqin2yrs) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  datatable()
```

Show 10 **entries** **Search:**

	Values	Good_pct	Bad_pct	Bad_Rate	WOE	IV	Efficiency
1	0	72.37	96.55	94.9	-2.88	6.96	12.09
2	1	17.72	2.82	68.99	18.38	27.39	7.45
3	2	5.6	0.4	49.82	26.39	13.72	2.6
4	3	4.31	0.23	42.78	29.31	11.96	2.04

Showing 1 to 4 of 4 entries Previous 1 Next

```
gb_integer_cat(train_set$NumberOfTimes90DaysLate, train_set$SeriousDlqin2yrs) %>%
  mutate_if(is.numeric, funs(round(.,2))) %>%
  datatable()
```

Show 10 **entries** **Search:**

	Values	Good_pct	Bad_pct	Bad_Rate	WOE	IV	Efficiency
1	0	65.37	96.52	95.37	-3.9	12.15	15.57
2	1	17.6	2.48	66.34	19.6	29.64	7.56
3	2	7.74	0.56	50.1	26.26	18.85	3.59
4	3	9.29	0.44	39.55	30.5	26.99	4.42

Showing 1 to 4 of 4 entries

Previous 1 Next

Discrete Variable Plots

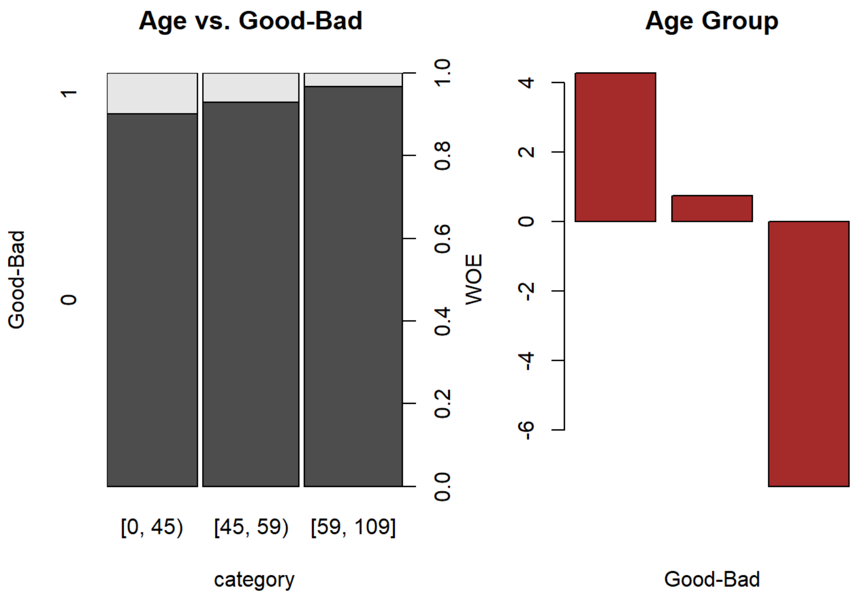
Age Groups

The bad rate declines by the higher age group. It means that the older borrowers become, the safer they are. This is interesting pattern, as the older people might be more cautious in spending and saving, that make them less likely to have a financial distress (comparing to young age).

```
Age <- gb_integer_cat(train_set$age, train_set$SeriousDlqin2yrs)
op1<-par(mfrow=c(1,2), new=TRUE)
```

```
## Warning in par(mfrow = c(1, 2), new = TRUE): calling par(new=TRUE) with no
## plot
```

```
plot(as.factor(train_set$age), as.factor(train_set$SeriousDlqin2yrs),
     ylab="Good-Bad", xlab="category",
     main="Age vs. Good-Bad ")
barplot(Age$WOE, col="brown", names.arg=c(Age$Levels),
        main="Age Group",
        xlab="Good-Bad",
        ylab="WOE")
```



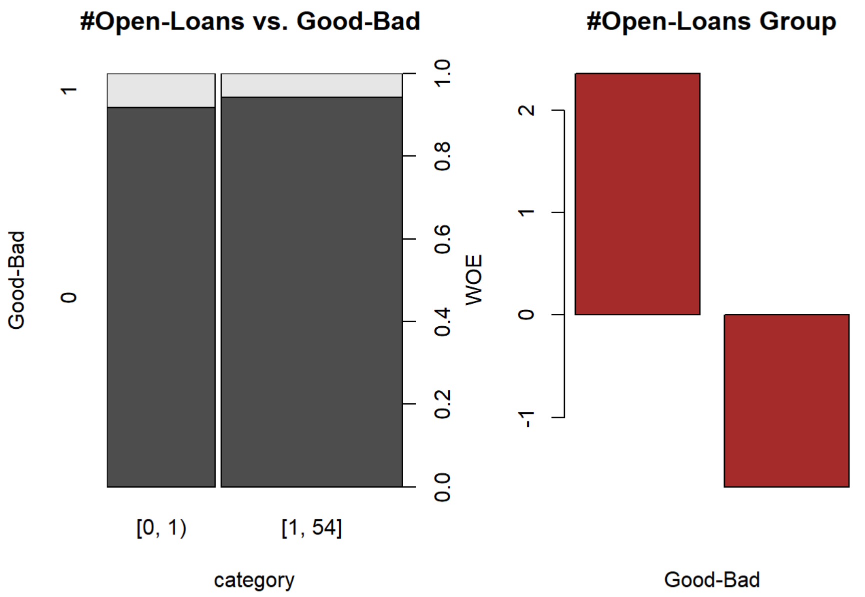
Open Loans

The bad rate is higher for the group with < 4 open loans, whom are likely to be out of risk appetite, then generally they do not win a loan/credit for themselves. However, the existence of loans have the diminishing positive signal, as once borrowers have too much loans, it means that they could easier to face a financial distress.

```
real_estate <- gb_integer_cat(train_set$NumberRealEstateLoansOrLines, train_set$SeriousDlqin2yrs)
op1<-par(mfrow=c(1,2), new=TRUE)
```

```
## Warning in par(mfrow = c(1, 2), new = TRUE): calling par(new=TRUE) with no
## plot
```

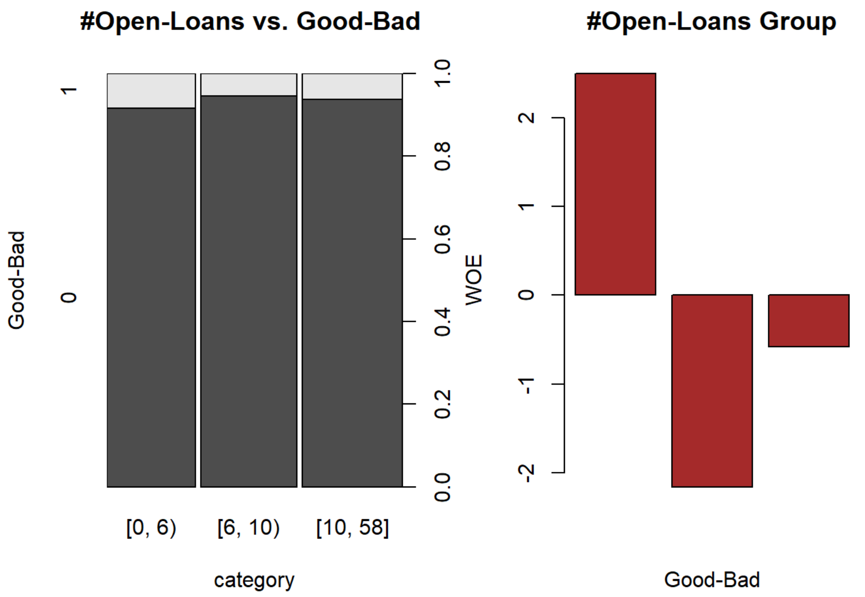
```
plot(as.factor(train_set$NumberRealEstateLoansOrLines), as.factor(train_set$SeriousDlqin2yrs),
     ylab="Good-Bad", xlab="category",
     main="#Open-Loans vs. Good-Bad ")
barplot(real_estate$WOE, col="brown", names.arg=c(real_estate$Levels),
        main="#Open-Loans Group",
        xlab="Good-Bad",
        ylab="WOE")
```



```
credit_lines <- gb_integer_cat(train_set$NumberOfOpenCreditLinesAndLoans, train_set$SeriousDlqin2yrs)
op1<-par(mfrow=c(1,2), new=TRUE)
```

```
## Warning in par(mfrow = c(1, 2), new = TRUE): calling par(new=TRUE) with no
## plot
```

```
plot(as.factor(train_set$NumberOfOpenCreditLinesAndLoans), as.factor(train_set$SeriousDlqin2yrs),
     ylab="Good-Bad", xlab="category",
     main="#Open-Loans vs. Good-Bad ")
barplot(credit_lines$WOE, col="brown", names.arg=c(credit_lines$Levels),
        main="#Open-Loans Group",
        xlab="Good-Bad",
        ylab="WOE")
```



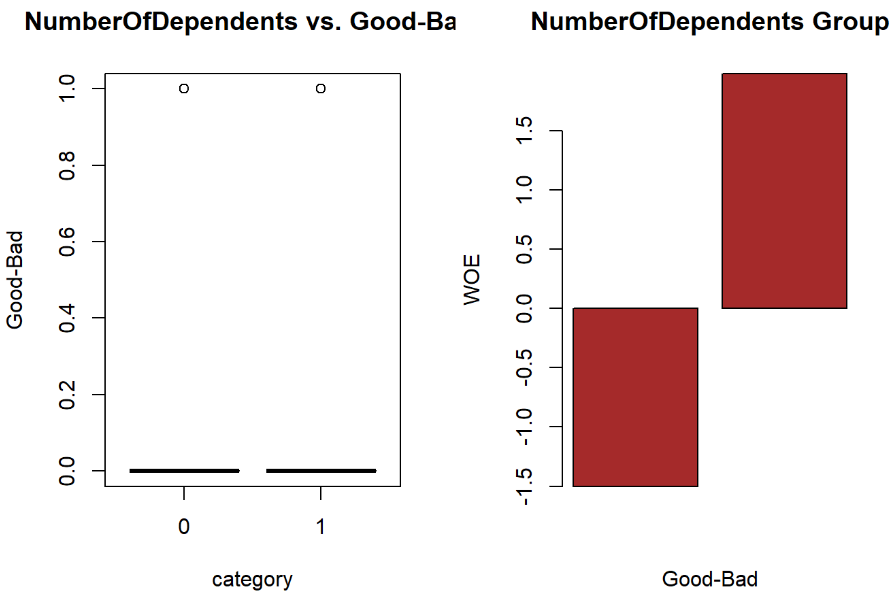
Number of Dependents

Obviously, the dependents are likely financial burden. That WOE is negative for people having no dependents.

