Evaluation of Credit Analysis Models

Patricia Yin, Josiah Suartono, Jiahui Liu

07/04/2021

Introduction:

Understanding the risk of default when assessing whether an applicant should be given a credit card is crucial for banks to avoid insolvency. If the borrower presents an acceptable level of default risk, the analyst can recommend the approval of the credit application at the agreed terms. The outcome of the credit risk analysis determines the risk rating that the borrower will be assigned and their ability to access credit. Our goal is to build a classification model to predict the type of credit card users in China based on their application information. Hence, we face a supervised learning situation and should use a classification model to predict the categorical outcomes. Furthermore, we use the AUC(area under the curve) as a performance measure for our classification problem. This project aims to compare different machine learning models against the stepwise logistic regression, which is the common industrial practice for such a task [1].

The data is from an anonymous Chinese bank, which was published on Kaggle [2]. The same data set can also be found as teaching materials in a “credit card applicant’s rating model building tutorial” in the link below [3]. Chinese banks will mark the debts which are overdue for more than 90 days as bad debt, and be entering legal proceeding if overdue was more than 180 days. Hence for performance measuring reasons, the study is focused on clients who already have a credit record of more than 3 months.

Data Dictionary:

- ID: Client number
- CODE_GENDER: Gender
- FLAG_OWN_CAR: Is there a car
- FLAG_OWN_REALTY: Is there a property
- CNT_CHILDREN: Number of children
- AMT_INCOME_TOTAL: Annual income in CNY
- NAME_INCOME_TYPE: Income category
- NAME_EDUCATION_TYPE: Education level
- NAME_FAMILY_STATUS: Marital status
- NAME_HOUSING_TYPE: Way of living
- DAYS_BIRTH: Birthday (Count backward from current day (0), -1 means yesterday)
- DAYS_EMPLOYED: Start date of employment (Count backward from current day(0). If positive, it means the person currently unemployed.)
- FLAG_MOBIL: Is there a mobile phone
- FLAG_WORK_PHONE: Is there a work phone
- FLAG_PHONE: Is there a phone
- FLAG_EMAIL: Is there an email
- OCCUPATION_TYPE: Occupation
- CNT_FAM_MEMBERS: Family size

I. Data Pre-processing

To get a first impression of both the data we take a look at the top 4 rows:

```
credit.dat <- read.csv("credit_record.csv")
credit.dat  %>%
  slice_head(n = 4) %>% #select the top 4 rows
  gt() # print output using gt
```

ID	MONTHS_BALANCE	STATUS
5001711	0	X
5001711	-1	0
5001711	-2	0
5001711	-3	0

```
application.dat <- read.csv("application_record.csv")
application.dat  %>%
  slice_head(n = 4) %>%  #select the top 4 rows
  gt() # print output using gt
```

ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_IN
5008804	M	Y	Y	0	
5008805	M	Y	Y	0	
5008806	M	Y	Y	0	
5008808	F	N	Y	0	

Next, we take a look at the data structure and check whether all data formats are correct:

```
glimpse(application.dat)
```

```
## Rows: 438,557
## Columns: 18
## $ ID                  <int> 5008804, 5008805, 5008806, 5008808, 5008809, 5008…
## $ CODE_GENDER         <chr> "M", "M", "M", "F", "F", "F", "F", "F", "F", "F",…
## $ FLAG_OWN_CAR        <chr> "Y", "Y", "Y", "N", "N", "N", "N", "N", "N", "N",…
## $ FLAG_OWN_REALTY     <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y",…
## $ CNT_CHILDREN        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ AMT_INCOME_TOTAL    <dbl> 427500, 427500, 112500, 270000, 270000, 270000, 2…
## $ NAME_INCOME_TYPE    <chr> "Working", "Working", "Working", "Commercial asso…
## $ NAME_EDUCATION_TYPE <chr> "Higher education", "Higher education", "Secondar…
## $ NAME_FAMILY_STATUS  <chr> "Civil marriage", "Civil marriage", "Married", "S…
## $ NAME_HOUSING_TYPE   <chr> "Rented apartment", "Rented apartment", "House / …
## $ DAYS_BIRTH          <int> -12005, -12005, -21474, -19110, -19110, -19110, -…
## $ DAYS_EMPLOYED       <int> -4542, -4542, -1134, -3051, -3051, -3051, -3051, …
## $ FLAG_MOBIL          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ FLAG_WORK_PHONE     <int> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0…
## $ FLAG_PHONE          <int> 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0…
## $ FLAG_EMAIL          <int> 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0…
## $ OCCUPATION_TYPE     <chr> "", "", "Security staff", "Sales staff", "Sales s…
## $ CNT_FAM_MEMBERS     <dbl> 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2…
```

```
glimpse(credit.dat)
```

```
## Rows: 1,048,575
## Columns: 3
## $ ID             <int> 5001711, 5001711, 5001711, 5001711, 5001712, 5001712, …
## $ MONTHS_BALANCE <int> 0, -1, -2, -3, 0, -1, -2, -3, -4, -5, -6, -7, -8, -9, …
## $ STATUS         <chr> "X", "0", "0", "0", "C", "C", "C", "C", "C", "C", "C",…
```

Having glimpsed the two data sets, we decide to firstly create the response variable and combine the application and credit data sets according to the ID. We then proceed to clean the response variables, continuous variables, and categorical variables.

1. Response Variable

In this study, we will treat accounts 90 days past due as a “bad” account (label=1) following the standard practice of Chinese banks.

```
# remove those account with insufficient history (less than 3 months)
credit.1 <- credit.dat %>%
  group_by(ID) %>%
  mutate(min_month=min(MONTHS_BALANCE)) %>%
  filter(min_month < -2)

# label 1 for those have ever been three or more months overdue
positive_label <- credit.dat %>%
  filter(STATUS == 3 | STATUS == 4 |STATUS == 5) %>%
  distinct(ID) %>%
  mutate(label = 1)
# Join the records with the same ID
credit.label <- full_join(credit.1, positive_label, by="ID")

# label 0 for those who are not 1
negative_label <- credit.label %>%
  filter(is.na(label)) %>%
  mutate(label = 0) %>%
  dplyr::select(ID, label)

# create a data frame with ID and label
id_label <- rbind(positive_label, negative_label) %>%
  distinct(ID, label)

# merge both data frame to include predictor variables and outcome variable by ID (re
moving accounts with no credit record)
card.dat <- inner_join(application.dat, id_label, by="ID")

# columns with "characters" into
cols <- c("CODE_GENDER", "FLAG_OWN_CAR", "FLAG_OWN_REALTY", "NAME_INCOME_TYPE",
          "NAME_EDUCATION_TYPE", "NAME_FAMILY_STATUS", "NAME_HOUSING_TYPE", "FLAG_MOB
IL",
          "FLAG_WORK_PHONE", "FLAG_PHONE", "FLAG_EMAIL", "OCCUPATION_TYPE", "label")
# coerce these variables to factors
card.dat[cols] <- lapply(card.dat[cols], factor)
summary(card.dat)
```

```
##        ID          CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  CNT_CHILDREN
##  Min.   :5008804   F:23422     N:21566      N:11486          Min.   : 0.0000
##  1st Qu.:5042020   M:11526     Y:13382      Y:23462          1st Qu.: 0.0000
##  Median :5074532                                           Median : 0.0000
##  Mean   :5078214                                           Mean   : 0.4307
##  3rd Qu.:5115401                                           3rd Qu.: 1.0000
##  Max.   :5150487                                           Max.   :19.0000
##
##  AMT_INCOME_TOTAL              NAME_INCOME_TYPE
##  Min.   :  27000   Commercial associate: 8160
##  1st Qu.: 121500   Pensioner           : 5882
##  Median : 157500   State servant       : 2867
##  Mean   : 187462   Student             :   11
##  3rd Qu.: 225000   Working             :18028
##  Max.   :1575000
##
##                      NAME_EDUCATION_TYPE            NAME_FAMILY_STATUS
##  Academic degree              :   31    Civil marriage      : 2797
##  Higher education             : 9508    Married             :24089
##  Incomplete higher            : 1346    Separated           : 2030
##  Lower secondary              :  366    Single / not married: 4585
##  Secondary / secondary special:23697    Widow               : 1447
##
##
##            NAME_HOUSING_TYPE   DAYS_BIRTH     DAYS_EMPLOYED    FLAG_MOBIL
##  Co-op apartment    :  163   Min.   :-25152   Min.   :-15713   1:34948
##  House / apartment  :31216   1st Qu.:-19439   1st Qu.: -3165
##  Municipal apartment: 1089   Median :-15578   Median : -1569
##  Office apartment   :  253   Mean   :-15989   Mean   : 59080
##  Rented apartment   :  543   3rd Qu.:-12483   3rd Qu.:  -413
##  With parents       : 1684   Max.   : -7705   Max.   :365243
##
##  FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL    OCCUPATION_TYPE  CNT_FAM_MEMBERS
##  0:27046         0:24606    0:31798               :10846   Min.   : 1.0
##  1: 7902         1:10342    1: 3150    Laborers   : 5897   1st Qu.: 2.0
##                                        Core staff : 3470   Median : 2.0
##                                        Sales staff: 3317   Mean   : 2.2
##                                        Managers   : 2934   3rd Qu.: 3.0
##                                        Drivers    : 2049   Max.   :20.0
##                                        (Other)    : 6435
##  label
##  0:34646
##  1:  302
##
##
##
##
##
```

2. Continuous Variable

(1) DAYS_BIRTH, DAYS_EMPLOYED

```
summary(card.dat$DAYS_BIRTH)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -25152  -19439  -15578  -15989  -12483   -7705
```

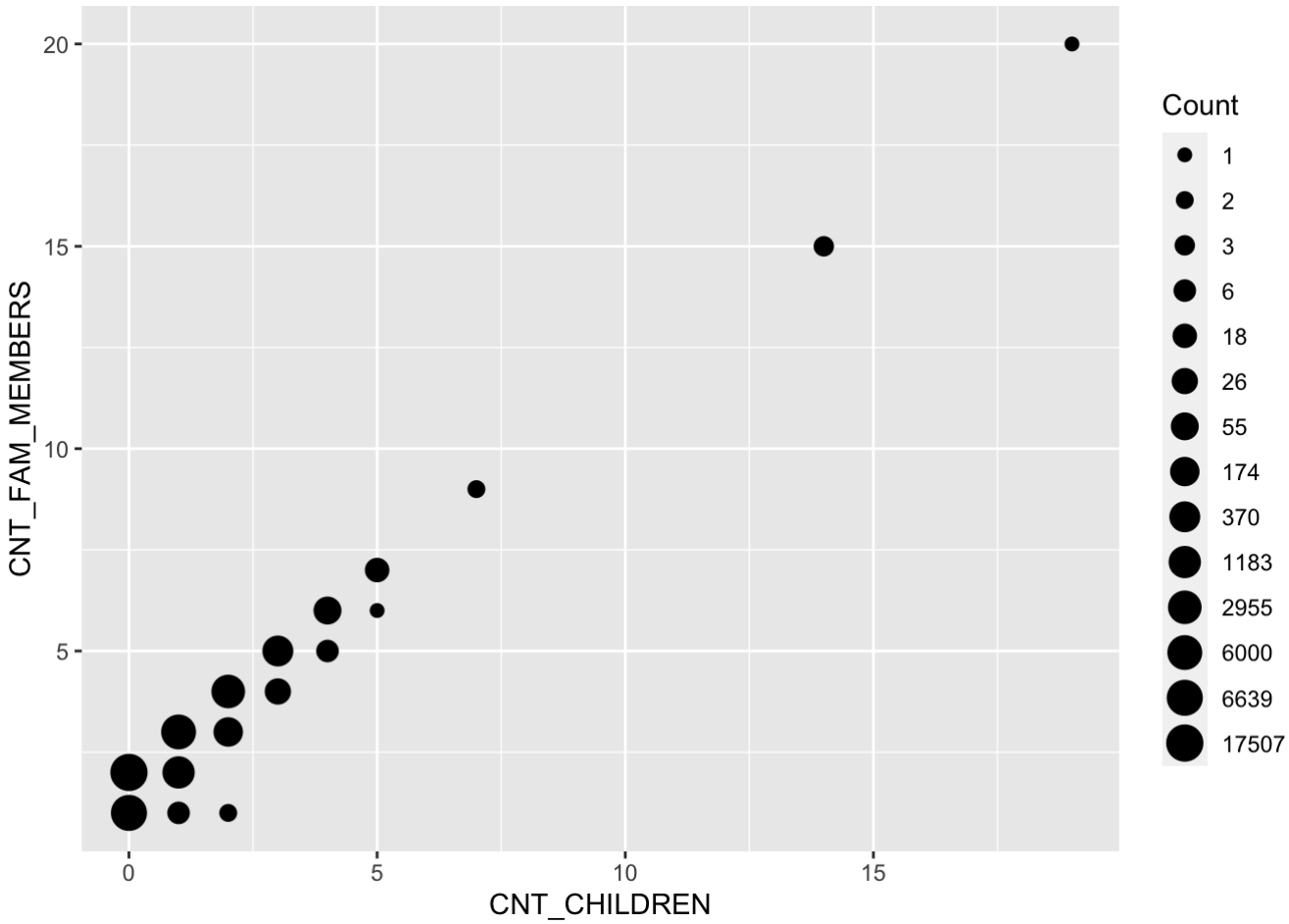
```
summary(card.dat$DAYS_EMPLOYED)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -15713   -3165   -1569   59080    -413  365243
```

```
#Negative numbers due to counting days backward
#Revert to counting days forward
#converting Birthday and Employed day to years
card.dat$age <- -card.dat$DAYS_BIRTH%/%365

#365243 in DAYS_EMPLOYED refers to unemployed by Data Dictionary
card.dat$years_employed <- ifelse(card.dat$DAYS_EMPLOYED != 365243,round(-card.dat$DA
YS_EMPLOYED/365, digits=2), 0)
```

(2) CNT_CHILDREN, CNT_FAM_MEMBERS

```
# correlation between CNT_CHILDREN and CNT_FAM_MEMBERS
ggplot(data=card.dat, aes(x=CNT_CHILDREN, y=CNT_FAM_MEMBERS)) +
  stat_sum(aes(size = factor(..n..)), geom = "point") + scale_size_discrete(name = "C
ount")
```

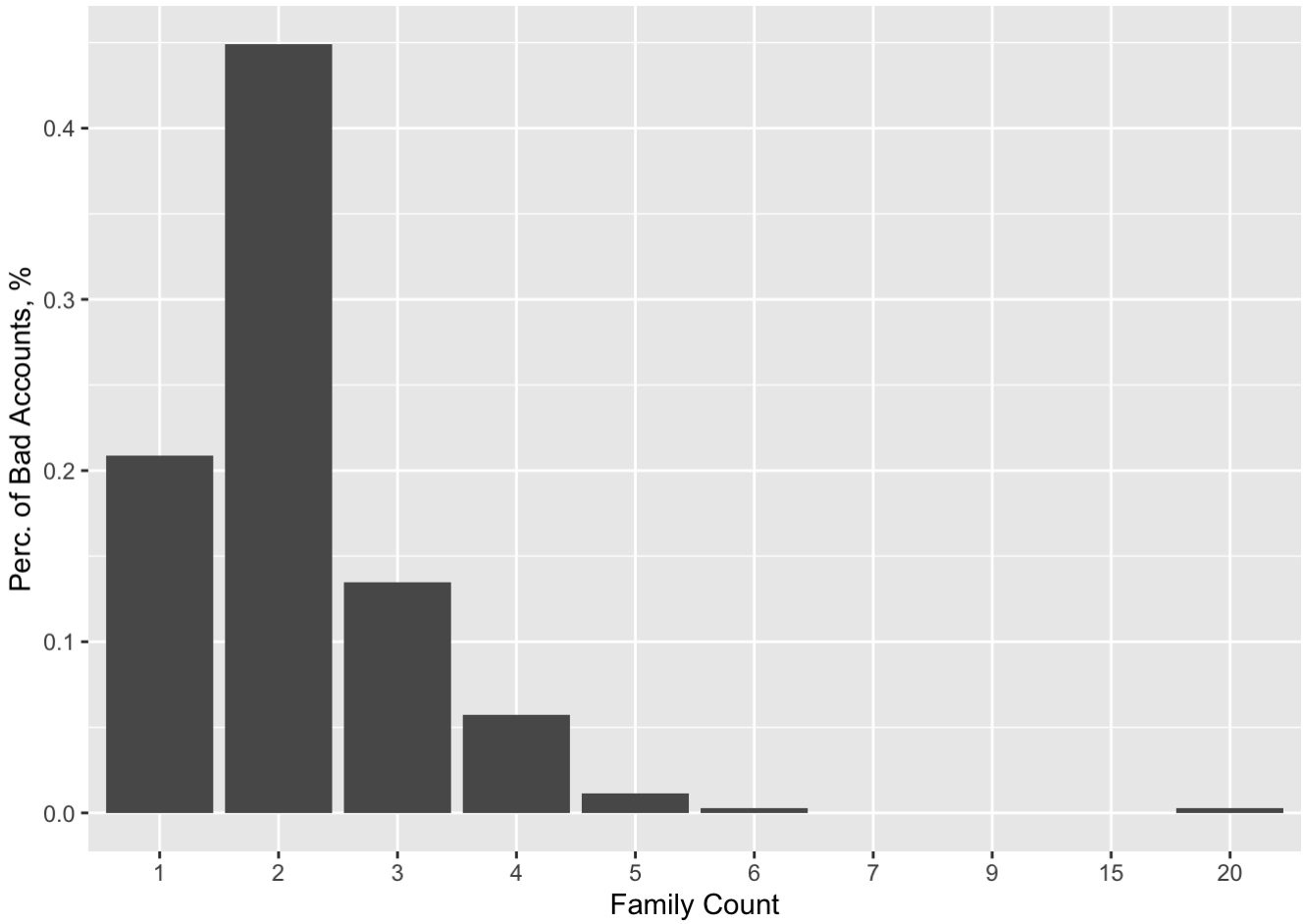


Due to the strong correlation, we decide to exclude CNT_CHILDREN from our future models.

```
card.dat <- card.dat %>% dplyr::select(-CNT_CHILDREN)
```

Also, we transform CNT_FAM_MEMBERS.

```
# calculate the percentage of "good/bad" in each level
famcnt.tab <- table(card.dat$CNT_FAM_MEMBERS, card.dat$label)
famcnt.pro.tab <- prop.table(famcnt.tab)
famcnt.df.b <- data.frame(famcnt.pro.tab)[10:20,] # data for "bad" account
# plot
famcnt.p <- ggplot(famcnt.df.b) + geom_col(aes(x=Var1, y=100*Freq)) +
  xlab("Family Count") + ylab("Perc. of Bad Accounts, %")
famcnt.p
```



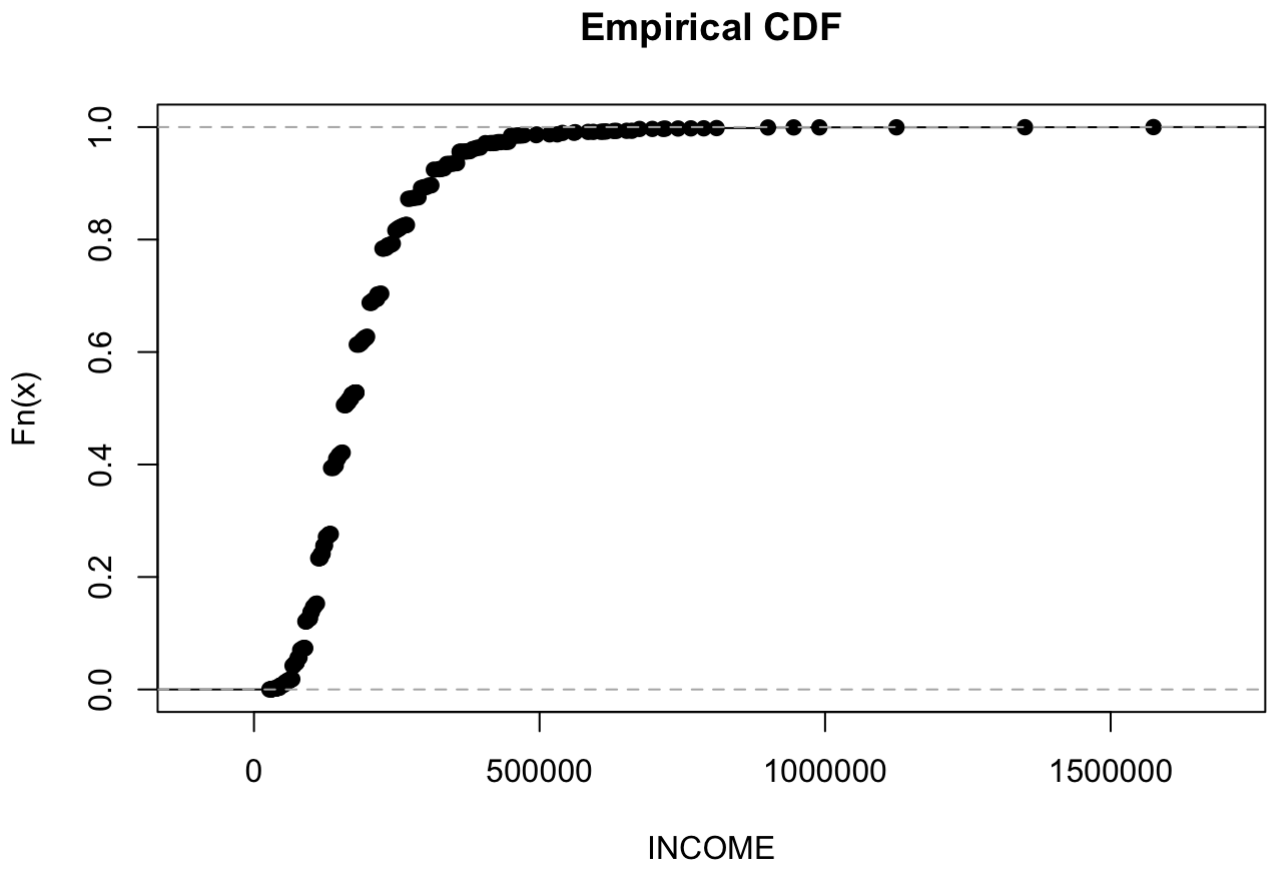
In order to avoid a level completely dominates by the “good” account, we decided to combine family count of more than 5.

```
card.dat$CNT_FAM_MEMBERS[card.dat$CNT_FAM_MEMBERS >= 5] <- "5+"
card.dat$CNT_FAM_MEMBERS[card.dat$CNT_FAM_MEMBERS == 4] <- "4"
card.dat$CNT_FAM_MEMBERS[card.dat$CNT_FAM_MEMBERS == 3] <- "3"
card.dat$CNT_FAM_MEMBERS[card.dat$CNT_FAM_MEMBERS == 2] <- "2"
card.dat$CNT_FAM_MEMBERS[card.dat$CNT_FAM_MEMBERS == 1] <- "1"

card.dat$CNT_FAM_MEMBERS <- as.factor(card.dat$CNT_FAM_MEMBERS)
table(card.dat$CNT_FAM_MEMBERS, card.dat$label)
```

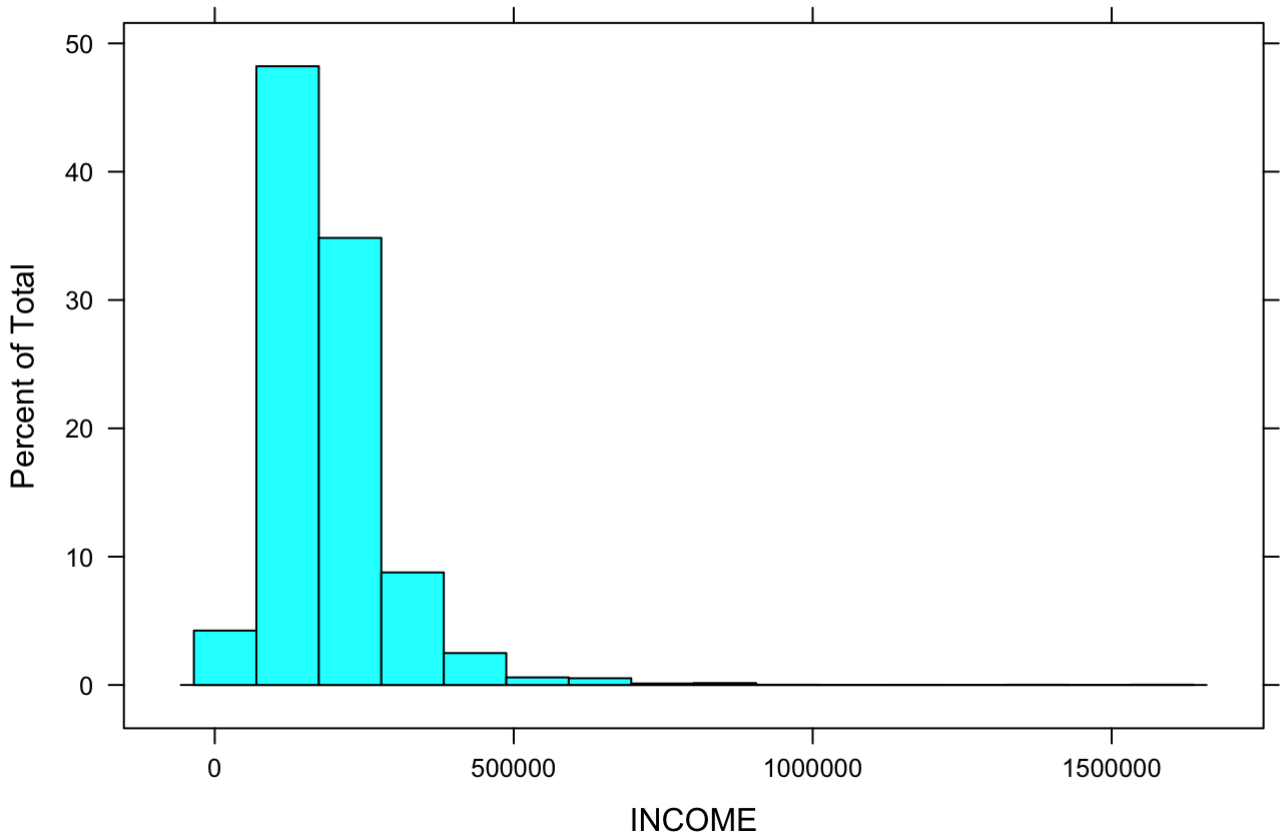
```
##
##          0     1
##   1   6574    73
##   2  18533   157
##   3   6127    47
##   4   2961    20
##   5+   451     5
```

(3) AMT_INCOME_TOTAL

```
plot(ecdf(card.dat$AMT_INCOME_TOTAL),main="Empirical CDF",xlab="INCOME")
```




```
histogram(card.dat$AMT_INCOME_TOTAL,main="PDF",xlab="INCOME")
```



Given the highly positively skewed distribution, we are interested in exploring how default risk is related to the exponential growth of income, instead of the unit increase, hence we log transform the variable to have a more direct interpretation.

```
card.dat$AMT_INCOME_TOTAL <- log(card.dat$AMT_INCOME_TOTAL)
```

3. Categorical Variables

(1) FLAG_MOBIL

```
summary(card.dat$FLAG_MOBIL)
```

```
##     1
## 34948
```

```
# Removing FLAG_MOBIL because it has only 1 unique value"
card.dat <- card.dat %>% dplyr::select(-FLAG_MOBIL)
```

(2) NAME_INCOME_TYPE

```
table(card.dat$NAME_INCOME_TYPE, card.dat$label)
```

```
##
##                             0     1
##   Commercial associate  8077    83
##   Pensioner             5811    71
##   State servant         2848    19
##   Student                 11     0
##   Working              17899   129
```

There are only 11 observations for “Student”, with no label as “1”(default). Due to the low sample size of the student subset and potential confounding effects (e.g. student tend to have student loans, but also likely to receive parents’ financial support other than income), we decide to remove the subset of students and instead focus on the default rate for working people and retirees.

```
card.dat <- subset(card.dat, card.dat$NAME_INCOME_TYPE!="Student")
card.dat$NAME_INCOME_TYPE <- droplevels(card.dat$NAME_INCOME_TYPE)
```

(3) NAME_EDUCATION_TYPE

```
table(card.dat$NAME_EDUCATION_TYPE, card.dat$label)
```

```
##
##                                   0     1
##   Academic degree                31     0
##   Higher education             9408    91
##   Incomplete higher            1326    20
##   Lower secondary               359     7
##   Secondary / secondary special 23511   184
```

There are only 31 observations for “Academic degree”. We decided to merge this level with “Higher education” because they indicated an educational level higher than secondary education.

```
levels(card.dat$NAME_EDUCATION_TYPE) <- c("Higher education","Higher education",
                                          "Incomplete higher", "Lower secondary",
                                          "Secondary / secondary special")
```

(4) NAME_FAMILY_STATUS

```
table(card.dat$NAME_FAMILY_STATUS, card.dat$label)
```

```
##
##                             0     1
##   Civil marriage         2787     8
##   Married               23883   198
##   Separated              2003    27
##   Single / not married   4533    51
##   Widow                  1429    18
```

(5) NAME_HOUSING_TYPE

```
table(card.dat$NAME_HOUSING_TYPE, card.dat$label)
```

```
##
##                           0     1
##   Co-op apartment       160     3
##   House / apartment   30946   260
##   Municipal apartment  1067    22
##   Office apartment      249     4
##   Rented apartment      538     5
##   With parents         1675     8
```

(6) OCCUPATION_TYPE

```
# Distribution of years of employment for people who do not specify occupation type
card.dat %>%
  filter(OCCUPATION_TYPE == "") %>%
  group_by(group = cut(years_employed,
                       breaks = c(-0.001, 0, 10, 20, 30, round(max(years_employed))),
                       include.lowest = FALSE)) %>%
  count() %>%
  rename(Years_Employed=group, Number_of_People=n)
```

Years_Employed <fct>	**Number_of_People** <int>
(-0.001,0]	5865
(0,10]	3696
(10,20]	973
(20,30]	245
(30,42]	66

5 rows

We observe that those who leave “OCCUPATION_TYPE” empty have varying years of employment. For people who leave OCCUPATION_TYPE blank, if they also have 0 days of employment, we infer that they are “unemployed”, otherwise, they are classified as “unknown”.

We decide to merge them according to the position’s required skill levels for the other levels.

```
card.dat <- card.dat %>%
  mutate(occupation_type = as.factor(case_when(
    # create level unemployed and unknown for occupation type
    OCCUPATION_TYPE=="" & years_employed==0 ~ "Unemployed",
    OCCUPATION_TYPE=="" & years_employed>0 ~ "Unknown",
    # recode other levels to "HighSkill" and "LowSkill"
    OCCUPATION_TYPE %in% c("Accountants", "Core Staff", "High skill tech staff",
                           "HR staff", "IT staff", "Managers", "Medicine staff") ~ "H
ighSkill",
    TRUE ~ "LowSkill"
  )))
```

II. Exploratory Analysis

Prepare dataset

```
# remove the ID and original variables for age, years_employed and occupation type
card.dat.new <- card.dat %>%
  dplyr::select(-c(ID, DAYS_BIRTH, DAYS_EMPLOYED, OCCUPATION_TYPE))
# move label to be the last column
card.dat.new <- card.dat.new[c(1:12, 14, 15, 16, 13)]
```

Class Imbalance

```
# Count of Bad account
sum(as.numeric(card.dat$label)-1)
```

```
## [1] 302
```

```
# Proportion of Bad account
badprop=sum(as.numeric(card.dat$label)-1)/length(card.dat$label)
badprop
```

```
## [1] 0.008644131
```

```
# Proportion of Good account
1-badprop
```

```
## [1] 0.9913559
```

The class imbalance problem is an underlying problem across many anomaly detection tasks. In particular, our data set consists of 99.13% to 0.86%, with 302 applicants with a bad record, which bizarrely means blindly predicting the majority class would yield 99.13% accuracy. In particular:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

- TP: True Positive
- TN: True Negative
- FP: False Positive
- FP: False Positive

The traditional technique to deal with this problem is to artificially create a balanced data set, by using a combination of over-sampling, such as Variational Autoencoders (VAE), Synthetic Minority Oversampling Technique (SMOTE), and other under-sampling techniques. However, we decided to tackle this problem by using a different performance metric, the Area Under the Curve (AUC), due to its insensitivity to the class imbalance.

The “Curve” refers to the ROC which plots false positive rate Vs. true positive rate at different probability threshold. In particular:

$$\text{false positive rate} = \frac{FP}{FP+TN}$$

$$\text{true positive rate} = \frac{TP}{TP+FN}$$

In our example, 99.13% of the majority class means that the TP and FN will account for the most of the predictions. However, the true positive rate does not change, as the change in TP and FN both cancels out.

Therefore, hereafter all models will be evaluated using AUC.

Information Value

```
# change response variable to be numeric in order to use create_infotables function
card.dat.new1 <- card.dat.new
card.dat.new1$label <- as.numeric(as.character(card.dat.new$label))

# calculate the information value for each variable
IV <- create_infotables(data=card.dat.new1, y="label", parallel=FALSE)
print(IV$Summary)
```

```
##               Variable          IV
## 14      years_employed 0.093779487
## 7   NAME_FAMILY_STATUS 0.091871637
## 4     AMT_INCOME_TOTAL 0.081969348
## 8    NAME_HOUSING_TYPE 0.058389659
## 5     NAME_INCOME_TYPE 0.051796796
## 13                 age 0.042849386
## 6  NAME_EDUCATION_TYPE 0.036034030
## 3      FLAG_OWN_REALTY 0.024889959
## 15     occupation_type 0.023733079
## 12     CNT_FAM_MEMBERS 0.021472674
## 1          CODE_GENDER 0.013057450
## 10           FLAG_PHONE 0.006966084
## 2          FLAG_OWN_CAR 0.003608747
## 9      FLAG_WORK_PHONE 0.001372343
## 11           FLAG_EMAIL 0.001058105
```

The table summarises the Information Value (IV, for continuous variable) or Net Information Value (NIV, for categorical variables) for each variable in the dataset. This value reflects the predictive power of a variable in regards to the binary response variable. In particular:

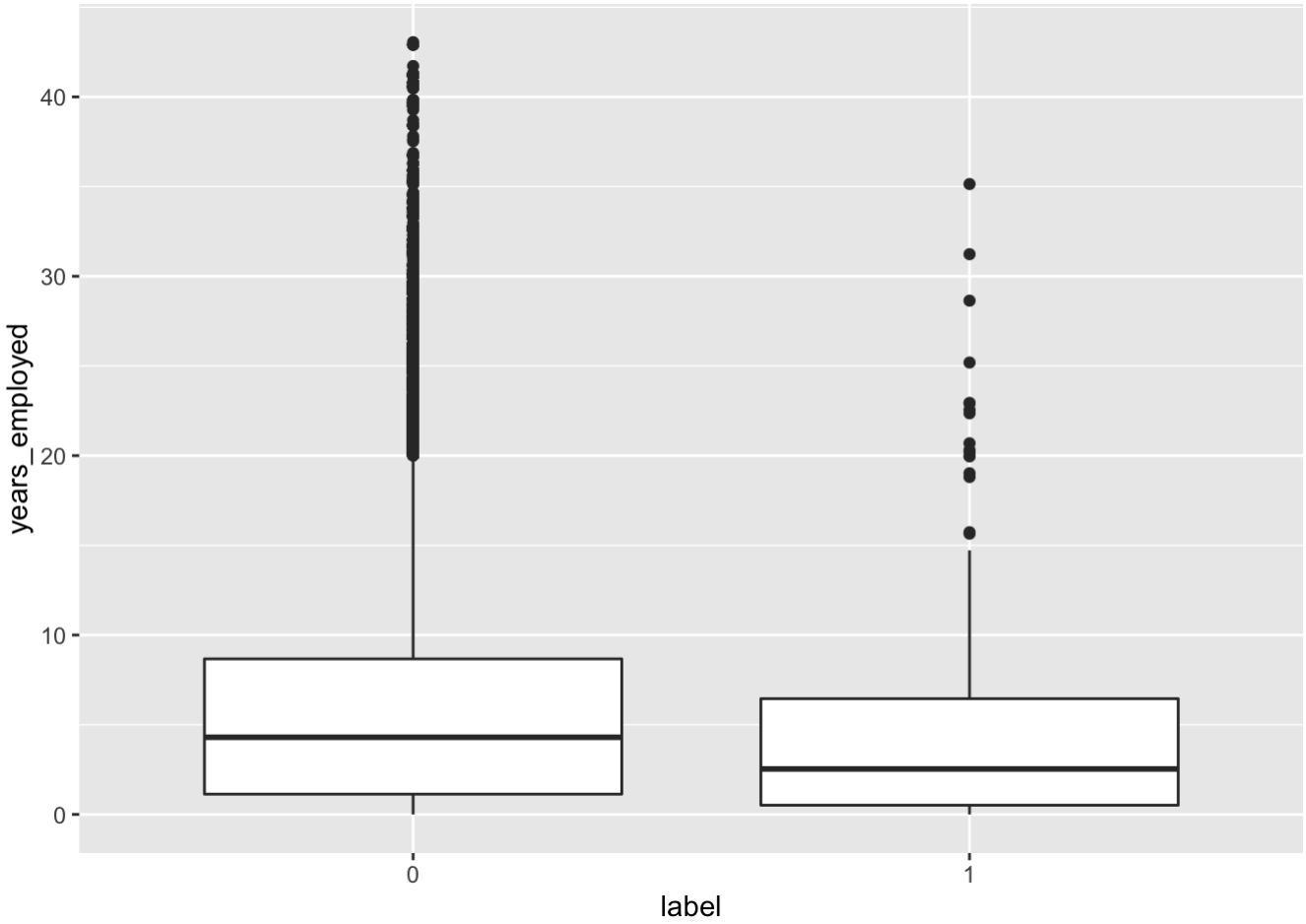
$$\text{WOE}_i = \log\left(\frac{Dist(Good)_i}{Dist(Bad)_i}\right)$$

$$\text{IV}_i = \sum_i \left(WOE_i * (Dist(Majority)_i - Dist(Minority)_i)\right)$$

- WOE: Weight of Evidence

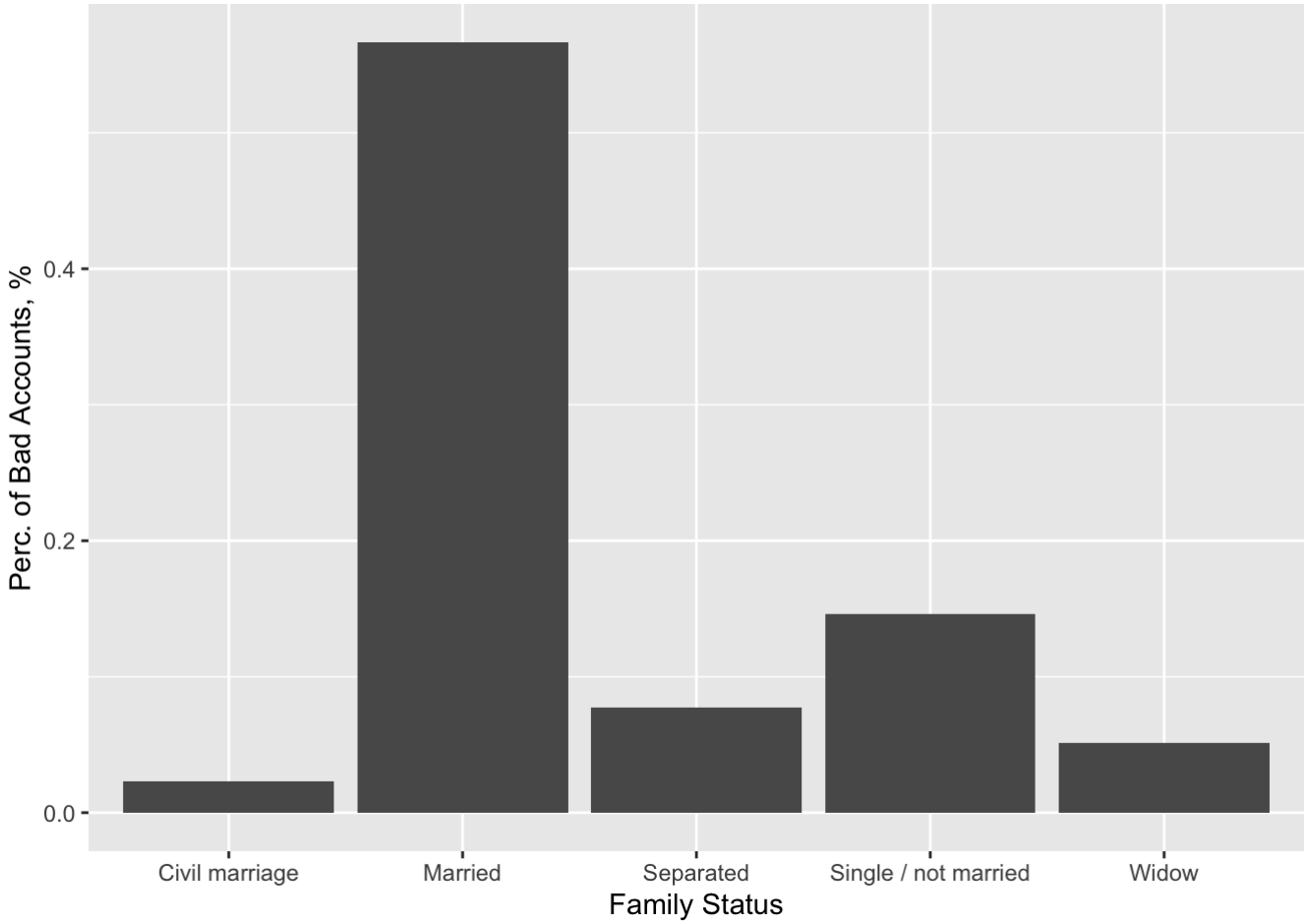
Based on this table, we see that years_employed holds the highest predictive power among all variables. From the graph below we could see that people who are identified as “bad” (label=1) tend to associate with shorter employment history. This is not surprising as this group tends to have a less stable income, affecting their ability to repay credit card debts on time.

```
# years_employed and label
ggplot(data=card.dat.new, aes(y=years_employed, x=label)) + geom_boxplot()
```



NAME_FAMILY_STATUS is the variable with the second-highest predictive power according to its NIV. Indeed, people with different family statuses tend to associate with different likelihoods of defaulting.

```
# calculate the percentage of "good/bad" in each level
fam.tab <- table(card.dat.new$NAME_FAMILY_STATUS, card.dat.new$label)
fam.pro.tab <- prop.table(fam.tab)
fam.df.b <- data.frame(fam.pro.tab)[6:10,] # data for "bad" accounts

# plot
fam.p <- ggplot(fam.df.b) +
  geom_col(aes(x=Var1, y=100*Freq)) +
  xlab("Family Status") +
  ylab("Perc. of Bad Accounts, %")
fam.p
```


Perc. of Bad Accounts, %
0.4
0.2
0.0
Civil marriage
Married
Separated
Single / not married
Widow
Family Status

III. Modeling

Local regression can perform poorly if p is much larger than about 3 or 4 because there will generally be very few training observations close to x_0. Nearest-neighbors regression, suffers from a similar problem in high dimensions. Because our data contains 15 predictor variables, we decide not to use local regression (e.g. loess) or Nearest-neighbors method to avoid this problem.

Instead, we decide to use logistic regression, SVM, and tree methods to make predictions. Because the logistic model is additive, we can still examine the effect of each x_j on Y individually while holding all of the other variables fixed. Hence if we are interested in inference, GAMS provides useful interpretation. Therefore, we would like to apply gradient descent to logistic regression as our baseline model.

```
# split training and testing dataset
set.seed(100)
card.split <- initial_split(card.dat.new, prop=3/4, strata="label")
card.train <- training(card.split)
card.test <- testing(card.split)
set.seed(103)
card.fold <- vfold_cv(card.train, v=5, strata="label")
```

Using the strata argument to make sure that the proportion of the response variable “label”=1 will be similar to that of the original data set, especially when we have imbalanced data here

1. Logistic Regression

Prerequisite

logistic regression full model

Due to the class imbalanced problem, it is not so clear what should be the threshold for classifying the predicted probabilities. Since all models in this section uses logistic regression, we proceed to use the full model’s predicted probabilities as a reference to decide on the best cut-off point.

Further splitting the training data into new.train and validation set, to avoid setting threshold based on test set

```
set.seed(40)
new.split <- initial_split(card.train, prop=3/4, strata="label")
new.train <- training(new.split)
validation <- testing(new.split)
```

```
# Fit the model
full_model <- glm(label ~., data = new.train, family = binomial)
vif(full_model)
```

```
##                         GVIF Df GVIF^(1/(2*Df))
## CODE_GENDER          1.361078  1        1.166652
## FLAG_OWN_CAR         1.227181  1        1.107782
## FLAG_OWN_REALTY      1.132009  1        1.063959
## AMT_INCOME_TOTAL     1.306029  1        1.142816
## NAME_INCOME_TYPE    14.575394  3        1.562920
## NAME_EDUCATION_TYPE  1.287929  3        1.043075
## NAME_FAMILY_STATUS   5.011797  4        1.223205
## NAME_HOUSING_TYPE    1.169356  5        1.015768
## FLAG_WORK_PHONE      1.315062  1        1.146762
## FLAG_PHONE           1.122579  1        1.059518
## FLAG_EMAIL           1.069705  1        1.034265
## CNT_FAM_MEMBERS      4.971235  4        1.221963
## age                  2.288959  1        1.512930
## years_employed       1.379470  1        1.174508
## occupation_type     16.288847  3        1.592142
```

VIF stands for Variance Inflation Factor, and it is calculated as:

$$\mathrm{VIF}_i = \frac{1}{1-R_i^2}$$

- Ri: Unadjusted coefficient for determination for regressing the ith independent variable on the remaining ones

As a reference, VIF higher than 10 have significant multicollinearity that needs to be corrected; VIF higher than 5 might indicate multicollinearity exist.

Seeing the results of VIF above, income type and occupation type seem to have multicollinearity problem, we remove income type to see whether the problem can be solved.

```
full_model <- glm(label ~.-NAME_INCOME_TYPE, data=new.train, family=binomial)
vif(full_model)
```

```
##                         GVIF Df GVIF^(1/(2*Df))
## CODE_GENDER         1.359460  1        1.165959
## FLAG_OWN_CAR        1.228847  1        1.108534
## FLAG_OWN_REALTY     1.124822  1        1.060576
## AMT_INCOME_TOTAL    1.277050  1        1.130066
## NAME_EDUCATION_TYPE 1.259584  3        1.039213
## NAME_FAMILY_STATUS  4.922860  4        1.220470
## NAME_HOUSING_TYPE   1.160347  5        1.014983
## FLAG_WORK_PHONE     1.306187  1        1.142885
## FLAG_PHONE          1.124610  1        1.060477
## FLAG_EMAIL          1.058311  1        1.028743
## CNT_FAM_MEMBERS     4.937778  4        1.220932
## age                 2.267276  1        1.505748
## years_employed      1.367081  1        1.169223
## occupation_type     2.728782  3        1.182120
```

The problem is solved as no vif value is larger than 5 now

We will now use Youden-Index to determine the best cut-off point

Youden-Index: Detemine the point for which (sensitifity + specificity) is maximal.

- correctly classified as “positive” = true-positive-rate = sensitivity

- correctly classified as “negative” = true-negative-rate = specificity

```
# Make predictions for validation data
full.probabilities.test <- full_model %>% predict(validation, type="response")

set.seed(1)
#find the best cut-off point using the validation data [4]
card.test.cp <- data.frame(validation, full.probabilities.test)
cp <- cutpointr(card.test.cp,full.probabilities.test,label,
                metric = sum_sens_spec,
                method = maximize_metric)
```

```
## Assuming the positive class is 1
```

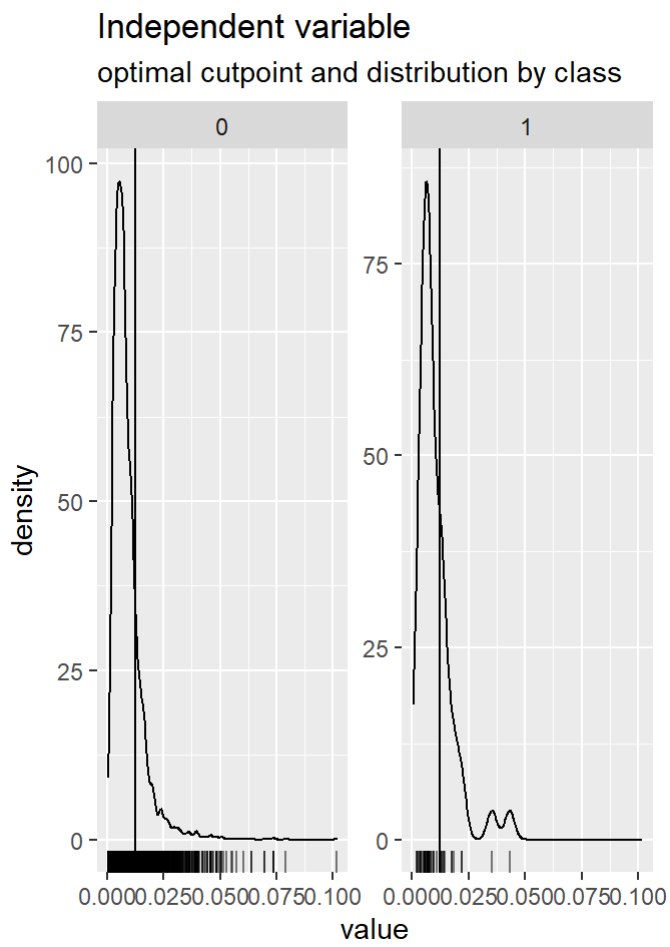
```
## Assuming the positive class has higher x values
```

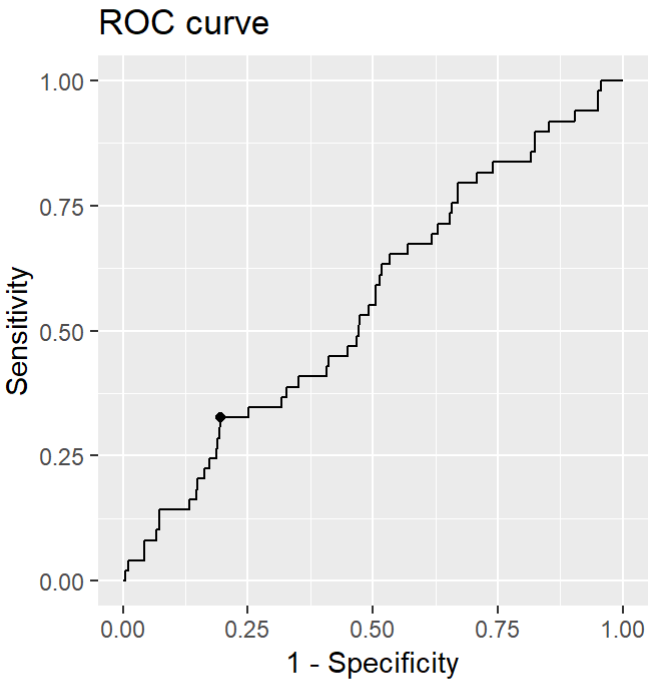
```
summary(cp)
```

cutpointr	desc	desc_by_class	n_...	n_...	n_...	confusion_mat
<list>	<list>	<list>	<int>	<int>	<int>	<lis
<cutpointr[,16]>	<tibble[,11]>	<df[,11] [2 × 11]>	6550	49	6501	<df[,5] [1 × 5

1 row

```
plot(cp)
```





The optimal cut-point is found where is the point where sensitivity + specificity is maximized. Hence hereafter we will use this threshold to evaluate all predicted probabilities.