```
dependent <- gb_integer_cat(train_set$NumberOfDependents, train_set$SeriousDlqin2yrs)
op1<-par(mfrow=c(1,2), new=TRUE)
```

```
## Warning in par(mfrow = c(1, 2), new = TRUE): calling par(new=TRUE) with no
## plot
```

```
plot(as.factor(train_set$NumberOfDependents), train_set$SeriousDlqin2yrs,
     ylab="Good-Bad", xlab="category",
     main="NumberOfDependents vs. Good-Bad ")
barplot(dependent$WOE, col="brown", names.arg=c(dependent$Levels),
        main="NumberOfDependents Group",
        xlab="Good-Bad",
        ylab="WOE")
```



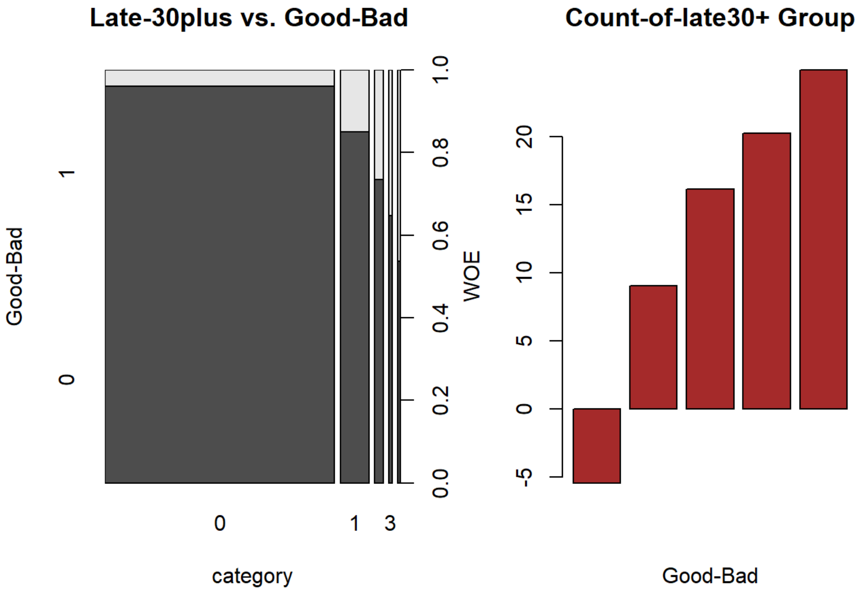
Number of Late Payments

For different windows of late payments, the pattern is the same. People with No ever-late payment is much better, while the number of being late increase, their probability of being bad (by the definition) also increases.

```
l30 <- gb_integer_cat(train_set$NumberOfTime30.59DaysPastDueNotWorse, train_set$SeriousDlqin2yrs)
op1<-par(mfrow=c(1,2), new=TRUE)
```

```
## Warning in par(mfrow = c(1, 2), new = TRUE): calling par(new=TRUE) with no
## plot
```

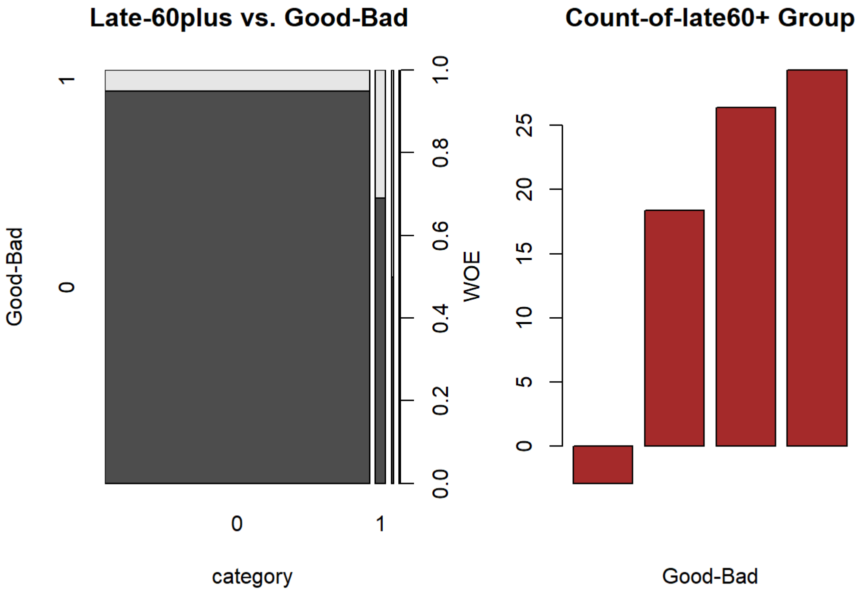
```
plot(as.factor(train_set$NumberOfTime30.59DaysPastDueNotWorse),
     as.factor(train_set$SeriousDlqin2yrs),
     ylab="Good-Bad", xlab="category",
     main="Late-30plus vs. Good-Bad ")
barplot(l30$WOE, col="brown", names.arg=c(l30$Levels),
        main="Count-of-late30+ Group",
        xlab="Good-Bad",
        ylab="WOE")
```



```
l60 <- gb_integer_cat(train_set$NumberOfTime60.89DaysPastDueNotWorse, train_set$SeriousDlqin2yrs)
op1<-par(mfrow=c(1,2), new=TRUE)
```

```
## Warning in par(mfrow = c(1, 2), new = TRUE): calling par(new=TRUE) with no
## plot
```

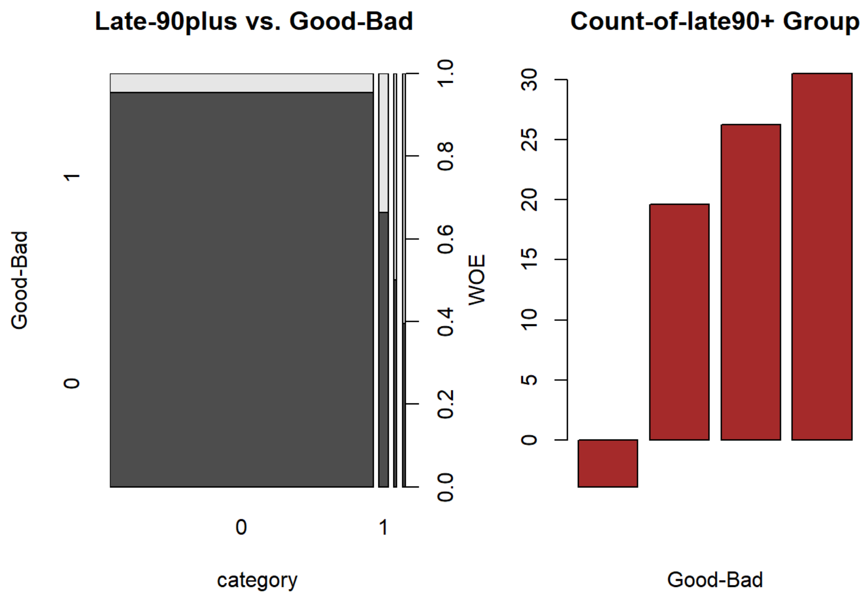
```
plot(as.factor(train_set$NumberOfTime60.89DaysPastDueNotWorse),
     as.factor(train_set$SeriousDlqin2yrs),
     ylab="Good-Bad", xlab="category",
     main="Late-60plus vs. Good-Bad ")
barplot(l60$WOE, col="brown", names.arg=c(l30$Levels),
        main="Count-of-late60+ Group",
        xlab="Good-Bad",
        ylab="WOE")
```



```
l90 <- gb_integer_cat(train_set$NumberOfTimes90DaysLate, train_set$SeriousDlqin2yrs)
op1<-par(mfrow=c(1,2), new=TRUE)
```

```
## Warning in par(mfrow = c(1, 2), new = TRUE): calling par(new=TRUE) with no
## plot
```

```
plot(as.factor(train_set$NumberOfTimes90DaysLate), as.factor(train_set$SeriousDlqin2yrs),
     ylab="Good-Bad", xlab="category",
     main="Late-90plus vs. Good-Bad ")
barplot(l90$WOE, col="brown", names.arg=c(l90$Levels),
        main="Count-of-late90+ Group",
        xlab="Good-Bad",
        ylab="WOE")
```



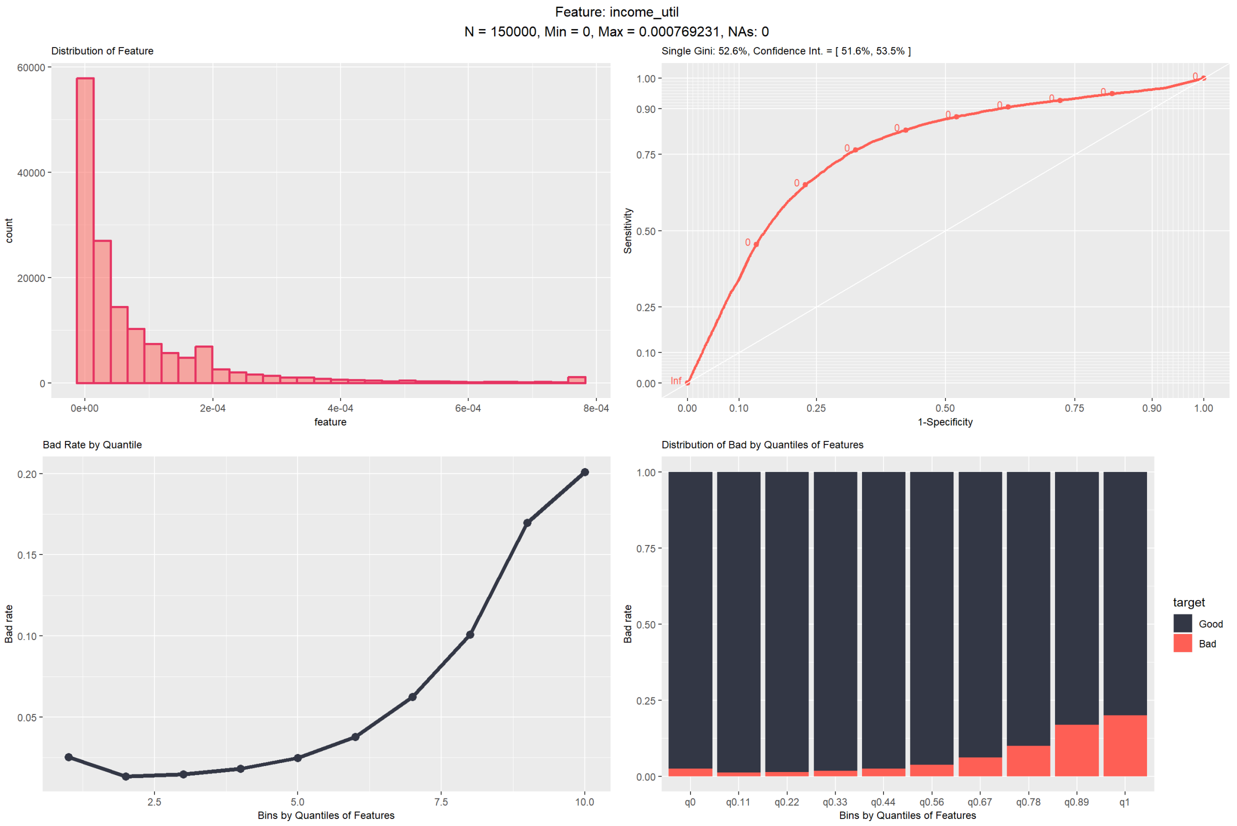
2. Feature Engineering

We are further motivated to do the feature engineering, in the sense that:

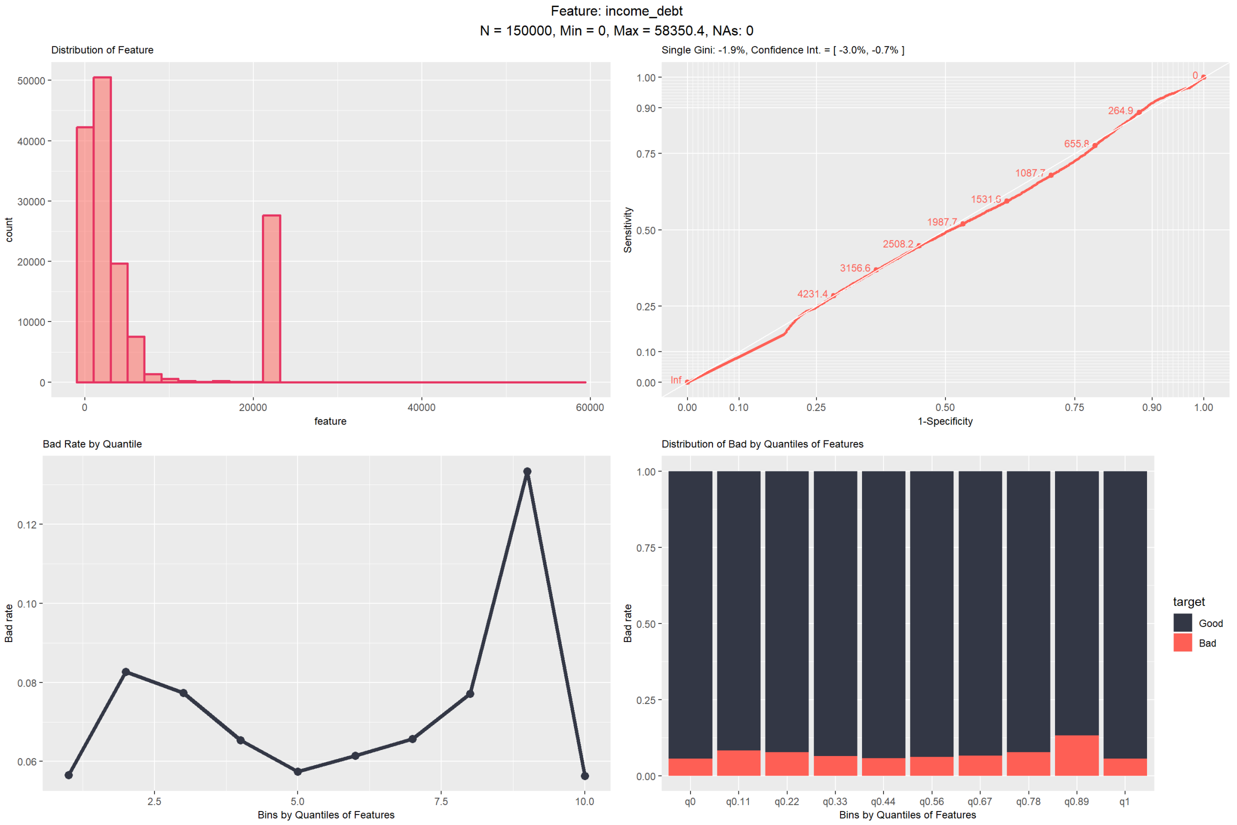
- Interaction between Monthly Income and Utilization of Credit Limits (after cleaning no `MonthlyIncome` is zero)
- Interaction between Monthly Income and Debt Ratio