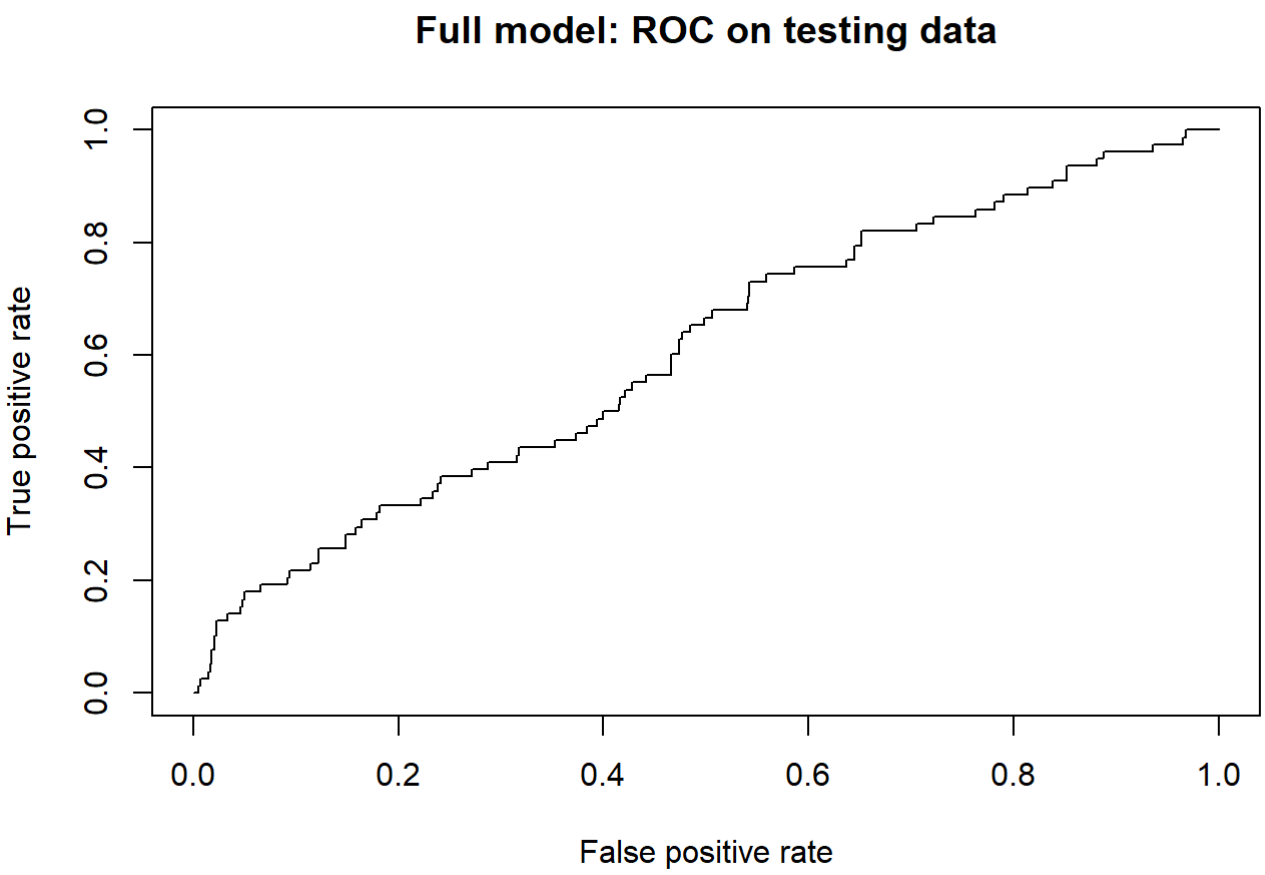
```
#apply to all the logistic regression
cpopt <- cp$optimal_cutpoint
```

Now we check the performance of the model

```
#Evaluate Full Model on testing set
full.probabilities.test <- full_model %>% predict(card.test, type="response")
full.predicted.classes.test <- as.factor(ifelse(full.probabilities.test > cpopt, 1, 0
))
# Model confusion matrix
full.cm.test <- confusionMatrix(full.predicted.classes.test,card.test$label)
full.cm.test[["table"]]
```

```
##            Reference
## Prediction    0    1
##          0 6970   52
##          1 1686   26
```

```
# ROC on testing data
## prepare an object for ROC curve
full.prediction <- prediction(full.probabilities.test, card.test$label)
full.roc.obj <- performance(full.prediction, measure="tpr", x.measure="fpr")
## plot ROC curve
plot(full.roc.obj, main="Full model: ROC on testing data")
```



```
#AUC Area under curve
full.log.auc <- performance(full.prediction, measure = "auc")@y.values
full.log.auc
```

```
## [[1]]
## [1] 0.6066305
```

A logistic model with almost all predictors is not good, as this includes many insignificant variables that could inhibit the result. Hence we now consider our baseline model with 2 predictors.

(1) Baseline - 2 predictors

As a baseline model, we decided to build a logistic regression with two predictors, years_employed and CODE_GENDER. The two predictors were picked based on domain knowledge: people with longer employment history tend to be less reliant on credit cards for expenses, therefore less likely to overdraft and have loan overdue. Also, several research papers have shown that women tend to be more credit conscious [5].

We build this baseline regression model using gradient descent.

```
# since gender only has two levels, we recode it to be a numeric variable
# with values 1 (for Male) and 2 (for Female) for simplicity in coding
x_gender <- as.numeric(card.train$CODE_GENDER)
x_emp <- card.train$years_employed

X <- cbind(rep(1, length(x_gender)), x_gender, x_emp)
Y <- as.matrix(as.numeric(as.character(card.train$label)))

# logit function
g <- function(p) return(z = log(p / (1-p)))

# inverse of logit function (sigmoid function)
g_inv <- function(z) return(p = 1 / (1+exp(-z)))

# loss function [referecne **]
loss_fn <- function(beta, X, Y) {
  m <- dim(X)[1] # no. of observations
  p <- g_inv(X%*%beta)
  J <- (t(-Y) %*% log(p) - t(1-Y) %*% log(1-p))/m
  return(J)
}

# gradient of loss function
grd_descent <- function(X, Y, beta, step_size) {
  m <- dim(X)[1]  # no. of observations
  # set random values in order to enter the while loop
  loss_t0 <- 100 ; loss_t1 <- 1
  iter <- 0
  while(abs(loss_t0 - loss_t1) > 0.000001) {
    loss_t0 <- loss_fn(beta, X, Y) # calculate initial loss
    p <- g_inv(X %*% beta) # probabilities at t0
    grad <- (1/m) * t(X)%*%(p - Y) # gradient of the loss function
    # update new estimates of beta (at t1)
    beta <- beta - (0.999)^iter * grad / sqrt(sum(grad^2))
    loss_t1 <- loss_fn(beta, X, Y) # calculate new loss
    iter <- iter + 1
  }
  return(beta)
}

# initialise beta
beta <- as.matrix(c(0, 0, 0))
# get coefficient after gradient descent
beta_f <- grd_descent(X, Y, beta, 0.001)
beta_f
```

```
##                 [,1]
##          -5.06965865
## x_gender  0.35299839
## x_emp    -0.03241457
```

In ordert to verify the correctness of our implementation of gradient descent, we then run the glm function to compare the results.

```
glm2.0 <- glm(Y ~ X, family=binomial)
summary(glm2.0)
```

```
##
## Call:
## glm(formula = Y ~ X, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.1594  -0.1396  -0.1308  -0.1203   3.4104
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.06873    0.21254 -23.849  < 2e-16 ***
## X                NA         NA      NA       NA
## Xx_gender    0.35449    0.13626   2.602  0.00928 **
## Xx_emp      -0.03126    0.01241  -2.518  0.01179 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2579.4  on 26202  degrees of freedom
## Residual deviance: 2565.6  on 26200  degrees of freedom
## AIC: 2571.6
##
## Number of Fisher Scoring iterations: 8
```

From the summary, we see that our results are similar to those generated by glm2.0. Also we note that both gender and years_employed are significant predictors.

We would use the model generated from our implementation of gradient descent for the discussion below. We conclude the following models for male and female:

$$\widehat{ln(odds_{female})} = -5.0697 + 0.3528 * 2 - 0.02957 * years\_employed$$

```
$= -4.3641 - 0.02957 * years\_employed$
```

$$\widehat{ln(odds_{male})} = -5.0697 + 0.3528 * 1 - 0.02957 * years\_employed$$

```
$= -4.7169 - 0.02957 * years\_employed$
```

In this model, we observe that if years_employed increases by 1 unit, odds of being “bad” changes by a factor of $ e^{-0.02957} $, which is 3% decrease. As compared to men, women’s odds of being “bad” increases by $(e^{0.3528} - 1) * 100\% = 42.3\%$.

To understand the probability, we assume the following two examples:

- female card holder who has an employment history of 3 years: based on this model, the odds of her being a “bad” account, which is to delay repayment for more than 90 days, is $\widehat{odds} = -4.3641 - 0.02957 * 3 = -4.4528$. The probability of her being a “bad” account is $\frac{e^{\widehat{odds}}}{1+e^{\widehat{odds}}} = 0.0115$

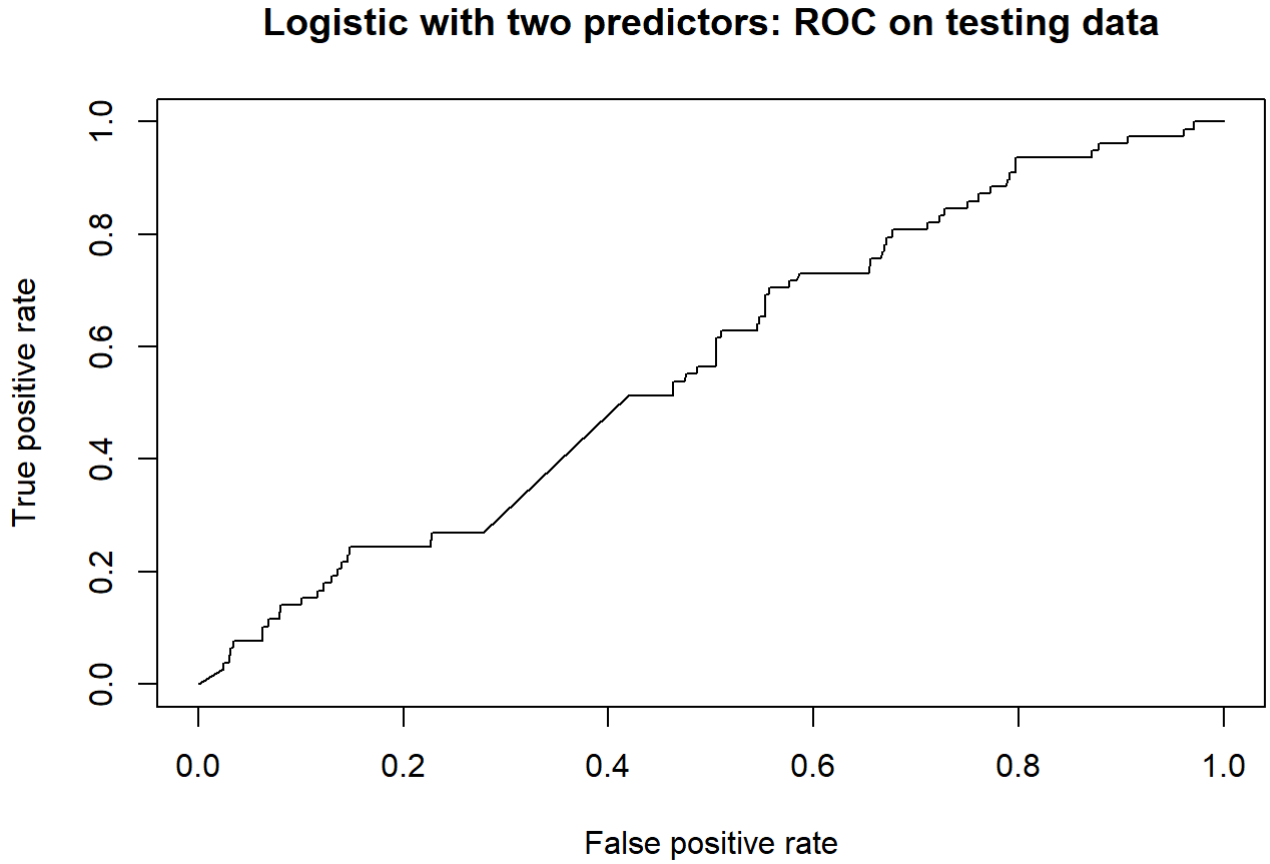
- male card holder who has an employment history of 0 year: based on this model, the odds of him being a "bad" account, is $\widehat{odds} = -4.7169 - 0.02957 * 0 = -4.7169$. The probability of her being a "bad" account is $\frac{e^{\widehat{odds}}}{1+e^{\widehat{odds}}} = 0.00886$.

The results are rather counter-intuitive as women are generally associated with less probability of defaulting and years of employment is expected to be a more influential predictor than gender. We test the performance on testing data and conclude it is unsatisfactory based on the confusion matrix.

```
## Test the model on testing data
# prepare test dataset to get X matrix for intercept and predictors
x_gender_t <- as.numeric(card.test$CODE_GENDER)
x_emp_t <- card.test$years_employed
X_t <- cbind(rep(1, length(x_gender_t)), x_gender_t, x_emp_t)
Y_t <- as.matrix(as.numeric(as.character(card.test$label)))
# calculate estimated probability of being 1
p_t <- g_inv(X_t%*%beta_f)
# convert matrix to numeric
test.prob <- as.numeric(p_t)

# Make predictions
glm2.predicted.classes <- as.factor(ifelse(test.prob > cpopt, 1, 0))
# Model accuracy
glm2.cm.test <- confusionMatrix(glm2.predicted.classes,card.test$label)
glm2.cm.test[["table"]]
```

```
##            Reference
## Prediction    0    1
##          0 8363   72
##          1  293    6
```

```
# ROC on testing data
## prepare an object for ROC curve
glm2.prediction <- prediction(test.prob, card.test$label)
glm2.roc.obj <- performance(glm2.prediction, measure="tpr", x.measure="fpr")
## plot ROC curve
plot(glm2.roc.obj, main="Logistic with two predictors: ROC on testing data")
```




```
# calculate AUC value
glm2.auc <- performance(glm2.prediction, measure = "auc")@y.values
glm2.auc
```

```
## [[1]]
## [1] 0.56834
```

(2) Best subset regression

Because we are interested in models with low test error, training set RSS and R^2 cannot be used to select from among a set of models with different number of variables. Because we are using maximized log-likelihood when running logistic regression, we use AIC as cross-validation criterion

```
#perform all-subset logistic regression based on Akaike Information Criteria (AIC)

# Fit the model

res.best.logistic <-
    bestglm(Xy = card.train,
            family = binomial,
            IC = "AIC",
            method = "exhaustive")
```

```
## Morgan-Tatar search since family is non-gaussian.
```

```
## Note: factors present with more than 2 levels.
```

```
## Show top 5 models
res.best.logistic$BestModels
```

CODE_GE… <lgl>	FLAG_OWN… <lgl>	FLAG_OWN_RE… <lgl>	AMT_INCOME_T… <lgl>	NAME_INCOME_… <lgl>	NAME_
TRUE	TRUE	TRUE	FALSE	TRUE	
TRUE	TRUE	TRUE	FALSE	TRUE	
TRUE	TRUE	TRUE	FALSE	TRUE	
TRUE	TRUE	TRUE	TRUE	TRUE	
TRUE	TRUE	TRUE	FALSE	TRUE	

5 rows | 1-6 of 16 columns

```
summary(res.best.logistic$BestModel)
```

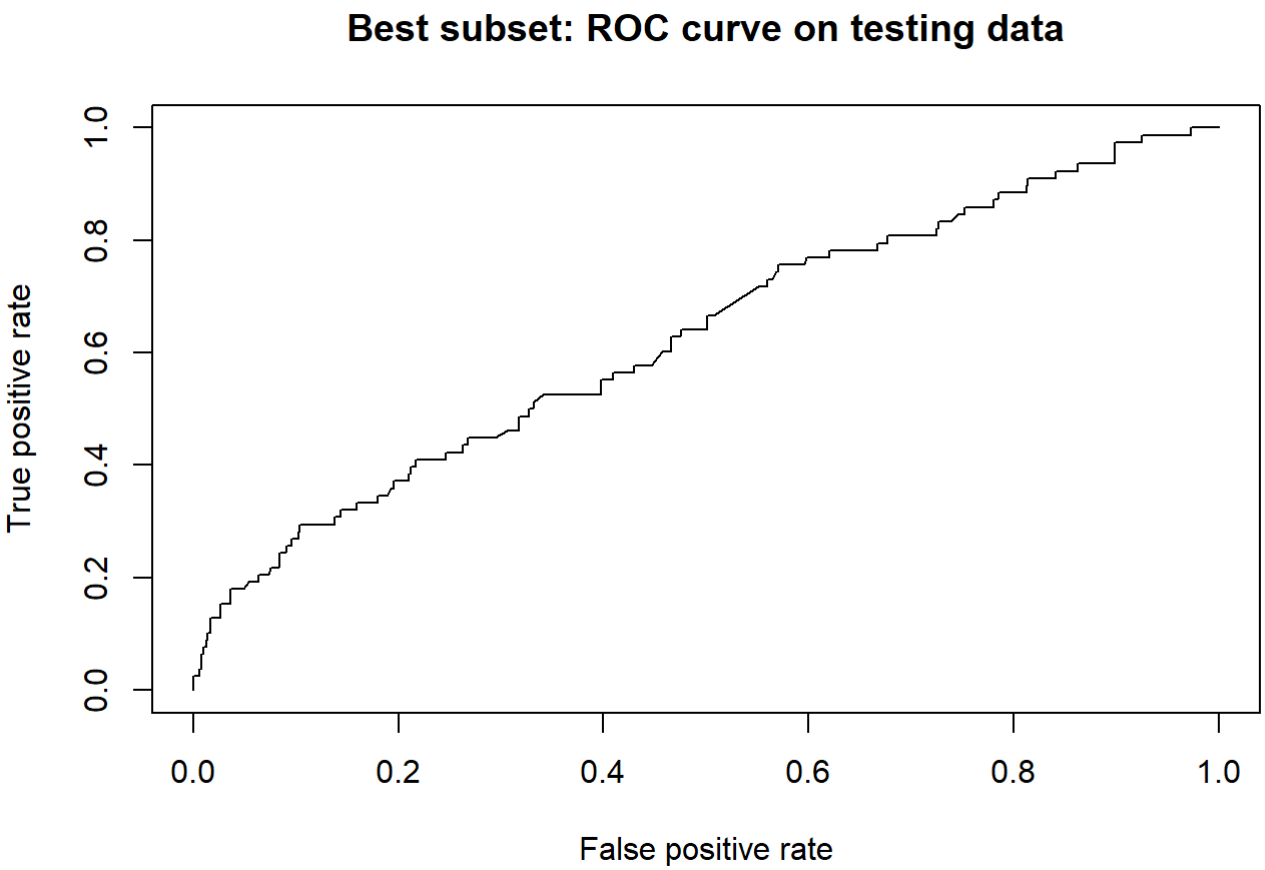
```
##
## Call:
## glm(formula = y ~ ., family = family, data = Xi, weights = weights)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2880  -0.1420  -0.1188  -0.1001   3.8491
##
## Coefficients:
##                                              Estimate Std. Error z value
## (Intercept)                                  -4.89511    0.84893  -5.766
## CODE_GENDERM                                  0.47694    0.15429   3.091
## FLAG_OWN_CARY                                -0.31338    0.15569  -2.013
## FLAG_OWN_REALTYY                             -0.29456    0.14263  -2.065
## NAME_INCOME_TYPEPensioner                     4.15985    0.59646   6.974
## NAME_INCOME_TYPEState servant                -0.20309    0.27938  -0.727
## NAME_INCOME_TYPEWorking                      -0.40824    0.16538  -2.468
## NAME_EDUCATION_TYPEIncomplete higher          0.58870    0.28543   2.062
## NAME_EDUCATION_TYPELower secondary            0.59316    0.47672   1.244
## NAME_EDUCATION_TYPESecondary / secondary special -0.24555    0.15986  -1.536
## NAME_FAMILY_STATUSMarried                     0.99451    0.39374   2.526
## NAME_FAMILY_STATUSSeparated                   1.52579    0.44597   3.421
## NAME_FAMILY_STATUSSingle / not married        1.06748    0.42205   2.529
## NAME_FAMILY_STATUSWidow                       1.61886    0.47267   3.425
## NAME_HOUSING_TYPEHouse / apartment           -0.61364    0.72362  -0.848
## NAME_HOUSING_TYPEMunicipal apartment          0.14031    0.76506   0.183
## NAME_HOUSING_TYPEOffice apartment            -0.47556    1.01405  -0.469
## NAME_HOUSING_TYPERented apartment            -0.60157    0.87859  -0.685
## NAME_HOUSING_TYPEWith parents                -1.28034    0.81600  -1.569
## years_employed                               -0.02528    0.01397  -1.810
## occupation_typeLowSkill                       0.45919    0.21256   2.160
## occupation_typeUnemployed                    -4.05785    0.64117  -6.329
## occupation_typeUnknown                        0.19335    0.26785   0.722
##                                              Pr(>|z|)
## (Intercept)                                  8.11e-09 ***
## CODE_GENDERM                                 0.001994 **
## FLAG_OWN_CARY                                0.044132 *
## FLAG_OWN_REALTYY                             0.038900 *
## NAME_INCOME_TYPEPensioner                    3.07e-12 ***
## NAME_INCOME_TYPEState servant                0.467271
## NAME_INCOME_TYPEWorking                      0.013568 *
## NAME_EDUCATION_TYPEIncomplete higher         0.039163 *
## NAME_EDUCATION_TYPELower secondary           0.213411
## NAME_EDUCATION_TYPESecondary / secondary special 0.124529
## NAME_FAMILY_STATUSMarried                    0.011545 *
## NAME_FAMILY_STATUSSeparated                  0.000623 ***
## NAME_FAMILY_STATUSSingle / not married       0.011430 *
## NAME_FAMILY_STATUSWidow                      0.000615 ***
## NAME_HOUSING_TYPEHouse / apartment           0.396427
## NAME_HOUSING_TYPEMunicipal apartment         0.854488
## NAME_HOUSING_TYPEOffice apartment            0.639093
## NAME_HOUSING_TYPERented apartment            0.493539
## NAME_HOUSING_TYPEWith parents                0.116638
## years_employed                               0.070303 .
## occupation_typeLowSkill                      0.030753 *
```

```
## occupation_typeUnemployed                          2.47e-10 ***
## occupation_typeUnknown                             0.470369
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2579.4  on 26202  degrees of freedom
## Residual deviance: 2476.3  on 26180  degrees of freedom
## AIC: 2522.3
##
## Number of Fisher Scoring iterations: 8
```

```
subset.glm <- res.best.logistic$BestModel

# Make predictions for test data
subset.probabilities.test <- subset.glm %>% predict(card.test, type = "response")
subset.predicted.classes.test <- as.factor(ifelse(subset.probabilities.test > cpopt,
1, 0))
# Model confusion matrix
subset.cm.test <- confusionMatrix(subset.predicted.classes.test,card.test$label)
subset.cm.test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7332   53
##          1 1324   25
##
##                Accuracy : 0.8423
##                  95% CI : (0.8345, 0.8499)
##     No Information Rate : 0.9911
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0185
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.84704
##             Specificity : 0.32051
##          Pos Pred Value : 0.99282
##          Neg Pred Value : 0.01853
##              Prevalence : 0.99107
##          Detection Rate : 0.83948
##    Detection Prevalence : 0.84555
##       Balanced Accuracy : 0.58378
##
##        'Positive' Class : 0
##
```

```
# plot ROC
## prepare an object for ROC curve
subset.prediction <- prediction(subset.probabilities.test, card.test$label)
subset.roc.obj <- performance(subset.prediction, measure="tpr", x.measure="fpr")
## plot ROC curve
plot(subset.roc.obj, main="Best subset: ROC curve on testing data")
```




```
#AUC Area under curve
subset.auc <- performance(subset.prediction, measure = "auc")@y.values
subset.auc
```

```
## [[1]]
## [1] 0.6272972
```

The best subset model has dropped INCOME_TOTAL, FLAG_WORK_PHONE, FLAG_PHONE, FLAG_EMAIL, CNT_FAMILY_MEMBERS, age. The algorithm's runtime is 1.5 hr, as it is considering every combination of the variables. To save computational time, we try stepwise logistic regression.

(3) Stepwise regression

```
#another way to do stepwise regression
set.seed(123)

# Fit the model--stepwise regression
step. <- glm(label ~.,
                data = card.train,
                family = binomial)

#using AIC as selection criteria
step.glm <- step.%>%
  stepAIC(trace = FALSE)

# Summarize the final selected model
summary(step.glm)
```

```
##
## Call:
## glm(formula = label ~ CODE_GENDER + FLAG_OWN_CAR + FLAG_OWN_REALTY +
##     NAME_INCOME_TYPE + NAME_EDUCATION_TYPE + NAME_FAMILY_STATUS +
##     NAME_HOUSING_TYPE + years_employed + occupation_type, family = binomial,
##     data = card.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2880  -0.1420  -0.1188  -0.1001   3.8491
##
## Coefficients:
##                                              Estimate Std. Error z value
## (Intercept)                                 -4.89511    0.84893  -5.766
## CODE_GENDERM                                 0.47694    0.15429   3.091
## FLAG_OWN_CARY                               -0.31338    0.15569  -2.013
## FLAG_OWN_REALTYY                            -0.29456    0.14263  -2.065
## NAME_INCOME_TYPEPensioner                    4.15985    0.59646   6.974
## NAME_INCOME_TYPEState servant               -0.20309    0.27938  -0.727
## NAME_INCOME_TYPEWorking                     -0.40824    0.16538  -2.468
## NAME_EDUCATION_TYPEIncomplete higher         0.58870    0.28543   2.062
## NAME_EDUCATION_TYPELower secondary           0.59316    0.47672   1.244
## NAME_EDUCATION_TYPESecondary / secondary special -0.24555    0.15986  -1.536
## NAME_FAMILY_STATUSMarried                    0.99451    0.39374   2.526
## NAME_FAMILY_STATUSSeparated                  1.52579    0.44597   3.421
## NAME_FAMILY_STATUSSingle / not married       1.06748    0.42205   2.529
## NAME_FAMILY_STATUSWidow                      1.61886    0.47267   3.425
## NAME_HOUSING_TYPEHouse / apartment          -0.61364    0.72362  -0.848
## NAME_HOUSING_TYPEMunicipal apartment         0.14031    0.76506   0.183
## NAME_HOUSING_TYPEOffice apartment           -0.47556    1.01405  -0.469
## NAME_HOUSING_TYPERented apartment           -0.60157    0.87859  -0.685
## NAME_HOUSING_TYPEWith parents               -1.28034    0.81600  -1.569
## years_employed                              -0.02528    0.01397  -1.810
## occupation_typeLowSkill                      0.45919    0.21256   2.160
## occupation_typeUnemployed                   -4.05785    0.64117  -6.329
## occupation_typeUnknown                       0.19335    0.26785   0.722
##                                             Pr(>|z|)
## (Intercept)                                 8.11e-09 ***
## CODE_GENDERM                                0.001994 **
## FLAG_OWN_CARY                               0.044132 *
## FLAG_OWN_REALTYY                            0.038900 *
## NAME_INCOME_TYPEPensioner                   3.07e-12 ***
## NAME_INCOME_TYPEState servant               0.467271
## NAME_INCOME_TYPEWorking                     0.013568 *
## NAME_EDUCATION_TYPEIncomplete higher        0.039163 *
## NAME_EDUCATION_TYPELower secondary          0.213411
## NAME_EDUCATION_TYPESecondary / secondary special 0.124529
## NAME_FAMILY_STATUSMarried                   0.011545 *
## NAME_FAMILY_STATUSSeparated                 0.000623 ***
## NAME_FAMILY_STATUSSingle / not married      0.011430 *
## NAME_FAMILY_STATUSWidow                     0.000615 ***
## NAME_HOUSING_TYPEHouse / apartment          0.396427
## NAME_HOUSING_TYPEMunicipal apartment        0.854488
## NAME_HOUSING_TYPEOffice apartment           0.639093
## NAME_HOUSING_TYPERented apartment           0.493539
```

```
## NAME_HOUSING_TYPEWith parents                    0.116638
## years_employed                                   0.070303 .
## occupation_typeLowSkill                          0.030753 *
## occupation_typeUnemployed                        2.47e-10 ***
## occupation_typeUnknown                           0.470369
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2579.4  on 26202  degrees of freedom
## Residual deviance: 2476.3  on 26180  degrees of freedom
## AIC: 2522.3
##
## Number of Fisher Scoring iterations: 8
```

Stepwise Regression has chosen to drop INCOME_TOTAL, FLAG_PHONE, FLAG_EMAIL, CNT_FAM_MEMBERS, age, just one less than the best subset model.

The most significant variable is the FAMILY_STATUS, with all of its levels having a p-value of less than 0.05. In contrast with the baseline model, gender and years employed have a relatively lower significance, although the coefficient for years employed is very similar to the baseline model.

In this model, we can interpret the coefficient as such: observe that if years_employed increases by 1 unit, odds of being “bad” changes by a factor of $ e^{-0.02528} $, which is 3% decrease. As compared to man, woman’s odds of being “bad” increases by $(e^{0.47694} - 1) * 100\% = 61.1\%$.

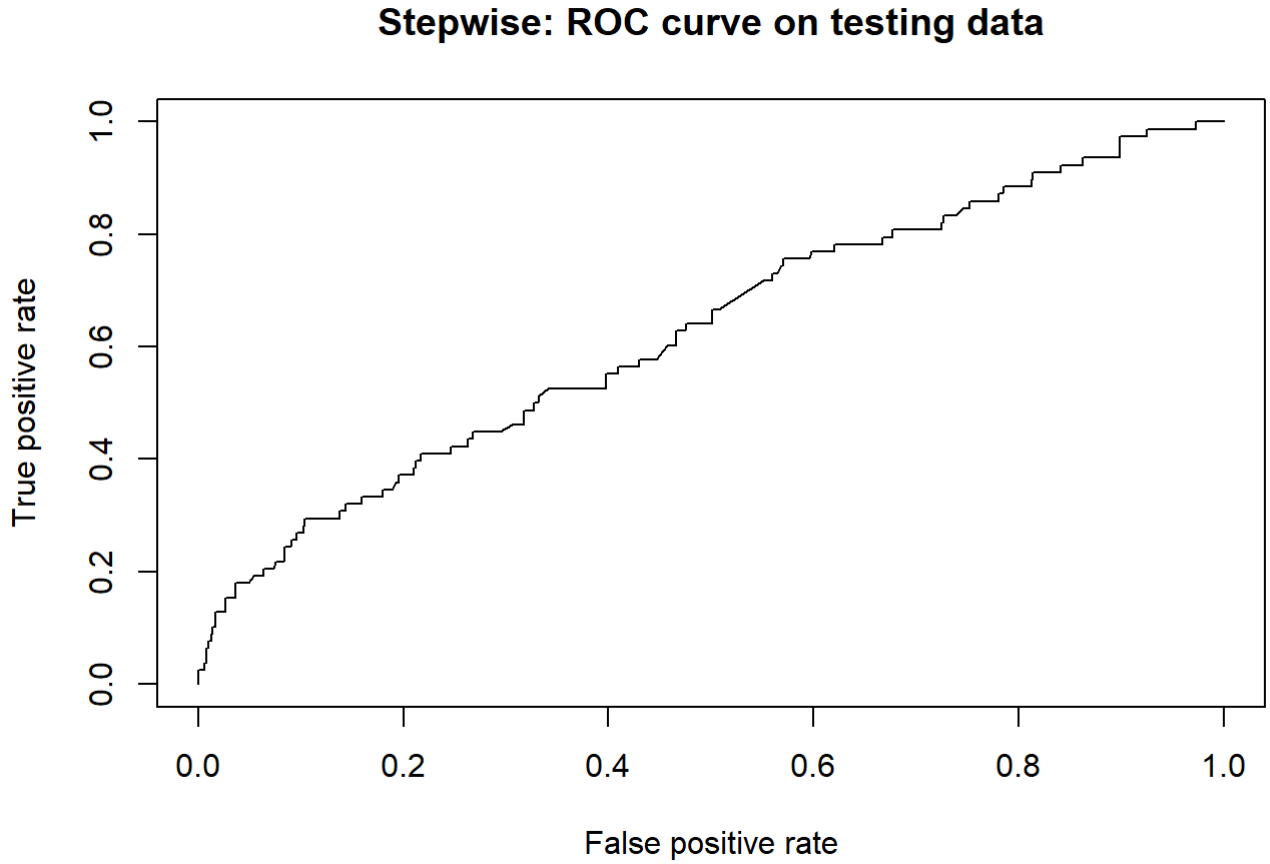
To understand the probability, we assume the following two examples: (For all variable that is not mentioned, assume the first level of each predictor)

- female card holder who has an employment history of 3 years: based on this model, the odds of her being a “bad” account, which is to delay repayment for more than 90 days, is $\widehat{odds} = -4.89511 - 0.47694 - 0.02528 * 3 = -5.44789$. The probability of her being a “bad” account is $\frac{e^{\widehat{odds}}}{1+e^{\widehat{odds}}} = 0.00428$
- male card holder who is a pensioner who is also seperated (marital status): based on this model, the odds of him being a “bad” account, is $\widehat{odds} = -4.89511 + 4.15985 + 1.52579 = 0.79053$. The probability of her being a “bad” account is $\frac{e^{\widehat{odds}}}{1+e^{\widehat{odds}}} = 0.6879$.

```
# Make predictions for test data
step.probabilities.test <-
  step.glm %>% predict(card.test, type = "response")
step.predicted.classes.test <-
  as.factor(ifelse(step.probabilities.test > cpopt, 1, 0))
# Model confusion matrix
step.cm.test <-
  confusionMatrix(step.predicted.classes.test,card.test$label)
step.cm.test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7332   53
##          1 1324   25
##
##                Accuracy : 0.8423
##                  95% CI : (0.8345, 0.8499)
##     No Information Rate : 0.9911
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0185
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.84704
##             Specificity : 0.32051
##          Pos Pred Value : 0.99282
##          Neg Pred Value : 0.01853
##              Prevalence : 0.99107
##          Detection Rate : 0.83948
##    Detection Prevalence : 0.84555
##       Balanced Accuracy : 0.58378
##
##        'Positive' Class : 0
##
```

```
# plot ROC
## prepare an object for ROC curve
step.prediction <- prediction(step.probabilities.test, card.test$label)
step.roc.obj <- performance(step.prediction, measure="tpr", x.measure="fpr")
## plot ROC curve
plot(step.roc.obj, main="Stepwise: ROC curve on testing data")
```




```
#AUC Area under curve
step.auc <- performance(step.prediction, measure = "auc")@y.values
```

Because stepwise selection involves fitting one null model, along with p – k models in the kth iteration, whereas best subset selection performs all 2^k iterations. It did not produce the model with the least test error given there are 15 predictors ,also seen in the AUC of ROC. Therefore, best subset selection outperforms here.

As an alternative, we can fit a model containing all p predictors using a technique that constrains or regularizes the coefficient estimates to reduce the variance. Ridge and the Lasso regression can be considered.

Unlike best subset, forward stepwise, and backward stepwise selection, which will generally select models that involve just a subset of the variables, ridge regression will include all p predictors in the final model.

This may not be a problem for prediction accuracy, but it can create a challenge in model interpretation in settings in which the number of variables p is quite large. In the Credit data set, it appears that the most important variables are GENDER and FLAG_OWN_CAR in previous best subset analysis. So we might wish to build a model including just these predictors.

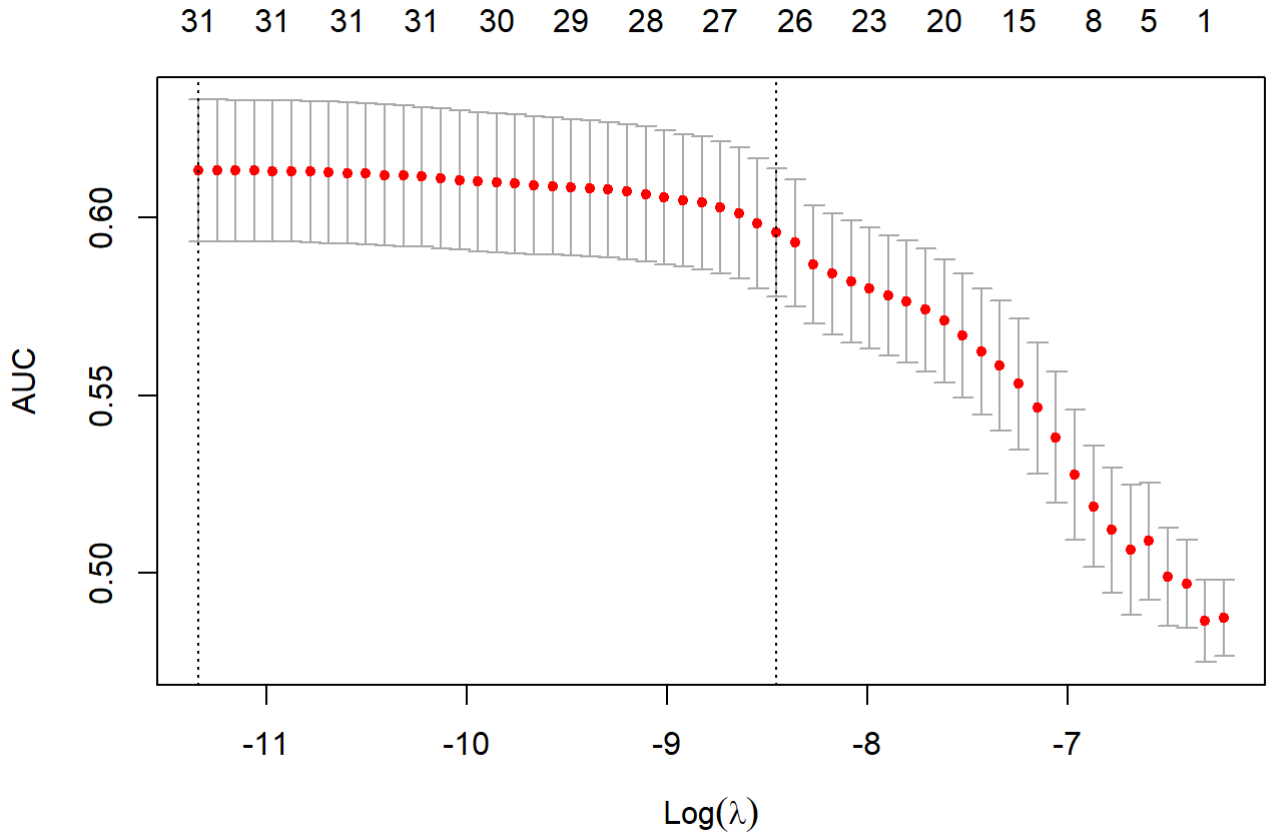
However, ridge regression will always generate a model involving all ten predictors. Increasing the value of λ will tend to reduce the magnitudes of the coefficients, but will not result in exclusion of any of the variables. Hence, we conducted the lasso as a alternative to ridge regression that overcomes this disadvantage by using l1 penalty (penalizing the sum of absolute values of the coefficients).