```
train_set %<>%
  mutate(income_util = RevolvingUtilizationOfUnsecuredLines / MonthlyIncome,
         income_debt = MonthlyIncome*DebtRatio)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



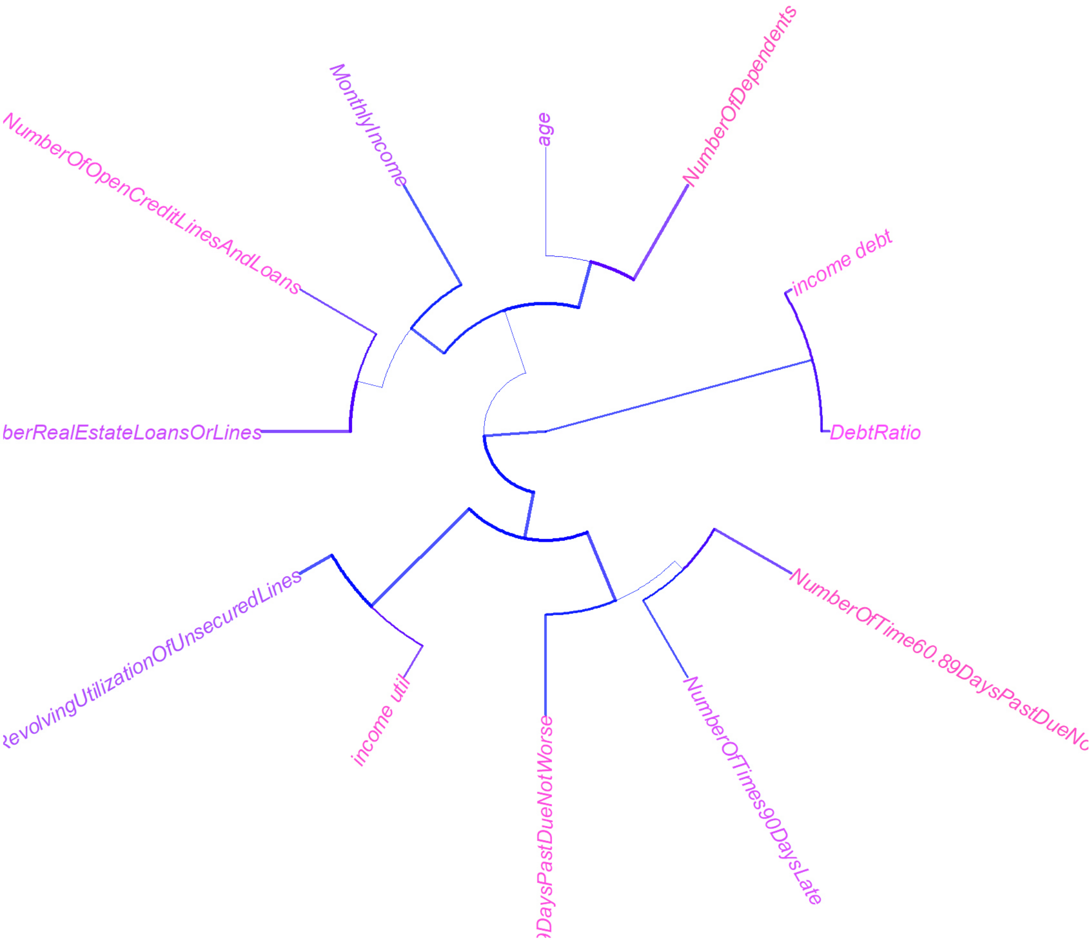
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



3. Multivariate Analysis

We do the hierarchical Clusterning of Variables to understand their interactions.

```
tree <- ClustOfVar::hclustvar(X.quanti=train_set %>%
                                select_if(is.numeric) %>% select(-SeriousDlqin2yrs),
                              X.quali=train_set %>% select_if(is.factor))
plot(as.phylo(tree), type = "fan",
     tip.color = hsv(runif(15, 0.65,  0.95), 1, 1, 0.7),
     edge.color = hsv(runif(10, 0.65, 0.75), 1, 1, 0.7),
     edge.width = runif(20,  0.5, 3), use.edge.length = TRUE, col = "gray80")
```



NumberOfOpenCreditLinesAndLoans
MonthlyIncome
age
NumberOfDependents
income debt
berRealEstateLoansOrLines
DebtRatio
RevolvingUtilizationOfUnsecuredLines
income util
DaysPastDueNotWorse
NumberOfTimes90DaysLate
NumberOfTime60.89DaysPastDueNo

- By this we see that `income_debt` and `income_util` correlated with their original variables (which is not surprising), yet the generated variables do not have better single gini than the original one. We would remove `income_debt` , but `income_util` would be kept (instead of `RevolvingUtilizationOfUnsecuredLines`) as we prefer ratio than absolute value
- `NumberRealEstateLoansOrLines` and `NumberOfOpenCreditLinesAndLoans` are correlated, which make sense that they are typically same type of information. We take the original values, sum up all loans, and take 5 bins by quantile
- While variables of late payments are correlated as the longer window of being late might involve the shorter window, we transform being late at 30+, 60+, 90+ as a binary, and sum up all count of late times. But we kept them at these steps.

```
train_set$num_loans_realestate = cs_training$NumberRealEstateLoansOrLines +
  cs_training$NumberOfOpenCreditLinesAndLoans

train_set %<>%
  mutate(num_loans_realestate = bin_data(num_loans_realestate, bins=3, binType = 'quantile')) %>%
  select(-NumberRealEstateLoansOrLines,
         -NumberOfOpenCreditLinesAndLoans,
         -RevolvingUtilizationOfUnsecuredLines,
         -income_debt)
```

3. Model Development

Use the built function to split the `train_set` by 60:40:

```
train <- data.frame()
test <- data.frame()
split_data(train_set, seed=1234, p=0.6, target='SeriousDlqin2yrs')
```

```
## Done! train and test are in your environment
## Train Set: 90000 - Percent: 0.6 - Bad Rate: 0.0661
## Test Set: 60000 - Percent: 0.4 - Bad rate: 0.06795
```

3.0. SMOTE for Unbalanced Data

The nature of data set is unbalance, with only 7% of bad rate. In this situation, the classification categories are not approximately equally represented, which migh lead to the overfitting, also some machine learning techniques are particularly sensitive to the balance of data (but not likely in our case). We deal with this situation by conducting SMOTE to adjust the class distribution of the dat set. The idea of SMOTE is taking the numeric/continuous variables in the features. The algorithm doing the oversampling by generating the synthetic data point from the vector among k-nearest neighbors. However, due to this manner of generating over-sampled data points, it might break the natural structure of data, in our case, the variables are integer.