(4) Lasso regression

Lasso regression puts constraints on the size of the coefficients associated to each variable. However, this value will depend on the magnitude of each variable. It is therefore necessary to standardize the variables to improve interpretability of coefficients. Also, it grants us the ability to rank the coefficient importance by the relative magnitude of post-shrinkage coefficient estimates. The result of centering the variables means that there is no longer an intercept.

In lasso, as λ increases, As the model is becoming more and more flexible the test RSS will reduce first and then start increasing when overfitting will start. Variance steadily increase with the increase in model flexibility. Therefore, we would like to use cross-validation on training data to find the lambda that minimizes prediction error on the during cross-validation.

```
set.seed(1234)
#bind the train and test data in order for creating matrix later
traintest=rbind(card.train,card.test)

#remove label from predicting variables [6]
X = sparse.model.matrix(label~.-label,
                        data = traintest)


# Find the best lambda using cross-validation
final_cv =
    cv.glmnet(X[1:nrow(card.train),], ##distill the training set from the combined ma
trix
              card.train[,16],
              family = "binomial",
              type.measure ="auc",
              nfolds = 5)
# Plot cross-validation error according to the log of lambda
plot(final_cv)
```



The plot displays the cross-validation error according to the log of lambda. The left dashed vertical line indicates that the log of the optimal value of log(lambda) is approximately -11, which is the one that minimizes the prediction error. This lambda value will give the most accurate model. The exact value of lambda can be viewed as follow:

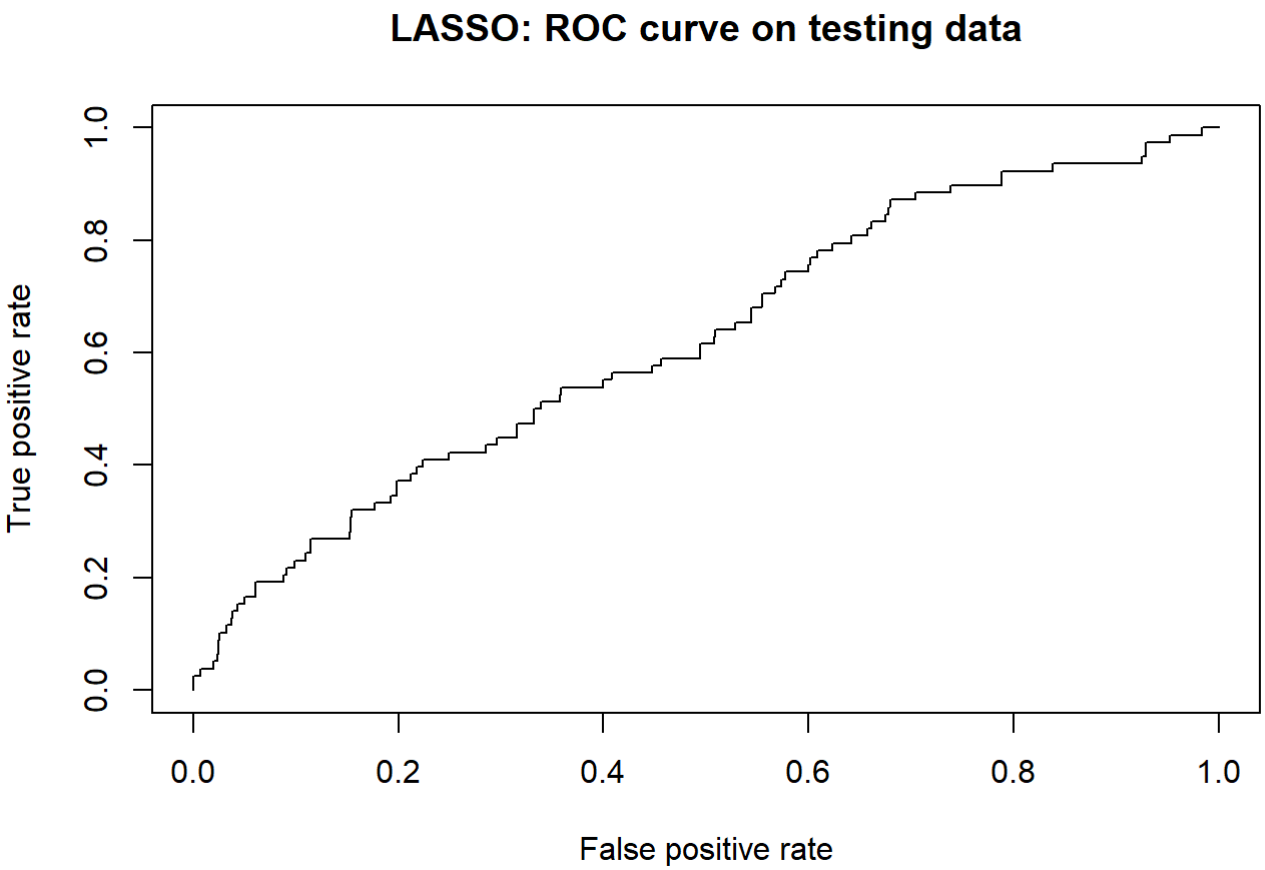
```
final_cv$lambda.min
```

```
## [1] 1.188614e-05
```

```
#predict on test set
lasso.pred = predict(final_cv, s='lambda.min',#
                     newx=X[-(1:nrow(card.train)),],
                     type="response")

#Model confusion matrix
lasso.pred.class <- as.factor(ifelse(lasso.pred >  cpopt, 1, 0))
lasso.cm <- confusionMatrix(lasso.pred.class,card.test$label)
lasso.cm[["table"]]
```

```
##            Reference
## Prediction    0    1
##          0 7271   53
##          1 1385   25
```

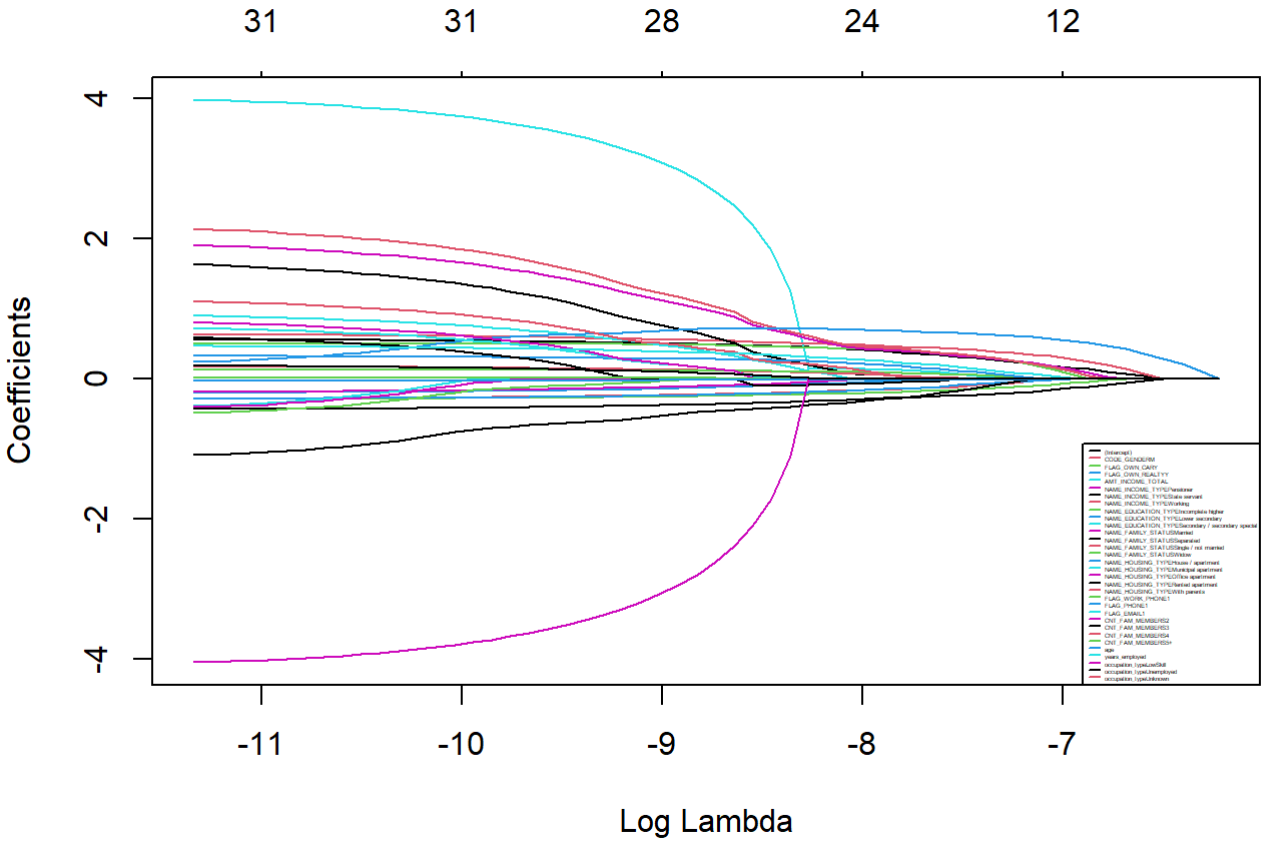
```
# plot ROC
## prepare an object for ROC curve
lasso.prediction <- prediction(lasso.pred, card.test$label)
lasso.roc.obj <- performance(lasso.prediction, measure="tpr", x.measure="fpr")
## plot ROC curve
plot(lasso.roc.obj, main="LASSO: ROC curve on testing data")
```




```
#AUC Area under curve
lasso.auc <- performance(lasso.prediction, measure="auc")@y.values
lasso.auc
```

```
## [[1]]
## [1] 0.6226421
```

```
plot(final_cv$glmnet.fit, xvar = "lambda")
legend("bottomright", lwd = 1, col = 1:6, legend = colnames(X), cex = .19)
```




```
##Using lambda.min as the best lambda, regression coefficients as follows:
```

We observe that at first, the lasso results in a model that contains only the GENDER predictor. Then FLAG_OWN_CAR and FLAG_OWN_REALTY enter the model almost simultaneously, shortly followed by AMT_INCOME_TOTAL. Eventually, the remaining variables enter the model.

```
coef(final_cv)
```

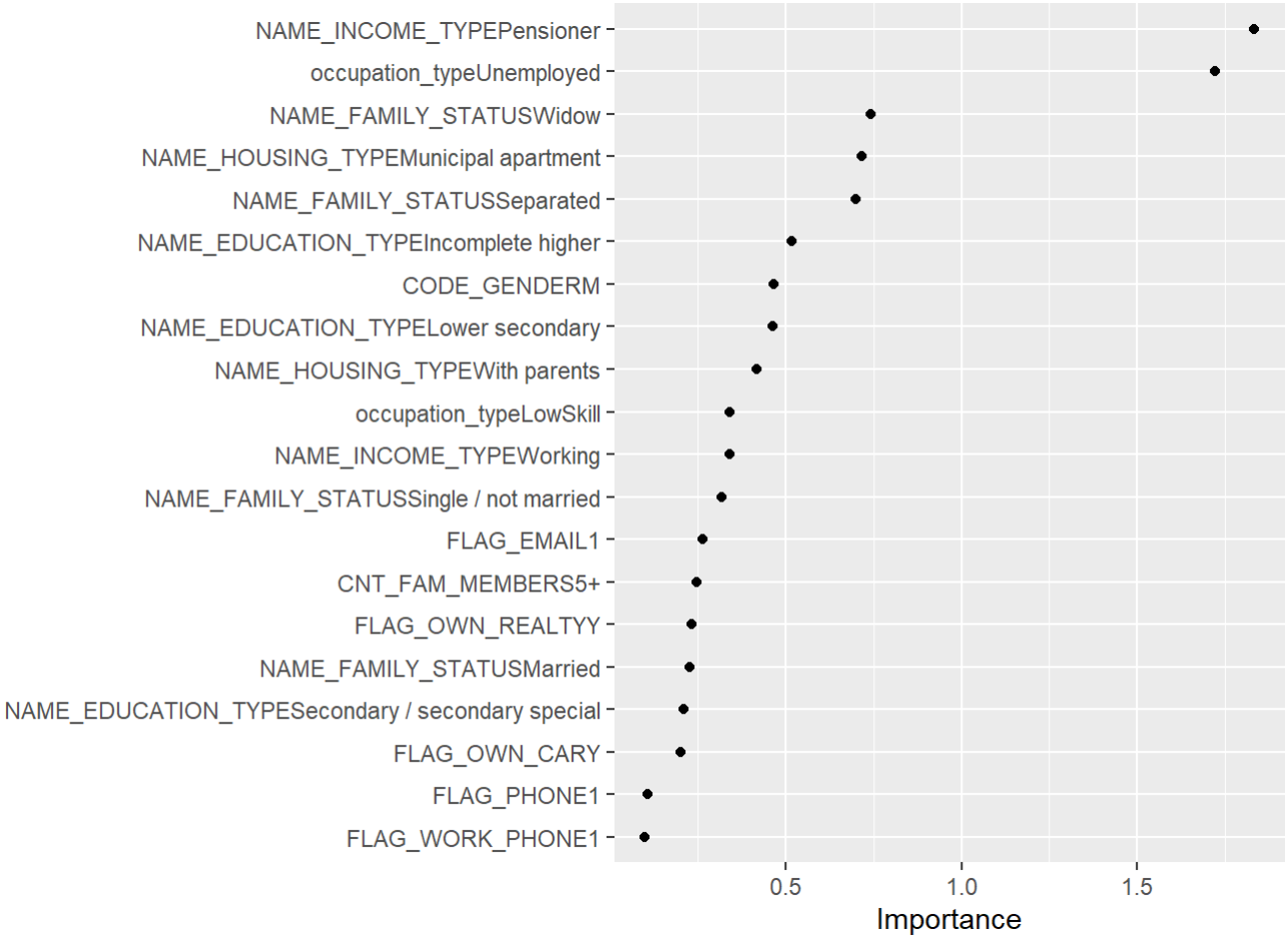
```
## 33 x 1 sparse Matrix of class "dgCMatrix"
##                                                          1
## (Intercept)                                   -3.924444298
## (Intercept)                                    .
## CODE_GENDERM                                   0.465824435
## FLAG_OWN_CARY                                 -0.202269470
## FLAG_OWN_REALTYY                              -0.232604689
## AMT_INCOME_TOTAL                              -0.095830616
## NAME_INCOME_TYPEPensioner                      1.833027925
## NAME_INCOME_TYPEState servant                 -0.072212103
## NAME_INCOME_TYPEWorking                       -0.340473652
## NAME_EDUCATION_TYPEIncomplete higher           0.518440890
## NAME_EDUCATION_TYPELower secondary             0.462243151
## NAME_EDUCATION_TYPESecondary / secondary special -0.209420119
## NAME_FAMILY_STATUSMarried                      0.227324612
## NAME_FAMILY_STATUSSeparated                    0.700620662
## NAME_FAMILY_STATUSSingle / not married         0.318974114
## NAME_FAMILY_STATUSWidow                        0.741840624
## NAME_HOUSING_TYPEHouse / apartment             .
## NAME_HOUSING_TYPEMunicipal apartment           0.716565100
## NAME_HOUSING_TYPEOffice apartment              .
## NAME_HOUSING_TYPERented apartment              .
## NAME_HOUSING_TYPEWith parents                 -0.416849193
## FLAG_WORK_PHONE1                               0.099297674
## FLAG_PHONE1                                    0.107725044
## FLAG_EMAIL1                                    0.263661797
## CNT_FAM_MEMBERS2                               0.009697828
## CNT_FAM_MEMBERS3                               .
## CNT_FAM_MEMBERS4                              -0.096772625
## CNT_FAM_MEMBERS5+                              0.246917324
## age                                            0.003781317
## years_employed                                -0.018608342
## occupation_typeLowSkill                        0.341586631
## occupation_typeUnemployed                     -1.720064050
## occupation_typeUnknown                         0.040948016
```

The lasso regression did not set any variable to zero, except for several levels within categorical variables. This makes sense as it justifies the tedious procedure for banks to collect applicants' comprehensive information before issuing them credit cards.

Within housing type where some levels are set to zero,we can infer that features like living in the municipal apartment will increase risk of defaulting. One possibility might be that municipal apartment are affordable housing provided by the government, so residents may have relatively low income stability and thus higher risk of incurring bad debt. Holding all else equal, banks should pay special attention to such applicants compared to residents in else where.

Variable importance for regularized models provides a similar interpretation as in logistic regression. Importance is determined by magnitude of the standardized coefficients. [8]

```
#Top 20 most important variables for the optimal regularized regression model.
vip(final_cv, num_features = 20, geom = "point")
```



The relationship between the predictor variables and response is monotonic linear. For relationships that are positive in nature (e.g. age), as the values in these features increase the average predicted probalibity of defaulting. For relationships that are negative in nature (e.g. years employed), as the values in these features increase the average predicted probalibity of defaulting.

2. Support Vector Machine

After logistic regression models, we move on to explore SVM with the radial kernel (Runtime: 1 hr) [9]

Dropping the normality assumption, we now use a more algorithmic approach to the model. Kernel SVM find a hyperplane in the embedded feature space that separates the two class in the response variable, with error margin constrained by the cost.

In this case, the radial kernel was chosen due to the algorithm being computationally intensive.

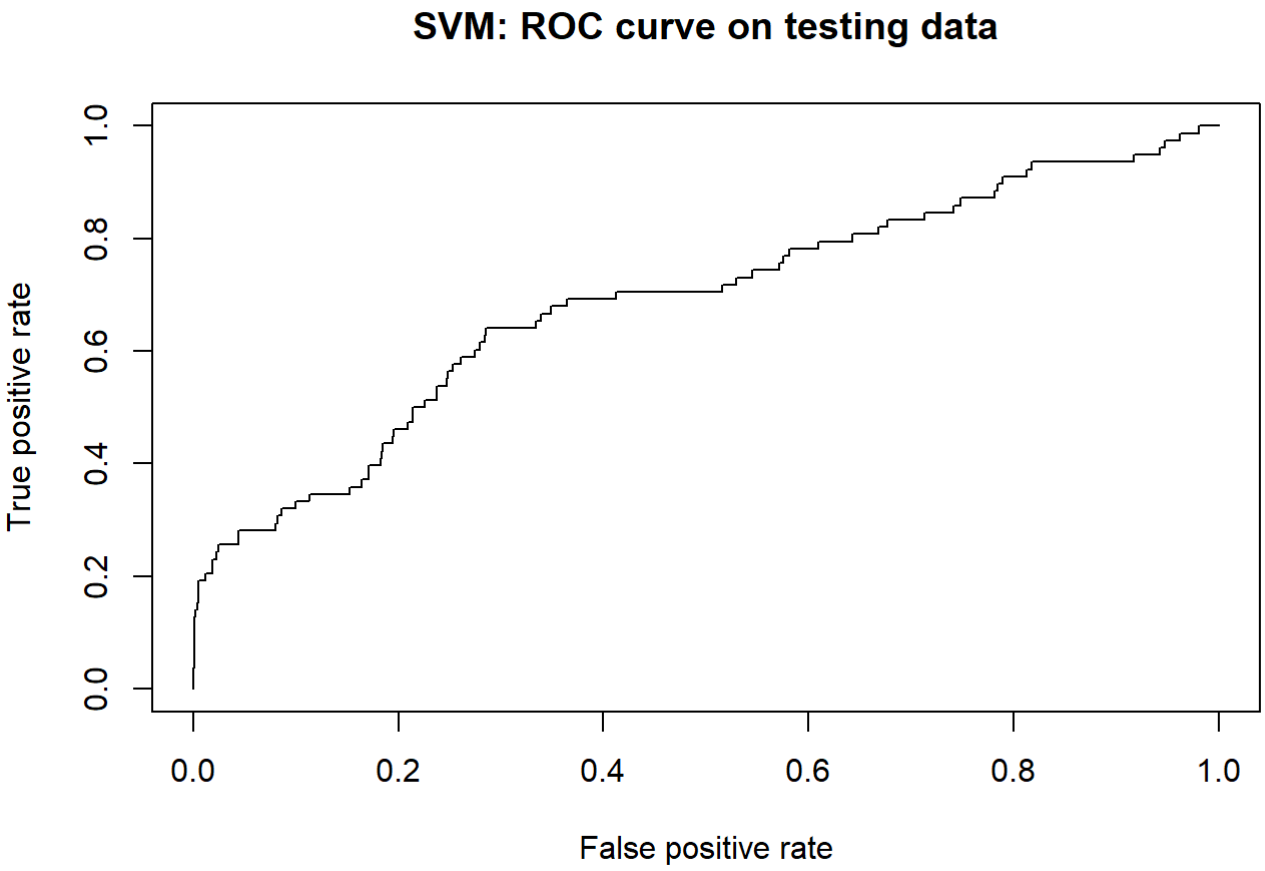
(1) Radial Kernel

```
# set recipe
cd.recipe <- training(card.split) %>%
  recipe(label ~ .) %>%
  prep()

# set model, cost and sigma to be selected through cross-validation
cd.svmrbf.mod <-svm_rbf(cost = tune(), rbf_sigma = tune()) %>%
  set_mode("classification") %>%
  set_engine("kernlab")

# set svm workflow
cd.workflow.svmrbf <- workflow() %>%
  add_recipe(cd.recipe) %>%
  add_model(cd.svmrbf.mod)

# set tune grid
svm_grid <- grid_regular(cost(),
                         rbf_sigma(),
                         levels = 5)

# tune svm
set.seed(612)
cd.fit.svmrbf <- tune_grid(cd.workflow.svmrbf,
                           resamples=card.fold,
                           grid=svm_grid,
                           metrics=metric_set(roc_auc))

# combination of Cost and Sigma that produces the lowest auc
cd.svmrbf.tune.param <- cd.fit.svmrbf %>% select_best()
cd.workflow.svmrbf.final <- finalize_workflow(cd.workflow.svmrbf,
                                              cd.svmrbf.tune.param)

# fit final model to train data and evaluate on test data
cd.svmrbf.eval <- cd.workflow.svmrbf.final %>%
  last_fit(split=card.split)

# AUC value for svm
svmrbf.metrics <- cd.svmrbf.eval %>%
  collect_metrics()
svmrbf.metrics
```

.metric <chr>	.estimator <chr>	.estimate <dbl>	.config <chr>
accuracy	binary	0.9902679	Preprocessor1_Model1
roc_auc	binary	0.6872512	Preprocessor1_Model1

2 rows

```
# plot ROC
## get the probabilities from svmrbf model
svmrbf.predict <- cd.svmrbf.eval %>% collect_predictions()
## prepare an object for ROC curve
svmrbf.prediction <- prediction(svmrbf.predict$.pred_1, card.test$label)
svmrbf.roc.obj <- performance(svmrbf.prediction, measure="tpr", x.measure="fpr")
## plot ROC curve
plot(svmrbf.roc.obj, main="SVM: ROC curve on testing data")
```




```
# confusion matrix
svmrbf.cm <- cd.svmrbf.eval %>%
  collect_predictions() %>%
  dplyr::select(.pred_class, label) %>%
  rename(predicted_label = .pred_class, actual_label = label) %>%
  table()
rownames(svmrbf.cm) <- c("Good", "Bad")
colnames(svmrbf.cm) <- c("Good", "Bad")
svmrbf.cm
```

```
##                 actual_label
## predicted_label Good  Bad
##            Good 8639   68
##            Bad    17   10
```

```
#Refitting the tuned model to the training set
svm.pulled <- cd.workflow.svmrbf.final %>% fit(card.train)

#View the model
svm.pulled
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: svm_rbf()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 32
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  1
##
## Number of Support Vectors : 2268
##
## Objective Function Value : -8985.184
## Training error : 0.00519
## Probability model included.
```

The hyperplane which is calculated using the RBF (Radial Basis Function) is: $\exp(-\gamma \sum_j (X_j - X_j')^2)$, with X_j be the j's feature of the X observation, and X_j' is one of the support vector's j's feature. The number of Support Vectors are 2268, so when new test data comes, each observations are combined with one of the support vectors in the RBF, and the inner product is calculated. Taking a linear combination of all inner products with an additional intercept term, with the coefficients and the intercept are fitted from the training data, will provide a score for each test observation. The more positive the score is, the more likely the test account will be classified as a "bad" account, and vise versa.

```
#pulling support vectors from SVM model
support_vector<-card.train[svm.pulled[["fit"]][["fit"]][["fit"]]@SVindex,]

#221 of bad account has been used as support vector
summary(support_vector$label)
```

```
##    0    1
## 2051  217
```

Within the 2268 support vectors, most of the support vectors are "good" accounts, although almost 72% of "bad" accounts end up being a support vector. This is because due to Class Imbalance, the hyperplane is "closer" to the "bad" accounts, conversely, the majority of the "good" accounts are "far" from the hyperplane. This also creates a misleadingly excellent accuracy on the SVM predictions, which once again is an indicator of why AUC should be used in this context.

Note: The score given to each observation almost behaves like a credit score, and could have the potential to be an indicator for the credit analyst to assess whether an account will default or not. However, given the enormous amount of customer data within a Chinese commercial bank (700 million credit card issued in total for a major bank), the computational time might make SVM model unrealistic for practical application.

3. Tree-Based Methods

Lastly, we use tree-based modeling for the classification of “good”/“bad” accounts. In this section, we build a decision tree and move on to improve it through bagging and random forest.

(1) Decision Tree

The decision tree model uses recursive binary splitting to segment predictor space. It has the advantage of being easily interpretable. We thus started with a single decision tree.

In order to optimize the AUC value, we used cross-validation to tune for the depth of tree and cost complexity.

```
# set recipe
# (same as for SVM, repeated to facilitate parallel knitting)
cd.recipe <- training(card.split) %>%
  recipe(label ~ .) %>%
  prep()

# set tree model, leave tree_depth and cost_complexity for tuning
cd.tree <- decision_tree(tree_depth = tune(), cost_complexity = tune()) %>%
  set_engine("rpart") %>%
  set_mode("classification")

# set workflow
cd.workflow.tree <- workflow() %>%
  add_recipe(cd.recipe) %>%
  add_model(cd.tree)

# model tuning
## set tune grid auto-generate 5 sensible values for each
tree.grid <- grid_regular(tree_depth(), cost_complexity(), levels=5)
set.seed(130)
## use cross-validation to tune
cd.fit.tree <- tune_grid(cd.workflow.tree, resamples=card.fold,
                         grid=tree.grid, metrics=metric_set(roc_auc))
## select the best combination
cd.tree.tune.param <- cd.fit.tree %>% select_best()
## finalise workflow tree
cd.workflow.tree.final <- finalize_workflow(cd.workflow.tree, cd.tree.tune.param)
```

From cross-validation, we reached the best combination of cost complexity and tree depth as follows,

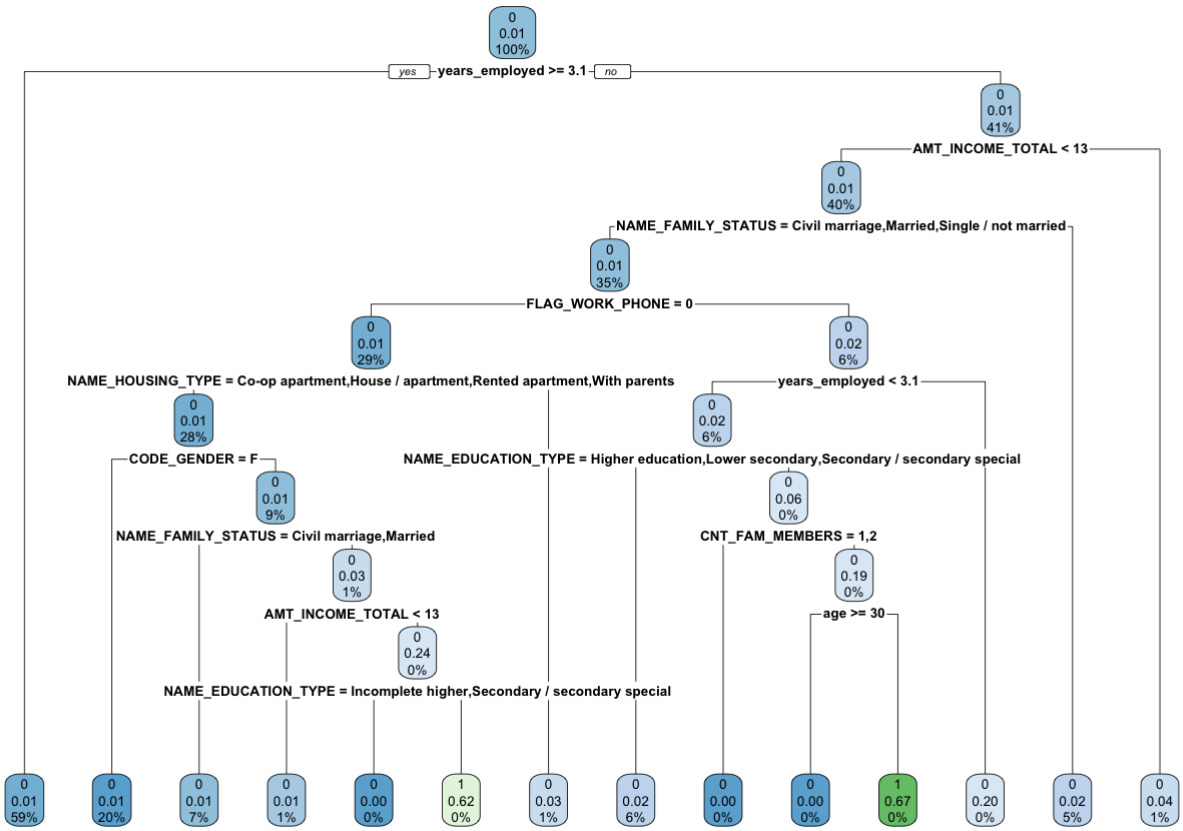
```
cd.tree.tune.param
```

cost_complexity	tree_depth	.config
<dbl>	<int>	<chr>
1e-10	11	Preprocessor1_Model04

1 row

This results in a decision tree as follows,

```
tree.fit <- cd.workflow.tree.final %>%
  fit(card.train) %>%
  pull_workflow_fit()
rpart.plot(tree.fit$fit, roundint=FALSE)
```

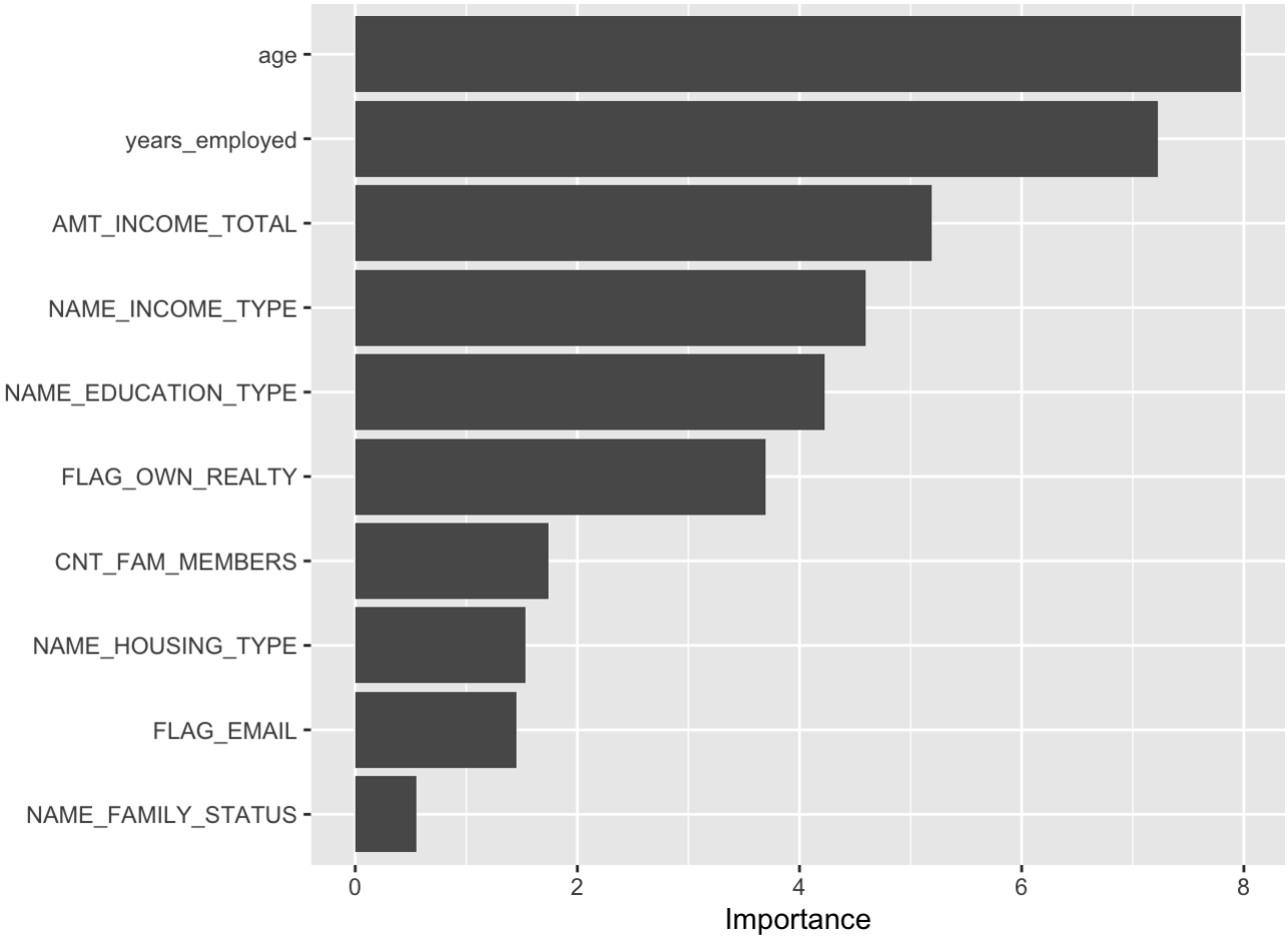


Based on the tree model, we see that years_employed is a crucial indicator on whether a person is likely to be a “bad” account. More specifically, those who have an employment history of more than 3.1 years will be classified as “good”. This is consistent with our results in the baseline model which suggests that longer employment history is associated with lower odds of being “bad”.