This issue is clearer when we apply SMOTE for Recursive Partitioning.

```
## The bad ratio before SMOTE ()
train$SeriousDlqin2yrs %>% table()
```

```
## .
##     0     1
## 84051  5949
```

```
train_smote <- train %>%
  mutate(SeriousDlqin2yrs = as.factor(SeriousDlqin2yrs)) ## for SMOTE, target should be factor

train_smote <- DMwR::SMOTE(SeriousDlqin2yrs ~ ., train_smote, perc.over = 100, perc.under = 500)
```

```
## Warning in if (class(data[, col]) %in% c("factor", "character")) {: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(data[, col]) %in% c("factor", "character")) {: the
## condition has length > 1 and only the first element will be used
```

```
## Warning in `[<-.factor`(`*tmp*`, ri, value = c(1, 1, 1, 2, 2, 1, 1, 1, 2, :
## invalid factor level, NA generated

## Warning in `[<-.factor`(`*tmp*`, ri, value = c(1, 1, 1, 2, 2, 1, 1, 1, 2, :
## invalid factor level, NA generated
```

```
train_smote$SeriousDlqin2yrs %>% table()
```

```
## .
##     0     1
## 29745 11898
```

3.1. Logistic Regression

In this part, we simply process the Logistic Regression, and then stepwise for variable selection.

```
m1 <- glm(SeriousDlqin2yrs ~ .,
          data=train,
          family=binomial())
m1 <- step(m1)
```

```
## Start:  AIC=34556.88
## SeriousDlqin2yrs ~ age + NumberOfTime30.59DaysPastDueNotWorse +
##     DebtRatio + MonthlyIncome + NumberOfTimes90DaysLate + NumberOfTime60.89DaysPastDueNotWorse +
##     NumberOfDependents + income_util + num_loans_realestate
##
##                                        Df Deviance   AIC
## <none>                                       34533 34557
## - MonthlyIncome                         1    34538 34560
## - DebtRatio                             1    34542 34564
## - NumberOfDependents                    1    34555 34577
## - num_loans_realestate                  2    34625 34645
## - age                                   2    34846 34866
## - NumberOfTime60.89DaysPastDueNotWorse  1    34924 34946
## - income_util                           1    35334 35356
## - NumberOfTime30.59DaysPastDueNotWorse  1    35649 35671
## - NumberOfTimes90DaysLate               1    36023 36045
```

```
##
## Call:
## glm(formula = SeriousDlqin2yrs ~ age + NumberOfTime30.59DaysPastDueNotWorse +
##     DebtRatio + MonthlyIncome + NumberOfTimes90DaysLate + NumberOfTime60.89DaysPastDueNotWorse +
##     NumberOfDependents + income_util + num_loans_realestate,
##     family = binomial(), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3439  -0.3159  -0.2601  -0.2059   2.9390
##
## Coefficients:
##                                       Estimate Std. Error z value
## (Intercept)                         -3.780e+00  4.977e-02 -75.950
## age.L                               -5.230e-01  3.061e-02 -17.082
## age.Q                               -1.370e-01  2.703e-02  -5.066
## NumberOfTime30.59DaysPastDueNotWorse  5.469e-01  1.554e-02  35.196
## DebtRatio                            3.227e-02  1.089e-02   2.962
## MonthlyIncome                        1.381e-05  5.836e-06   2.366
## NumberOfTimes90DaysLate              9.128e-01  2.334e-02  39.104
## NumberOfTime60.89DaysPastDueNotWorse  6.224e-01  3.096e-02  20.105
## NumberOfDependents                   1.491e-01  3.199e-02   4.660
## income_util                          3.025e+03  1.032e+02  29.322
## num_loans_realestate.L               2.137e-01  2.836e-02   7.535
## num_loans_realestate.Q               1.631e-01  2.858e-02   5.707
##                                     Pr(>|z|)
## (Intercept)                          < 2e-16 ***
## age.L                                < 2e-16 ***
## age.Q                               4.06e-07 ***
## NumberOfTime30.59DaysPastDueNotWorse  < 2e-16 ***
## DebtRatio                            0.00305 **
## MonthlyIncome                        0.01797 *
## NumberOfTimes90DaysLate              < 2e-16 ***
## NumberOfTime60.89DaysPastDueNotWorse  < 2e-16 ***
## NumberOfDependents                  3.16e-06 ***
## income_util                          < 2e-16 ***
## num_loans_realestate.L              4.90e-14 ***
## num_loans_realestate.Q              1.15e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43818  on 89999  degrees of freedom
## Residual deviance: 34533  on 89988  degrees of freedom
## AIC: 34557
##
## Number of Fisher Scoring iterations: 6
```

Now, scoring on `test` set

```
test$m1_score <- predict(m1,type='response',test)
m1_pred <- ROCR::prediction(test$m1_score, test$SeriousDlqin2yrs)
m1_perf <- ROCR::performance(m1_pred,"tpr","fpr")

m1_KS <- round(max(attr(m1_perf,'y.values')[[1]]-attr(m1_perf,'x.values')[[1]])*100, 2)
m1_AUROC <- round(performance(m1_pred, measure = "auc")@y.values[[1]]*100, 2)
m1_Gini <- (2*m1_AUROC - 100)
cat("Test set: AUROC: ",m1_AUROC,"\tKS: ", m1_KS, "\tGini:", m1_Gini, "\n")
```

```
## Test set: AUROC:  84.27  KS:  54.29  Gini: 68.54
```

```
m1_score2 <- predict(m1,type='response',train)
m1_pred2 <- ROCR::prediction(m1_score2, train$SeriousDlqin2yrs)
m1_perf2 <- ROCR::performance(m1_pred2,"tpr","fpr")

m1_KS2 <- round(max(attr(m1_perf2,'y.values')[[1]]-attr(m1_perf2,'x.values')[[1]])*100, 2)
m1_AUROC2 <- round(performance(m1_pred2, measure = "auc")@y.values[[1]]*100, 2)
m1_Gini2 <- (2*m1_AUROC2 - 100)
cat("Test set: AUROC: ",m1_AUROC2,"\tKS: ", m1_KS2, "\tGini:", m1_Gini2, "\n")
```

```
## Test set: AUROC:  84.33  KS:  54.32  Gini: 68.66
```

Most of variables are strongly significant, `MonthlyIncome` is only significant at 5% and this information is already involved in `income_util` , we remove it. Also, the information of debt/loans are included in other features, we remove `DebtRatio` :

```
train2 <- train %>% select(-MonthlyIncome, -DebtRatio)
m1b <- glm(SeriousDlqin2yrs ~ .,
          data=train2,
          family=binomial())
m1b <- step(m1b)
```

```
## Start:  AIC=34565.18
## SeriousDlqin2yrs ~ age + NumberOfTime30.59DaysPastDueNotWorse +
##     NumberOfTimes90DaysLate + NumberOfTime60.89DaysPastDueNotWorse +
##     NumberOfDependents + income_util + num_loans_realestate
##
##                                       Df Deviance   AIC
## <none>                                      34545 34565
## - NumberOfDependents                   1    34566 34584
## - num_loans_realestate                 2    34648 34664
## - age                                  2    34853 34869
## - NumberOfTime60.89DaysPastDueNotWorse  1    34937 34955
## - income_util                          1    35465 35483
## - NumberOfTime30.59DaysPastDueNotWorse  1    35664 35682
## - NumberOfTimes90DaysLate              1    36039 36057
```

```
## 
## Call:
## glm(formula = SeriousDlqin2yrs ~ age + NumberOfTime30.59DaysPastDueNotWorse + 
##     NumberOfTimes90DaysLate + NumberOfTime60.89DaysPastDueNotWorse + 
##     NumberOfDependents + income_util + num_loans_realestate, 
##     family = binomial(), data = train2)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -3.3482  -0.3163  -0.2594  -0.2067   2.9065  
## 
## Coefficients:
##                                        Estimate Std. Error  z value
## (Intercept)                            -3.64981    0.02603 -140.211
## age.L                                  -0.51676    0.03056  -16.910
## age.Q                                  -0.14428    0.02693   -5.357
## NumberOfTime30.59DaysPastDueNotWorse    0.54768    0.01554   35.251
## NumberOfTimes90DaysLate                 0.91395    0.02334   39.157
## NumberOfTime60.89DaysPastDueNotWorse    0.62348    0.03096   20.136
## NumberOfDependents                      0.14206    0.03118    4.556
## income_util                          2904.02617   90.20735   32.193
## num_loans_realestate.L                  0.22574    0.02801    8.059
## num_loans_realestate.Q                  0.16534    0.02854    5.793
##                                      Pr(>|z|)    
## (Intercept)                           < 2e-16 ***
## age.L                                 < 2e-16 ***
## age.Q                                8.46e-08 ***
## NumberOfTime30.59DaysPastDueNotWorse  < 2e-16 ***
## NumberOfTimes90DaysLate               < 2e-16 ***
## NumberOfTime60.89DaysPastDueNotWorse  < 2e-16 ***
## NumberOfDependents                   5.21e-06 ***
## income_util                           < 2e-16 ***
## num_loans_realestate.L               7.67e-16 ***
## num_loans_realestate.Q               6.92e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 43818  on 89999  degrees of freedom
## Residual deviance: 34545  on 89990  degrees of freedom
## AIC: 34565
## 
## Number of Fisher Scoring iterations: 6
```

```
## Train set - AUROC:  84.31    KS:  54.03  Gini: 68.62
```

```
## Test set - AUROC:  84.25     KS:  53.95  Gini: 68.5
```

Now, we run the logistic on SMOTE train set.

```
train_smote <- train_smote %>% select(-MonthlyIncome, -DebtRatio)
m1c <- glm(SeriousDlqin2yrs ~ .,
          data=train_smote,
          family=binomial())
m1c <- step(m1b)
```

```
## Start:  AIC=34565.18
## SeriousDlqin2yrs ~ age + NumberOfTime30.59DaysPastDueNotWorse + 
##     NumberOfTimes90DaysLate + NumberOfTime60.89DaysPastDueNotWorse + 
##     NumberOfDependents + income_util + num_loans_realestate
## 
##                                        Df Deviance   AIC
## <none>                                       34545 34565
## - NumberOfDependents                    1    34566 34584
## - num_loans_realestate                  2    34648 34664
## - age                                   2    34853 34869
## - NumberOfTime60.89DaysPastDueNotWorse  1    34937 34955
## - income_util                           1    35465 35483
## - NumberOfTime30.59DaysPastDueNotWorse  1    35664 35682
## - NumberOfTimes90DaysLate               1    36039 36057
```

```
##
## Call:
## glm(formula = SeriousDlqin2yrs ~ age + NumberOfTime30.59DaysPastDueNotWorse +
##     NumberOfTimes90DaysLate + NumberOfTime60.89DaysPastDueNotWorse +
##     NumberOfDependents + income_util + num_loans_realestate,
##     family = binomial(), data = train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3482  -0.3163  -0.2594  -0.2067   2.9065
##
## Coefficients:
##                                        Estimate Std. Error  z value
## (Intercept)                            -3.64981    0.02603 -140.211
## age.L                                  -0.51676    0.03056  -16.910
## age.Q                                  -0.14428    0.02693   -5.357
## NumberOfTime30.59DaysPastDueNotWorse    0.54768    0.01554   35.251
## NumberOfTimes90DaysLate                 0.91395    0.02334   39.157
## NumberOfTime60.89DaysPastDueNotWorse    0.62348    0.03096   20.136
## NumberOfDependents                      0.14206    0.03118    4.556
## income_util                          2904.02617   90.20735   32.193
## num_loans_realestate.L                  0.22574    0.02801    8.059
## num_loans_realestate.Q                  0.16534    0.02854    5.793
##                                      Pr(>|z|)
## (Intercept)                           < 2e-16 ***
## age.L                                 < 2e-16 ***
## age.Q                                8.46e-08 ***
## NumberOfTime30.59DaysPastDueNotWorse  < 2e-16 ***
## NumberOfTimes90DaysLate               < 2e-16 ***
## NumberOfTime60.89DaysPastDueNotWorse  < 2e-16 ***
## NumberOfDependents                   5.21e-06 ***
## income_util                           < 2e-16 ***
## num_loans_realestate.L               7.67e-16 ***
## num_loans_realestate.Q               6.92e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43818  on 89999  degrees of freedom
## Residual deviance: 34545  on 89990  degrees of freedom
## AIC: 34565
##
## Number of Fisher Scoring iterations: 6
```

```
## Train set - AUROC:  84.27    KS:  53.82  Gini: 68.54
```

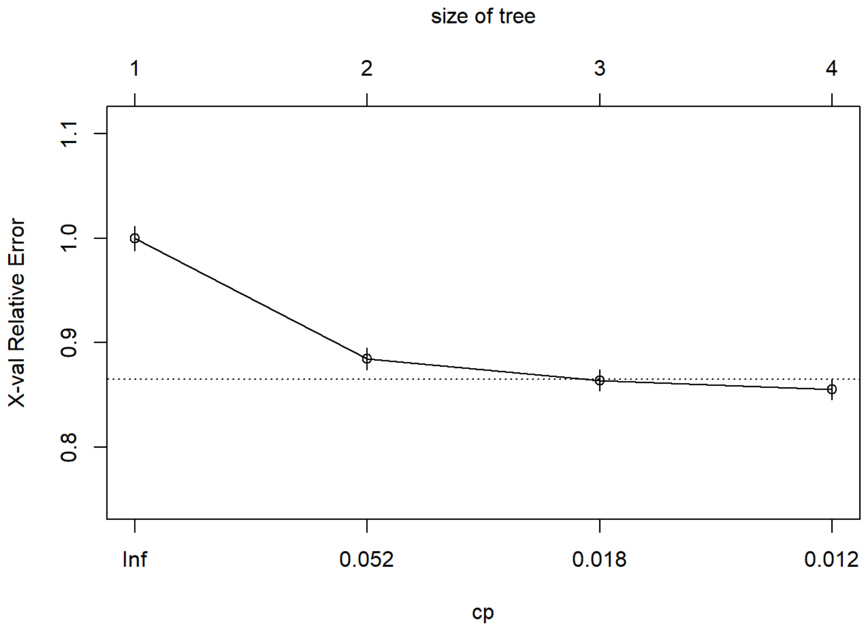
```
## Test set - AUROC:  84.25     KS:  53.95  Gini: 68.5
```

3.2. Recursive Partitioning

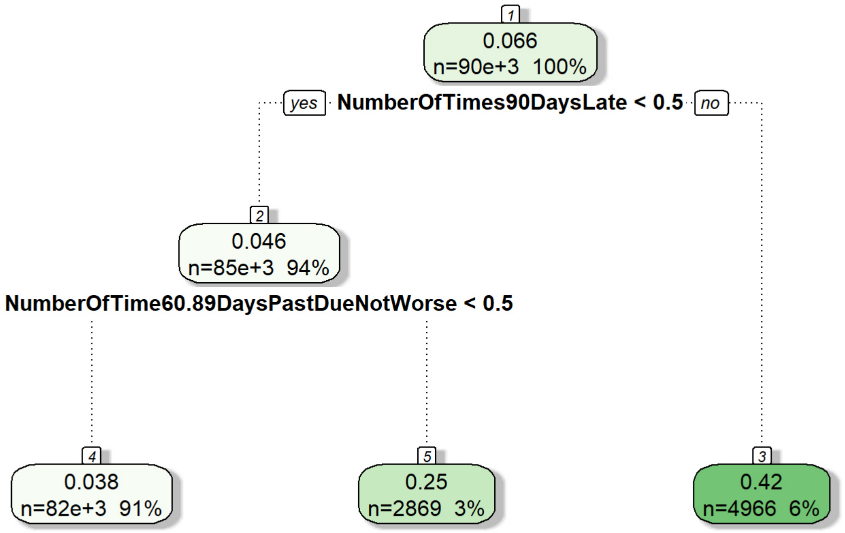
Recursive Paritioning is a tree-based algorithm to create a decision tree, which segments the population into sub-populations. The split is by a set of predictors. With our dataset, we use the regression tree for numeric variables.

The overfitting is avoided by pruning the tree. We do that by looking into the cross-validation relative error `X-val Relative Error` and select the `cp` value at the minimum error considering the size of the tree.

```
m2 <- rpart(SeriousDlqin2yrs~.,data=train2)
# Print tree detail
plotcp(m2)
```



```
## Prunning the tree
ctrl <- rpart::rpart.control(cp = 0.018)
m2 <- rpart(SeriousDlqin2yrs~.,data=train2, control = ctrl)
rattle::fancyRpartPlot(m2)
```



Rattle 2019-Mar-13 16:37:46 anh.dang

```
test$m2_score <- predict(m2,type='vector',test)
m2_pred <- prediction(test$m2_score, test$SeriousDlqin2yrs)
m2_perf <- performance(m2_pred,"tpr","fpr")

m2_KS <- round(max(attr(m2_perf,'y.values')[[1]]-attr(m2_perf,'x.values')[[1]])*100, 2)
m2_AUROC <- round(performance(m2_pred, measure = "auc")@y.values[[1]]*100, 2)
m2_Gini <- (2*m2_AUROC - 100)

m2_score2 <- predict(m2,type='vector',train2)
m2_pred2 <- prediction(m2_score2, train2$SeriousDlqin2yrs)
m2_perf2 <- performance(m2_pred2,"tpr","fpr")

m2_KS2 <- round(max(attr(m2_perf2,'y.values')[[1]]-attr(m2_perf2,'x.values')[[1]])*100, 2)
m2_AUROC2 <- round(performance(m2_pred2, measure = "auc")@y.values[[1]]*100, 2)
m2_Gini2 <- (2*m2_AUROC2 - 100)

cat("Train set - AUROC: ",m2_AUROC2,"\tKS: ", m2_KS2, "\tGini:", m2_Gini2, "\n")
```

```
## Train set - AUROC:  70.73     KS:  41     Gini: 41.46
```

```
cat("Test set - AUROC: ",m2_AUROC,"\tKS: ", m2_KS, "\tGini:", m2_Gini, "\n")
```

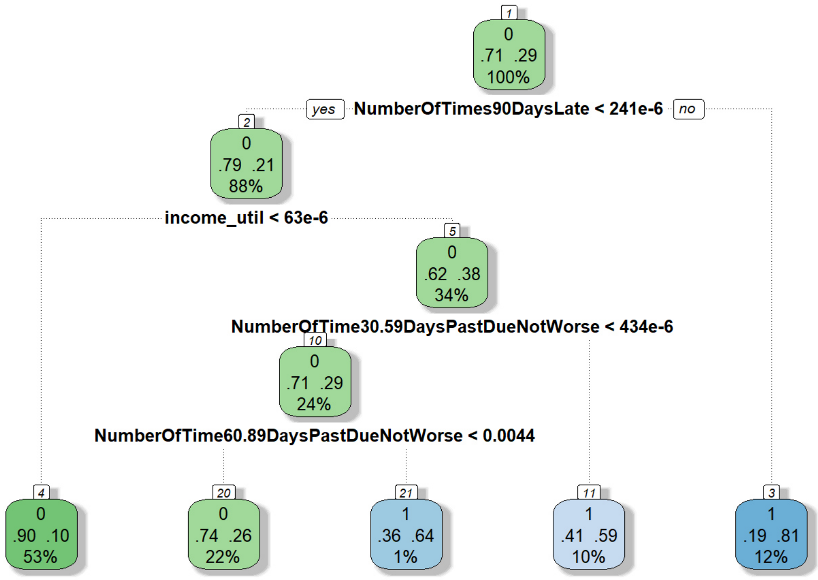
```
## Test set - AUROC:  70.52     KS:  40.58  Gini: 41.04
```

As mentioned above, SMOTE does over-sampling by creating the synthetic data point from k-nearest neighbor. There, we see the tree split the population at some ‘weird’ cut-off. Considering that our original variable taking integer values, the cut-off is not integer, such as 0.0044 of `NumberOfTime60.89DaysPastDueNotWorse` . In fact, we could consider it to be zero.

```
m2_smote <- rpart(SeriousDlqin2yrs~.,data=train_smote)
# Print tree detail
printcp(m2_smote)
```

```
## 
## Classification tree:
## rpart(formula = SeriousDlqin2yrs ~ ., data = train_smote)
## 
## Variables actually used in tree construction:
## [1] income_util
## [2] NumberOfTime30.59DaysPastDueNotWorse
## [3] NumberOfTime60.89DaysPastDueNotWorse
## [4] NumberOfTimes90DaysLate
## 
## Root node error: 11898/41643 = 0.28571
## 
## n= 41643
## 
##          CP nsplit rel error  xerror      xstd
## 1 0.269289      0   1.00000 1.00000 0.0077482
## 2 0.033871      1   0.73071 0.73080 0.0069711
## 3 0.013700      3   0.66297 0.66415 0.0067252
## 4 0.010000      4   0.64927 0.65045 0.0066715
```

```
rattle::fancyRpartPlot(m2_smote)
```



Rattle 2019-Mar-13 16:37:47 anh.dang

```
## Train set - AUROC:  74.97     KS:  49.94  Gini: 49.94
```

```
## Test set - AUROC:  74.55     KS:  49.1    Gini: 49.1
```

3.3. Unbiased Non-parametric Model-Based (Logistic)

Model-based Recurive Partitioning (MOB) is the combination of Logistic Regression with the tree-based algorithms. Thus, it combines the advantages and augmented the disadvantages of both methods. Generally speaking, MOB combines two main layers: (1) the recursive partitioning by the features (could be categorical or binned numeric variables), then within each terminal node (2), it conduct the regression (logistic) by the determined features within each node.

The algorithm follows:

1. Fitting the model on regressors (i.e. by logistic), using the sample at the current node
2. Assess the stability of the model parameters with resoect to the parition by the paritioning variables. Once it is instable, split at the partitioning variable with smallest p-val.
3. Then searching the cut-off to split within the specific partitionaing variable (selected at Step 2)

Conceptual Meanings:

- The tree-based layer augments the disadvantage of Logistics (and GLM in general) favouring linearity and mono-tone relationsip in predictions. The tree-based layer helps to capture the non-linearity and interaction between predictors
- The tree-based layers is effective to deal with categorical variables, which ease the burden of GLM to do the variable selection among levels of categorical variables.
- The bottom-layer of Logistics would still capture some strong numeric, linear, monotone variables (such as: `RevolvingUtilizationOfUnsecuredLines`)

Practical Meanings:

- MOB could be used for segmentations (such as by industries, ages of business, levels of revenue, etc.) in credit scoring modelling. Within each segment, the Logistic, which is popular in the industry could be applied as normally
- The tendency to rely on the stability of model parameters aligns with common framework in credit scoring model validation
- The visualisation of MOB, if not being applied for the main model development, is still be beneficial to be used for model validation to test the stability and changes of slopes across different segmetns. Also, it is useful to exploring the data to advice whether we need the segmentation in model development.

3.3.1. Model-Based Recursive Partitioning (Feature Engineering)

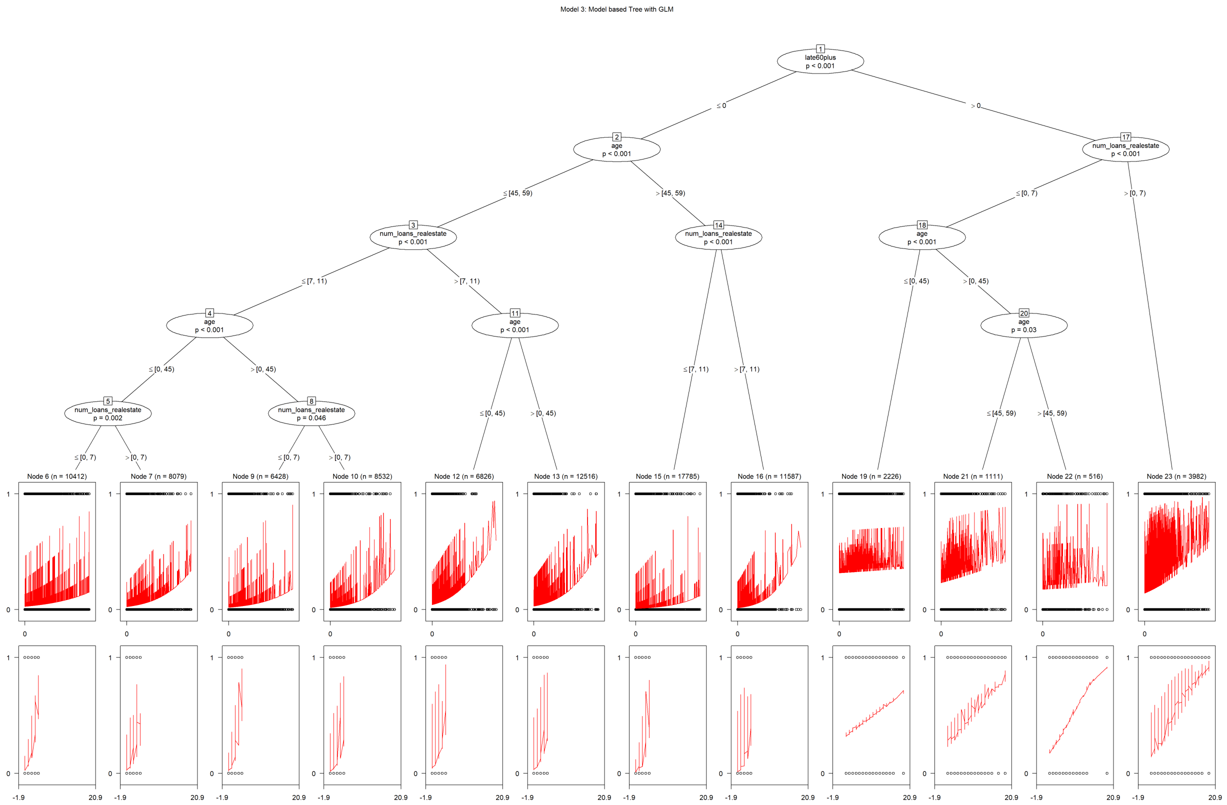
In this part, we do some feature engineering to combine the count of late payments in 30+, 60+, and 90+ by arbitrarily assigning weight for them. Intuitively, being late at 90+ would give stronger signals about the risk of the observation, we assign 3 for 90+, then 2, and 1 for 60+ and 30+ (this is totally arbitrarily, later we would try some more sophisticated weighting method). Also, we keep a flag of ever being late 60+ or 90+, to reserve a part of discrimination.

When running mob, we put `income_util` and `late_count` as regression of the logistic layer, and the other binned variables as partitioning features of the tree-based layers. One could imagine it as the process of segmenting by being late, age, and number of loans, then within each segment, we run the logit model on two variables.

As the task of model validation, we could look into the graph to see if the stability of slopes in each features. * Here, obviously, we see the slope of `income_util` changes across segments, while the the slope is steep (it means the feature is more predictive) in the segment without ever-being late 60+ and 90+, young age, and have fewer number of loans. This factor is not discriminative in the riskier population (being late 60 and 90+). Also, even within the same branch, this utilization over income seems to be more predictive in older people. We can guess that once the older people over-use their credit limits, it is a very strong signal of being 'bad', while the over-using of credit limit seems to be more popular (then less discriminative) among young people. * Similar, the number of times being late is more preditive among the group of 'better' people, while it is not predictive in the group that everyone are 'bad' at the quite similar level.

Similar to other algorithm, MOB could be tuned by `mob_control()` , by doing some grid search. Yet, considering the limited number of predictors, and the data is quite simple, no tuning exercise is taken.

```r
train <- train %>%
  mutate(late_count = NumberOfTime30.59DaysPastDueNotWorse*1 +
           NumberOfTime60.89DaysPastDueNotWorse*2 +
           NumberOfTimes90DaysLate*3,
         late60plus = ifelse(NumberOfTime60.89DaysPastDueNotWorse > 0 |
                               NumberOfTimes90DaysLate > 0, 1, 0))

m3 <- party::mob(SeriousDlqin2yrs ~ income_util + late_count |
                   num_loans_realestate +
                   late60plus +
                   #NumberOfTime30.59DaysPastDueNotWorse +
                   #NumberOfDependents +
                   age,
                 #control = mob_control(maxdepth = 4),
                 data=train,
                 family=binomial())

plot(m3, main="Model 3: Model based Tree with GLM")
```



```
## Train set - AUROC:  84.69    KS:  54.36  Gini: 69.38
```

```
## Test set - AUROC:  84.59     KS:  54.23  Gini: 69.18
```

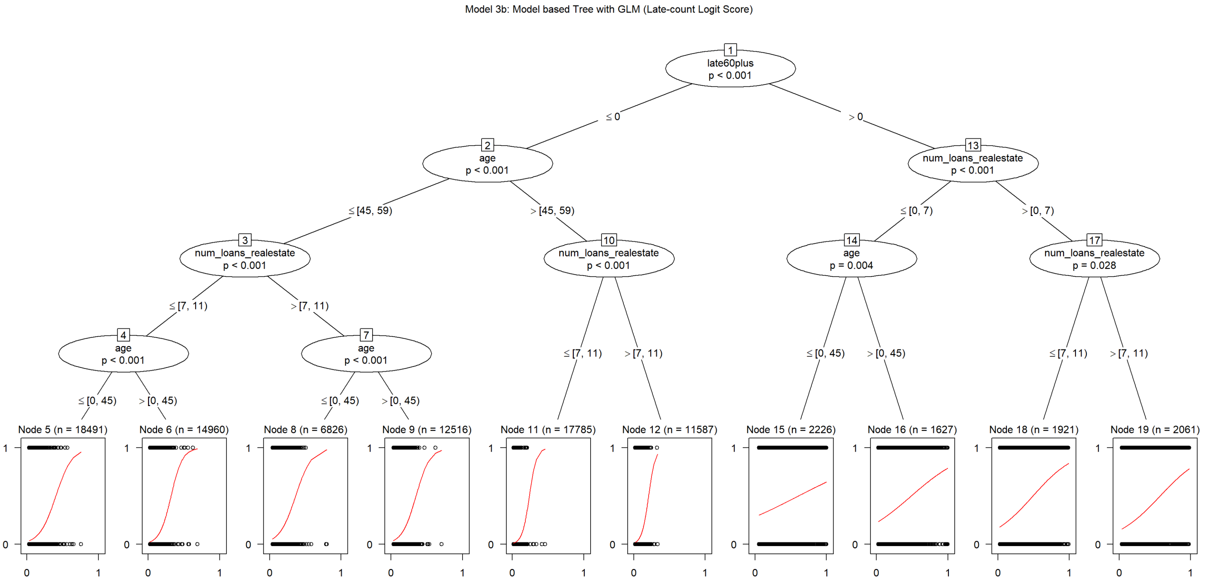
3.3.2. Model-Based Recursive Partitioning (Logit Score)

As the previous promise, now we decide the weights of counts of being lates by doing the logistic regression of target variable on these variables. For more simple version, we also combine `income_util` into the logistic score. Same as the previous part, we see that the slope (model parameter of `logit_late`) changes acoss the segment.

```
logit_late <- glm(SeriousDlqin2yrs ~
                    NumberOfTime30.59DaysPastDueNotWorse +
                    NumberOfTime60.89DaysPastDueNotWorse +
                    NumberOfTimes90DaysLate +
                    income_util,
                  data = train,
                  family = binomial)
summary(logit_late)
```

```
##
## Call:
## glm(formula = SeriousDlqin2yrs ~ NumberOfTime30.59DaysPastDueNotWorse +
##     NumberOfTime60.89DaysPastDueNotWorse + NumberOfTimes90DaysLate +
##     income_util, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3695  -0.3013  -0.2489  -0.2408   2.6703
##
## Coefficients:
##                                        Estimate Std. Error z value
## (Intercept)                            -3.53649    0.02118 -166.96
## NumberOfTime30.59DaysPastDueNotWorse    0.57824    0.01523   37.98
## NumberOfTime60.89DaysPastDueNotWorse    0.63158    0.03090   20.44
## NumberOfTimes90DaysLate                 0.90403    0.02282   39.62
## income_util                          2977.28210   81.39777   36.58
##                                      Pr(>|z|)
## (Intercept)                            <2e-16 ***
## NumberOfTime30.59DaysPastDueNotWorse   <2e-16 ***
## NumberOfTime60.89DaysPastDueNotWorse   <2e-16 ***
## NumberOfTimes90DaysLate                <2e-16 ***
## income_util                            <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43818  on 89999  degrees of freedom
## Residual deviance: 35014  on 89995  degrees of freedom
## AIC: 35024
##
## Number of Fisher Scoring iterations: 6
```

```
train$late_count_logit = predict(m1b, train, type = 'response')
train <- train %>%
  mutate(late60plus = ifelse(NumberOfTime60.89DaysPastDueNotWorse > 0 |
                               NumberOfTimes90DaysLate > 0, 1, 0))

m3b <- party::mob(SeriousDlqin2yrs ~ late_count_logit |
                   num_loans_realestate +
                   late60plus +
                   #NumberOfTime30.59DaysPastDueNotWorse +
                   #NumberOfDependents +
                   age,
          #control = mob_control(maxdepth = 3, alpha = 0.05),
          data=train,
          family=binomial())

plot(m3b, main="Model 3b: Model based Tree with GLM (Late-count Logit Score)")
```



```
## Train set - AUROC:  83.21    KS:  51.85  Gini: 66.42
```

```
## Test set - AUROC:  83    KS:  51.77  Gini: 66
```

3.4. Chi-square Automatic Interaction Detector (CHAID)

As CHAID only take the caterogical variables, to conduct the chi-squared among features. It is more analytical-oriented, as it is a tool used ot discover the relationship between categorical variables (hence, all variables are converted to categorical for CHAID algorithm).

With the same manner as CART, CHAID figures out the way to merge and split among different categorical variables, and within different levels of one categorical variables to segment the population into different subset, corresponding to the target variables.

Different from CART/Regression Tree, CHAID splits the population by features, depending on the statistical significance of chi-square of independency among two categorical features/two levels of one categorical feature. That makes CHAID to be more favoured for analytics, comparing to other predictive tree-based. Also, different from CART, CHAID is able to split non-binary at each node.

In this part, we would illustrate the process of tuning a tree model by grid search (which are similar to other machine learning techniques, within and out of this work). Yet, depending on the business sense and purpose of analytics/predictics, we can also determine the desired level of complexity of the model. For CHAID, we would compare grid-search tunned vs. arbitrary parameter controls.

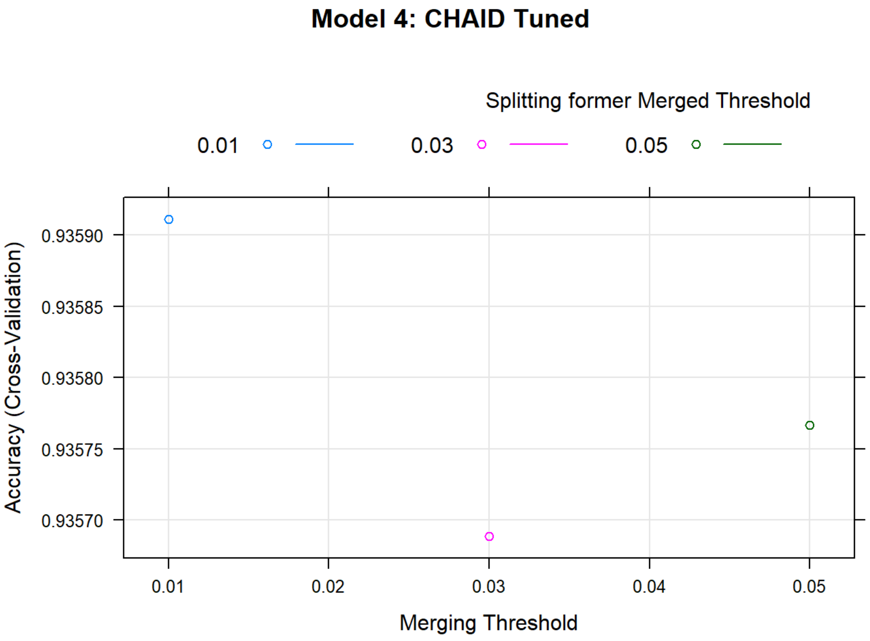
```
y = c(train$SeriousDlqin2yrs, test$SeriousDlqin2yrs)
traintest_chaid <- rbind(train, test[,names(train)]) %>%
  mutate_if(function(v) length(unique(v)) < 10, as.factor) %>%
  mutate_if(is.numeric, funs(as.factor(as.numeric((bin_data(., bins = 5, binType = 'quantile'))))))

traintest_chaid$SeriousDlqin2yrs = as.factor(y)

train_chaid = traintest_chaid[1:nrow(train),]
test_chaid = traintest_chaid[(nrow(train)+1):nrow(traintest_chaid),]

summary(train_chaid)
```

```
##  SeriousDlqin2yrs        age        NumberOfTime30.59DaysPastDueNotWorse
##  0:84051          [0, 45)  :28834   0:75618                             
##  1: 5949          [45, 59) :30271   1: 9653                             
##                   [59, 109]:30895   2: 2756                             
##                                     3: 1043                             
##                                     4:  930                             
##  DebtRatio MonthlyIncome NumberOfTimes90DaysLate
##  1:18042   1:17917       0:85034                
##  2:17934   2:18169       1: 3154                
##  3:18048   3:35968       2:  912                
##  4:35976   4:17946       3:  900                
##                                                 
##  NumberOfTime60.89DaysPastDueNotWorse NumberOfDependents income_util
##  0:85490                              0:54485            1:18063    
##  1: 3412                              1:35515            2:17918    
##  2:  671                                                 3:18055    
##  3:  427                                                 4:18053    
##                                                          5:17911    
##  num_loans_realestate late_count late60plus late_count_logit
##  [0, 7)   :29890      1:71791    0:82165    1:17893         
##  [7, 11)  :27120      2:18209    1: 7835    2:17707         
##  [11, 112]:32990                            3:18421         
##                                             4:17967         
##                                             5:18012         
```

```
train_control <- trainControl(method = "cv",
                              number = 5,
                              verboseIter = TRUE,
                              savePredictions = "final")

chaid.m1 <- train(
   x = train_chaid %>% select(-SeriousDlqin2yrs),
   y = train_chaid$SeriousDlqin2yrs,
   method = "chaid",
   metric = "Accuracy",
   trControl = train_control
 )
```

```
## + Fold1: alpha2=0.05, alpha3=-1, alpha4=0.05
## - Fold1: alpha2=0.05, alpha3=-1, alpha4=0.05
## + Fold1: alpha2=0.03, alpha3=-1, alpha4=0.03
## - Fold1: alpha2=0.03, alpha3=-1, alpha4=0.03
## + Fold1: alpha2=0.01, alpha3=-1, alpha4=0.01
## - Fold1: alpha2=0.01, alpha3=-1, alpha4=0.01
## + Fold2: alpha2=0.05, alpha3=-1, alpha4=0.05
## - Fold2: alpha2=0.05, alpha3=-1, alpha4=0.05
## + Fold2: alpha2=0.03, alpha3=-1, alpha4=0.03
## - Fold2: alpha2=0.03, alpha3=-1, alpha4=0.03
## + Fold2: alpha2=0.01, alpha3=-1, alpha4=0.01
## - Fold2: alpha2=0.01, alpha3=-1, alpha4=0.01
## + Fold3: alpha2=0.05, alpha3=-1, alpha4=0.05
## - Fold3: alpha2=0.05, alpha3=-1, alpha4=0.05
## + Fold3: alpha2=0.03, alpha3=-1, alpha4=0.03
## - Fold3: alpha2=0.03, alpha3=-1, alpha4=0.03
## + Fold3: alpha2=0.01, alpha3=-1, alpha4=0.01
## - Fold3: alpha2=0.01, alpha3=-1, alpha4=0.01
## + Fold4: alpha2=0.05, alpha3=-1, alpha4=0.05
## - Fold4: alpha2=0.05, alpha3=-1, alpha4=0.05
## + Fold4: alpha2=0.03, alpha3=-1, alpha4=0.03
## - Fold4: alpha2=0.03, alpha3=-1, alpha4=0.03
## + Fold4: alpha2=0.01, alpha3=-1, alpha4=0.01
## - Fold4: alpha2=0.01, alpha3=-1, alpha4=0.01
## + Fold5: alpha2=0.05, alpha3=-1, alpha4=0.05
## - Fold5: alpha2=0.05, alpha3=-1, alpha4=0.05
## + Fold5: alpha2=0.03, alpha3=-1, alpha4=0.03
## - Fold5: alpha2=0.03, alpha3=-1, alpha4=0.03
## + Fold5: alpha2=0.01, alpha3=-1, alpha4=0.01
## - Fold5: alpha2=0.01, alpha3=-1, alpha4=0.01
## Aggregating results
## Selecting tuning parameters
## Fitting alpha2 = 0.01, alpha3 = -1, alpha4 = 0.01 on full training set
```

```
chaid_graph <- function(model, title){

  plot(
    model,
    main = title, ## title
    gp = grid::gpar(
      col = "blue",
      lty = "solid",
      lwd = 3,
      fontsize = 8
    )
  )
}

chaid_graph(chaid.m1, title = 'Model 4: CHAID Tuned')
```



```
test$m4_score <- predict(chaid.m1,type='prob',test_chaid)
m4_pred <- prediction(test$m4_score[,2], test$SeriousDlqin2yrs)
m4_perf <- performance(m4_pred,"tpr","fpr")

m4_KS <- round(max(attr(m4_perf,'y.values')[[1]]-attr(m4_perf,'x.values')[[1]])*100, 2)
m4_AUROC <- round(performance(m4_pred, measure = "auc")@y.values[[1]]*100, 2)
m4_Gini <- (2*m4_AUROC - 100)

m4_score2 <- predict(chaid.m1, type='prob',train_chaid)
m4_pred2 <- prediction(m4_score2[,2], train$SeriousDlqin2yrs)
m4_perf2 <- performance(m4_pred2,"tpr","fpr")

m4_KS2 <- round(max(attr(m4_perf2,'y.values')[[1]]-attr(m4_perf2,'x.values')[[1]])*100, 2)
m4_AUROC2 <- round(performance(m4_pred2, measure = "auc")@y.values[[1]]*100, 2)
m4_Gini2 <- (2*m4_AUROC2 - 100)

cat("Train set - AUROC: ",m4_AUROC2,"\tKS: ", m4_KS2, "\tGini:", m4_Gini2, "\n")
```

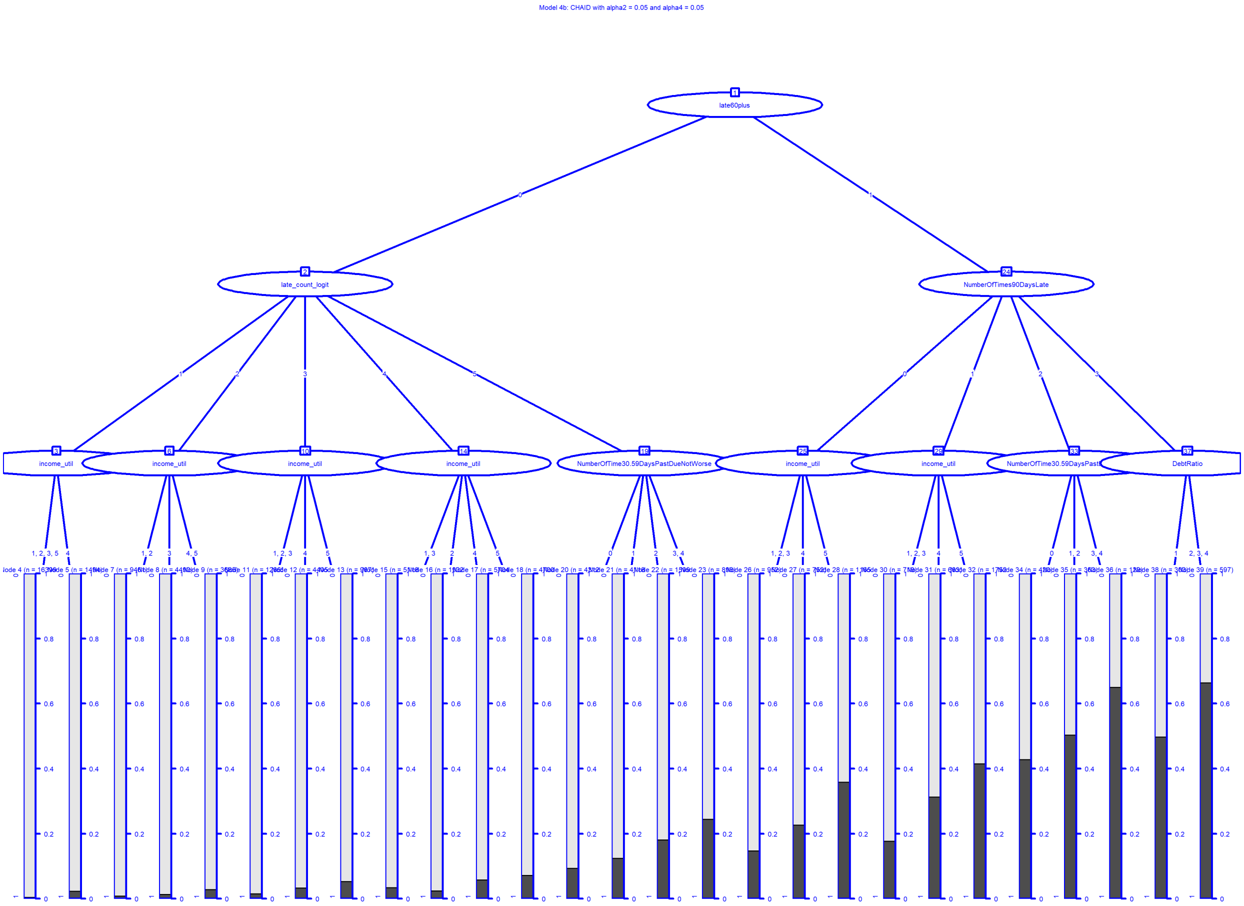
```
## Train set - AUROC:  85.79    KS:  55.49  Gini: 71.58
```

```
cat("Test set - AUROC: ",m4_AUROC,"\tKS: ", m4_KS, "\tGini:", m4_Gini, "\n")
```

```
## Test set - AUROC:  85.13     KS:  55.5   Gini: 70.26
```

```
# install.packages("CHAID", repos="http://R-Forge.R-project.org")
ctrl.alpha <- CHAID::chaid_control(alpha2 = 0.05, alpha4 = 0.05, maxheight = 3)
model4 <- CHAID::chaid(SeriousDlqin2yrs ~ ., data = train_chaid, control = ctrl.alpha)
```

```
chaid_graph(model4, title = 'Model 4b: CHAID with alpha2 = 0.05 and alpha4 = 0.05')
```



```
test$m4b_score <- predict(model4,type='prob',test_chaid)
m4b_pred <- prediction(test$m4b_score[,2], test$SeriousDlqin2yrs)
m4b_perf <- performance(m4b_pred,"tpr","fpr")

m4b_KS <- round(max(attr(m4b_perf,'y.values')[[1]]-attr(m4b_perf,'x.values')[[1]])*100, 2)
m4b_AUROC <- round(performance(m4b_pred, measure = "auc")@y.values[[1]]*100, 2)
m4b_Gini <- (2*m4b_AUROC - 100)

m4b_score2 <- predict(model4, type='prob',train_chaid)
m4b_pred2 <- prediction(m4b_score2[,2], train$SeriousDlqin2yrs)
m4b_perf2 <- performance(m4b_pred2,"tpr","fpr")

m4b_KS2 <- round(max(attr(m4b_perf2,'y.values')[[1]]-attr(m4b_perf2,'x.values')[[1]])*100, 2)
m4b_AUROC2 <- round(performance(m4b_pred2, measure = "auc")@y.values[[1]]*100, 2)
m4b_Gini2 <- (2*m4b_AUROC2 - 100)

cat("Train set - AUROC: ",m4b_AUROC2,"\tKS: ", m4b_KS2, "\tGini:", m4b_Gini2, "\n")
```

```
## Train set - AUROC:  85.01    KS:  54.51  Gini: 70.02
```

```
cat("Test set - AUROC: ",m4b_AUROC,"\tKS: ", m4b_KS, "\tGini:", m4b_Gini, "\n")
```

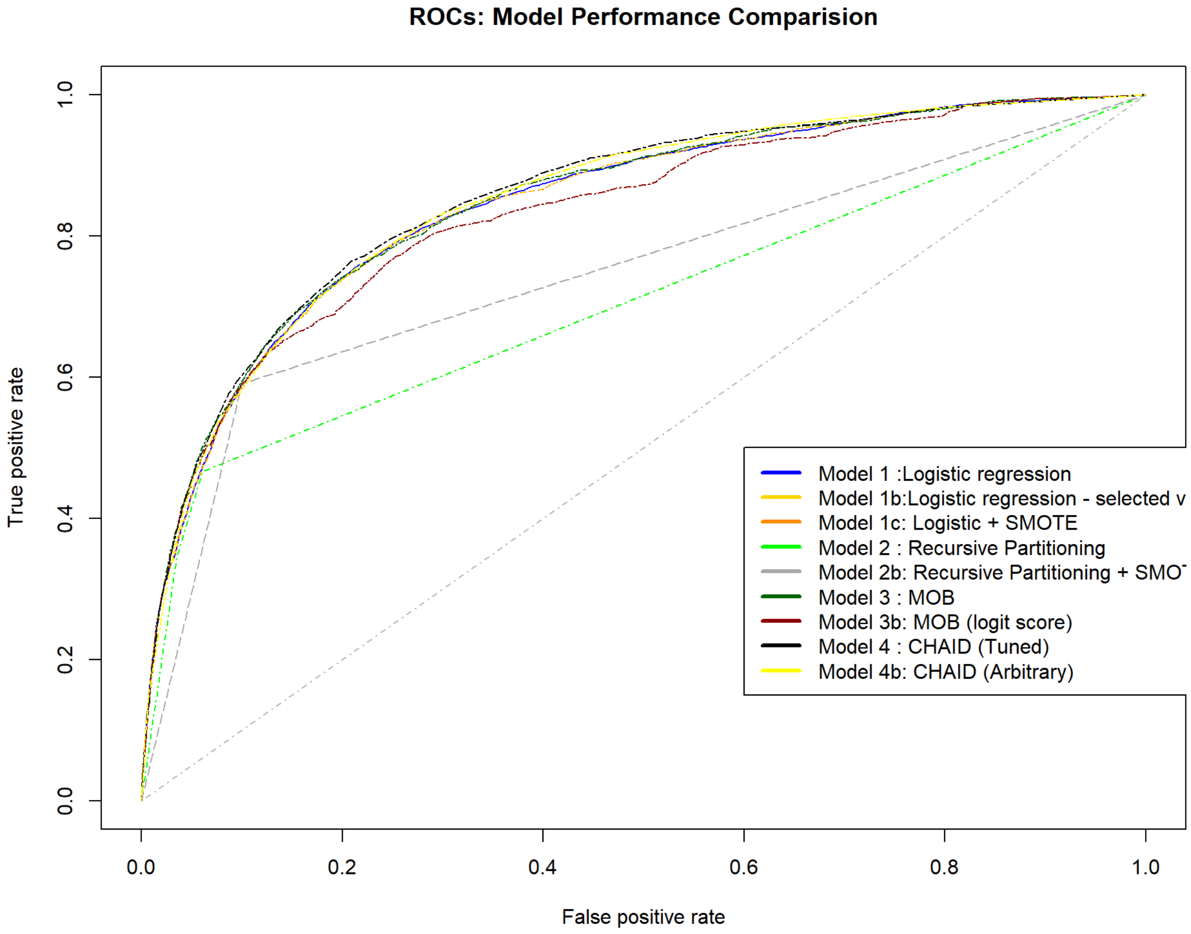
```
## Test set - AUROC:  84.75     KS:  54.25  Gini: 69.5
```

4. Model Performance

Different algorithms in this work are compared by ROC and Gini. However, please keep in mind that the best method in this exercise not necessary be the superior for other situations. The performance of methods heavily depends on the status of data and the purpose of modellers.

4.1. ROC Curve

```
plot(m1_perf, col='blue', lty=1, main='ROCs: Model Performance Comparision') # Logistic regression
plot(m1b_perf, col='gold',lty=2, add=TRUE); # Logit selected var
plot(m1c_perf, col='dark orange',lty=3, add=TRUE); # Logit + SMOTE
plot(m2_perf, col='green',add=TRUE,lty=4); # tree
plot(m2b_perf, col='dark gray',add=TRUE,lty=5); # Tree SMOTE
plot(m3_perf, col='dark green',add=TRUE,lty=6); # MOB
plot(m3b_perf, col='dark red',add=TRUE,lty=6); # MOB (logit score)
plot(m4_perf, col='black',add=TRUE,lty=6, size = 1); # CHAID Tuned
plot(m4b_perf, col='yellow',add=TRUE,lty=6); # CHAID Arbitrate
    legend(0.6,0.5,
           c('Model 1 :Logistic regression',
             'Model 1b:Logistic regression - selected vars',
             'Model 1c: Logistic + SMOTE',
             'Model 2 : Recursive Partitioning',
             'Model 2b: Recursive Partitioning + SMOTE',
             'Model 3 : MOB',
             'Model 3b: MOB (logit score)',
             'Model 4 : CHAID (Tuned)',
             'Model 4b: CHAID (Arbitrary)'),
           col=c('blue','gold', 'dark orange','green', 'dark gray', 'dark green','dark red',
                 'black','yellow'),
           lwd=3);
lines(c(0,1),c(0,1),col = "gray", lty = 4 ) # random line
```


ROCs: Model Performance Comparision
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Model 1 :Logistic regression
Model 1b:Logistic regression - selected v
Model 1c: Logistic + SMOTE
Model 2 : Recursive Partitioning
Model 2b: Recursive Partitioning + SMO
Model 3 : MOB
Model 3b: MOB (logit score)
Model 4 : CHAID (Tuned)
Model 4b: CHAID (Arbitrary)

4.2. Table of Performance Metrics

```
# Performance Table
models <- c('Model 1 :Logistic regression',
            'Model 1b:Logistic regression - selected vars',
            'Model 1c: Logistic + SMOTE',
            'Model 2 : Recursive Partitioning',
            'Model 2b: Recursive Partitioning + SMOTE',
            'Model 3 : MOB',
            'Model 3b: MOB (logit score)',
            'Model 4 : CHAID Tuned',
            'Model 4b : CHAID Arbitrary')

# Gini_train
models_Gini_train <- c(m1_Gini2, m1b_Gini2, m1c_Gini2,
                 m2_Gini2, m2b_Gini2,
                 m3_Gini2, m3b_Gini2,
                 m4_Gini2, m4b_Gini2)
# Gini_test
models_Gini_test <- c(m1_Gini, m1b_Gini, m1c_Gini,
                 m2_Gini, m2b_Gini,
                 m3_Gini, m3b_Gini,
                 m4_Gini, m4b_Gini)

# gini change
modesl_Gini_ch <- (((models_Gini_test - models_Gini_train)/models_Gini_train)*100) %>% round(2)

# Combine AUC and KS
model_performance_metric <- as.data.frame(cbind(models, models_Gini_train,
                                                models_Gini_test,
                                                modesl_Gini_ch))

# Colnames
colnames(model_performance_metric) <- c("Model", "Gini Train", "Gini Test", "%Gini change")

# Display Performance Reports
kable(model_performance_metric, caption ="Comparision of Model Performances") %>%
  kable_styling()
```

Comparision of Model Performances

Model	Gini Train	Gini Test	%Gini change
Model 1 :Logistic regression	68.66	68.54	-0.17
Model 1b:Logistic regression - selected vars	68.62	68.5	-0.17
Model 1c: Logistic + SMOTE	68.54	68.5	-0.06
Model 2 : Recursive Partitioning	41.46	41.04	-1.01
Model 2b: Recursive Partitioning + SMOTE	49.94	49.1	-1.68
Model 3 : MOB	69.38	69.18	-0.29
Model 3b: MOB (logit score)	66.42	66	-0.63
Model 4 : CHAID Tuned	71.58	70.26	-1.84
Model 4b : CHAID Arbitrary	70.02	69.5	-0.74

5. References

1. *Statistics Solutions*, CHAID: Non-parametric Analysis, https://www.statisticssolutions.com/non-parametric-analysis-chaid/) (https://www.statisticssolutions.com/non-parametric-analysis-chaid/))
2. *Chuck Powell* CHAID and CARET: A Good Combo (June 6, 2018) https://www.r-bloggers.com/chaid-and-caret-a-good-combo-june-6-2018/ (https://www.r-bloggers.com/chaid-and-caret-a-good-combo-june-6-2018/)
3. *Ariful Mondal* Classifications in R: Response Modeling/Credit Scoring/Credit Rating using Machine Learning Techniques https://rstudio-pubs-static.s3.amazonaws.com/225209_df0130c5a0614790b6365676b9372c07.html#32_recursive_partitioning_for_classification (https://rstudio-pubs-static.s3.amazonaws.com/225209_df0130c5a0614790b6365676b9372c07.html#32_recursive_partitioning_for_classification)
4. *Zeileis, Hothorn, & Hornik 2006* party with the mob: Model-Based Recursive Partitioning in R https://cran.r-project.org/web/packages/party/vignettes/MOB.pdf (https://cran.r-project.org/web/packages/party/vignettes/MOB.pdf)