The second split takes place at AMT_INCOME_TOTAL. When the value is greater than 13, an individual is classified as “good”. This is understandable as an individual with little income may have less capability to repay credit card loans.

Other variables used in the tree include NAME_FAMILY_STATUS, FLAG_WORK_PHONE, NAME_HOUSING_TYPE, CODE_GENDER, NAME_EDUCATION_TYPE, CNT_FAMILY_MEMBERS, and age. We use variable importance to measure the contribution of variables to the model. This value is calculated as the sum of the error decrease when split the tree by a variable [10].

```
# check variables that are important in the tree [11]:
tree.vip <- cd.workflow.tree.final %>%
  fit(data=card.train) %>% # fit model on training data
  pull_workflow_fit() %>%
  vip()
tree.vip
```



We could see that age makes the most contribution to the decision tree model followed by years_employed.

We move on to assess the performance of this tree on testing data.

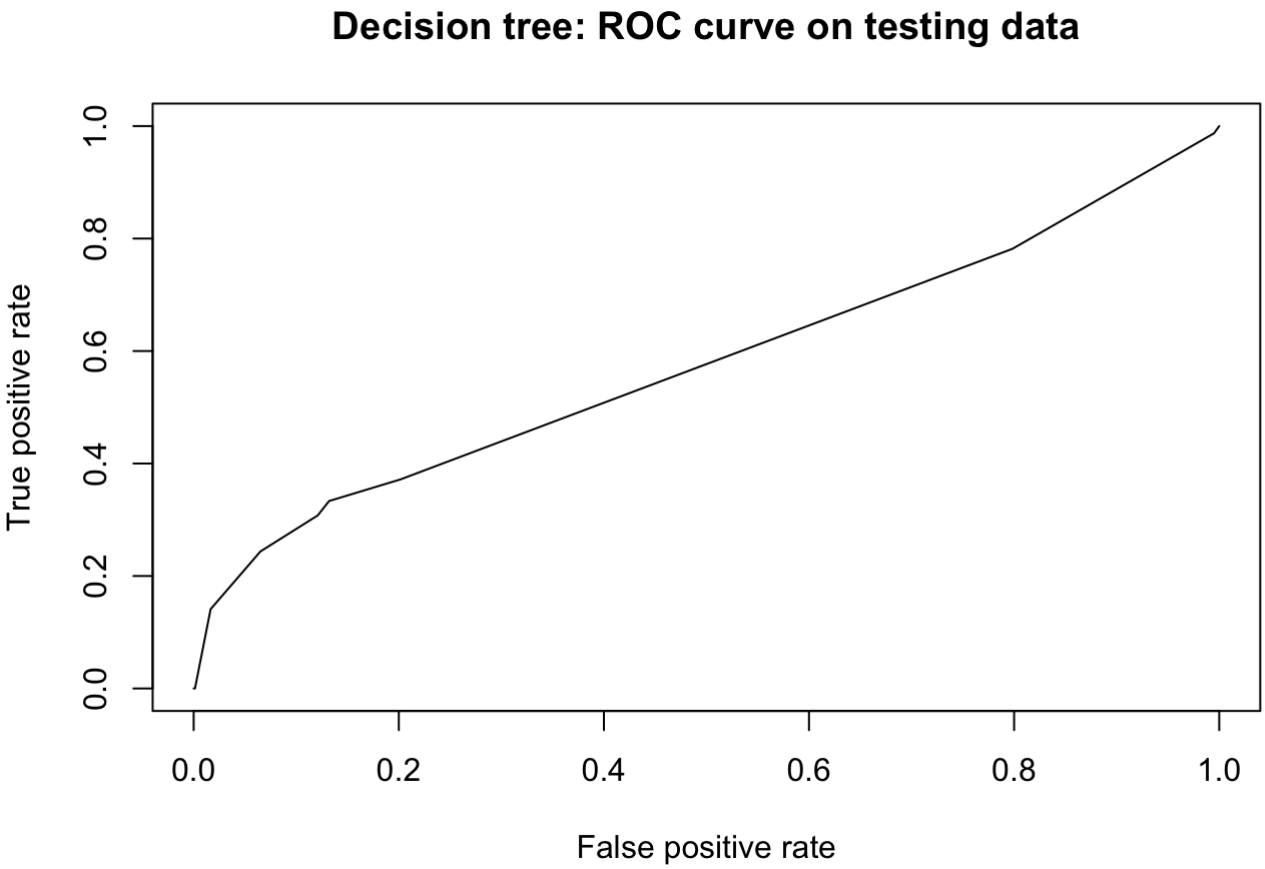
```
# check performance
## fit final model on training data and evaluate performance on test data
cd.tree.eval <- cd.workflow.tree.final %>%
  last_fit(card.split)
## AUC value for decision tree
tree.metrics <- cd.tree.eval %>%
  collect_metrics()
tree.metrics
```

.metric	.estimator	.estimate	.config
<chr>	<chr>	<dbl>	<chr>
accuracy	binary	0.9899244	Preprocessor1_Model1
roc_auc	binary	0.5769979	Preprocessor1_Model1

2 rows

We can observe that the decision tree makes a small improvement in AUC from the baseline model (0.5692).

```
# plot ROC
## get the probabilities from tree model
tree.predict <- cd.tree.eval %>% collect_predictions()
## prepare an object for ROC curve
tree.prediction <- prediction(tree.predict$.pred_1, card.test$label)
tree.roc.obj <- performance(tree.prediction, measure="tpr", x.measure="fpr")
## plot ROC curve
plot(tree.roc.obj, main="Decision tree: ROC curve on testing data")
```




```
# confusion matrix
tree.cm <- cd.tree.eval %>%
  collect_predictions() %>%
  dplyr::select(.pred_class, label) %>%
  rename(predicted = .pred_class, actual = label) %>%
  table()
rownames(tree.cm) <- c("Good", "Bad")
colnames(tree.cm) <- c("Good", "Bad")
tree.cm
```

```
##           actual
## predicted Good  Bad
##      Good 8646   78
##      Bad    10    0
```

Unfortunately, no actual “bad” accounts are picked up by this model. Using this model in real life would result in significant financial loss.

(2) Bagging

To improve from the single decision tree model, we use bagging to reduce the variance. Bagging creates trees based on different subsamples of a given dataset (random selection with replacement). When selecting a split point, all variables can be looked at. Then the trees formed on each sample are aggregated to return a classification.

For the bagging model, we set the depth of the tree to be 11, the tuned parameter from single decision tree model. We leave cost complexity for tuning.

```
# set bag model, leave cost_complexity for tuning
cd.bag <- bag_tree(tree_depth = 11, cost_complexity = tune("C")) %>%
  set_engine("rpart", times=5) %>%
  set_mode("classification")

# set workflow
cd.workflow.bag <- workflow() %>%
  add_recipe(cd.recipe) %>%
  add_model(cd.bag)

# model tuning
set.seed(280)
# grid is set after several trials
cd.fit.bag <- tune_grid(cd.workflow.bag,
                        grid=data.frame(C=2^(-13:-8)), resamples=card.fold, metrics=m
etric_set(roc_auc))

## select the best combination
cd.bag.tune.param <- cd.fit.bag %>% select_best()
## finalise workflow bag
cd.workflow.bag.final <- finalize_workflow(cd.workflow.bag, cd.bag.tune.param)
```

From tuning, we reach the best cost complexity as follows,

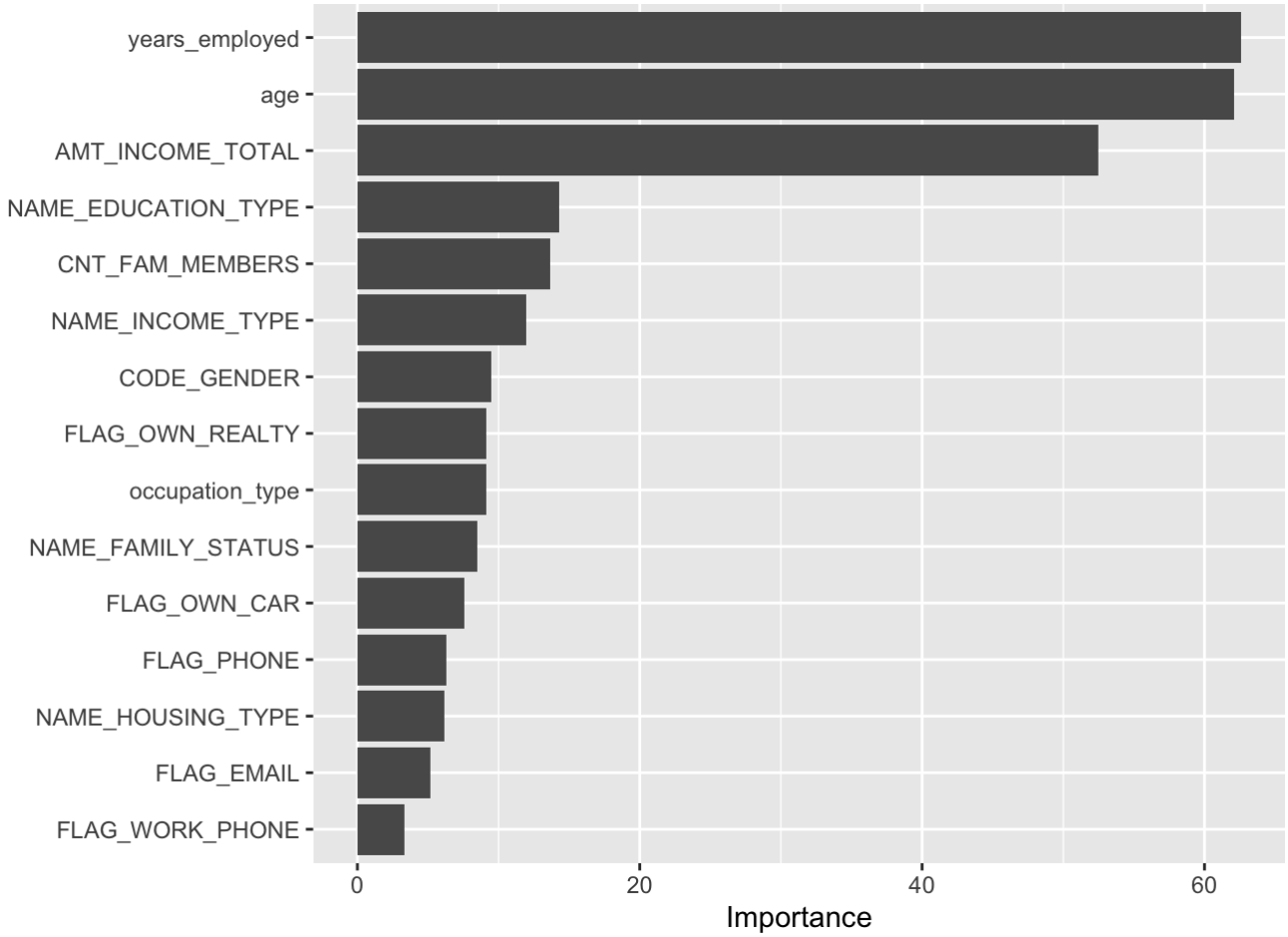
```
cd.bag.tune.param
```

C	.config
<dbl>	<chr>
0.0009765625	Preprocessor1_Model4

1 row

After fitting the bagged tree model, we can check the variable importance.

```
# check variables that are important
bag.vip.bagger <- cd.workflow.bag.final %>%
  fit(data=card.train) %>% # fit model on training data
  pull_workflow_fit()
# extract variable importance from _bagger class
bag.vip <- bag.vip.bagger$fit$imp %>%
  # user reorder to ensure the descending order of importance
  ggplot(aes(y=reorder(term, value), x=value)) + geom_col() +
  ylab("") + xlab("Importance")
bag.vip
```



The top three variables in terms of importance are years_employed, age, and AMT_INCOME_TOTAL, consistent with the results from single decision tree, although years_employed is second after age in tree. We can also observe that the importance values for other variables are significantly smaller.

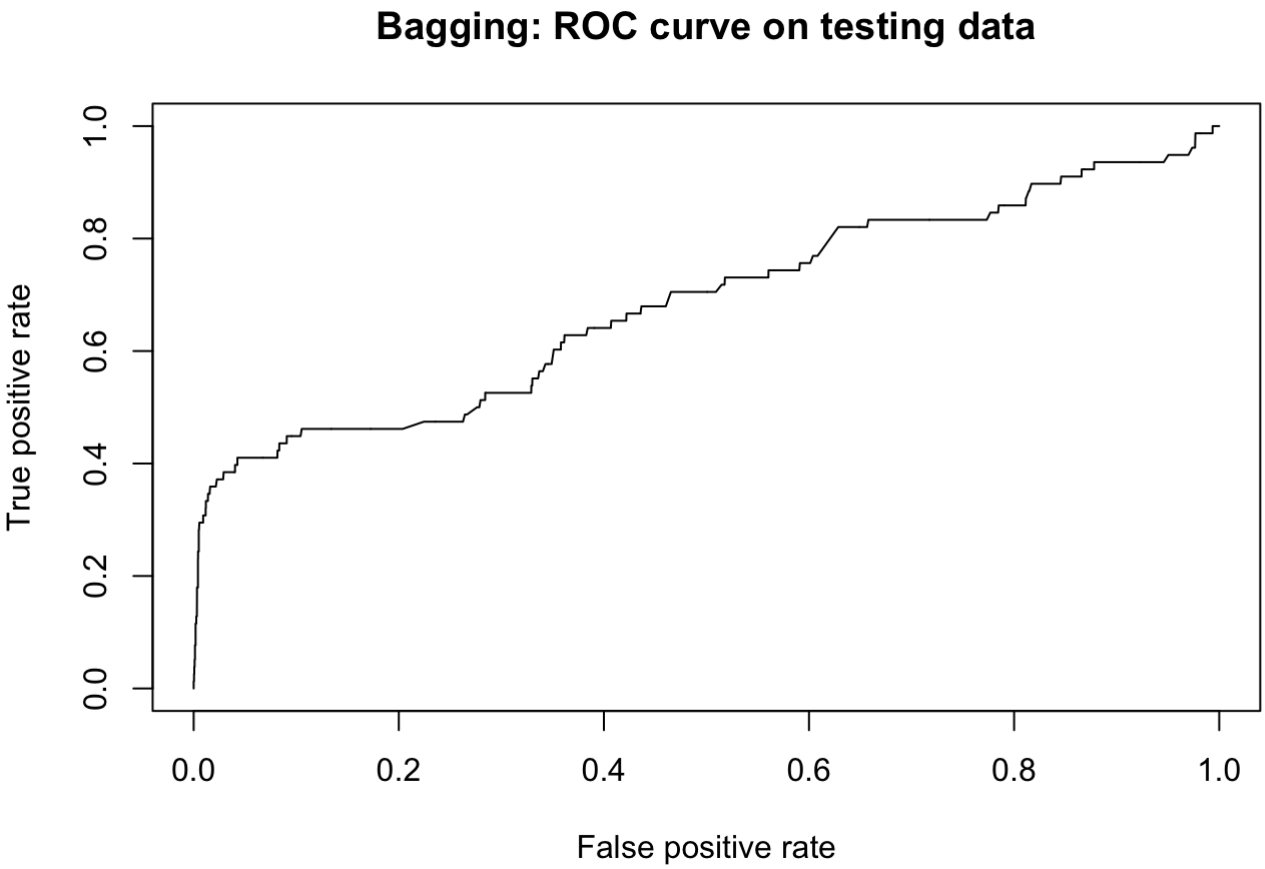
Now we check the performance of this model on testing data.

```
# check performance
## fit final model on training data and evaluate performance on test data
cd.bag.eval <- cd.workflow.bag.final %>%
  last_fit(card.split)
## AUC value for decision bag
bag.metrics <- cd.bag.eval %>%
  collect_metrics()
bag.metrics
```

.metric	.estimator	.estimate	.config
<chr>	<chr>	<dbl>	<chr>
accuracy	binary	0.9906114	Preprocessor1_Model1
roc_auc	binary	0.6841231	Preprocessor1_Model1

2 rows

```
# plot ROC
## get the probabilities from bag model
bag.predict <- cd.bag.eval %>% collect_predictions()
## prepare an object for ROC curve
bag.prediction <- prediction(bag.predict$.pred_1, card.test$label)
bag.roc.obj <- performance(bag.prediction, measure="tpr", x.measure="fpr")
## plot ROC curve
plot(bag.roc.obj, main="Bagging: ROC curve on testing data")
```




```
# confusion matrix
bag.cm <- cd.bag.eval %>%
  collect_predictions() %>%
  dplyr::select(.pred_class, label) %>%
  rename(predicted_label = .pred_class, actual_label = label) %>%
  table()
rownames(bag.cm) <- c("Good", "Bad")
colnames(bag.cm) <- c("Good", "Bad")
bag.cm
```

```
##                actual_label
## predicted_label Good  Bad
##            Good 8648   74
##            Bad     8    4
```

For bagging, we see that the AUC value is much higher than the decision tree (0.577) and the baseline model (0.5692). It also correctly identifies four bad accounts, an improvement from decision tree.

(3) Random Forest

Lastly, we try to add more variability in aggregated trees and improve from the bagged decision trees. To do this, we use random forest. Now, when selecting a split point, only a (random) sample of, instead of all, predictors can be looked at. This reduces collinearity between trees and increases information gain.

To determine the number of predictors to looked at each time, we use cross-validation to select a value.

```
# set model, no. of predictor to be selected through cross-validation
cd.rf <- rand_forest(trees=100, mtry=tune()) %>%
  set_engine("randomForest") %>%
  set_mode("classification")

# set random forest workflow
cd.workflow.rf <- workflow() %>%
  add_recipe(cd.recipe) %>%
  add_model(cd.rf)

# tune random forest
set.seed(500)
cd.fit.rf <- tune_grid(cd.workflow.rf, resamples=card.fold, metrics=metric_set(roc_au
c))
cd.rf.tune.param <- cd.fit.rf %>% select_best()
cd.workflow.rf.final <- finalize_workflow(cd.workflow.rf, cd.rf.tune.param)
```

The number of predictors is thus set as follows,

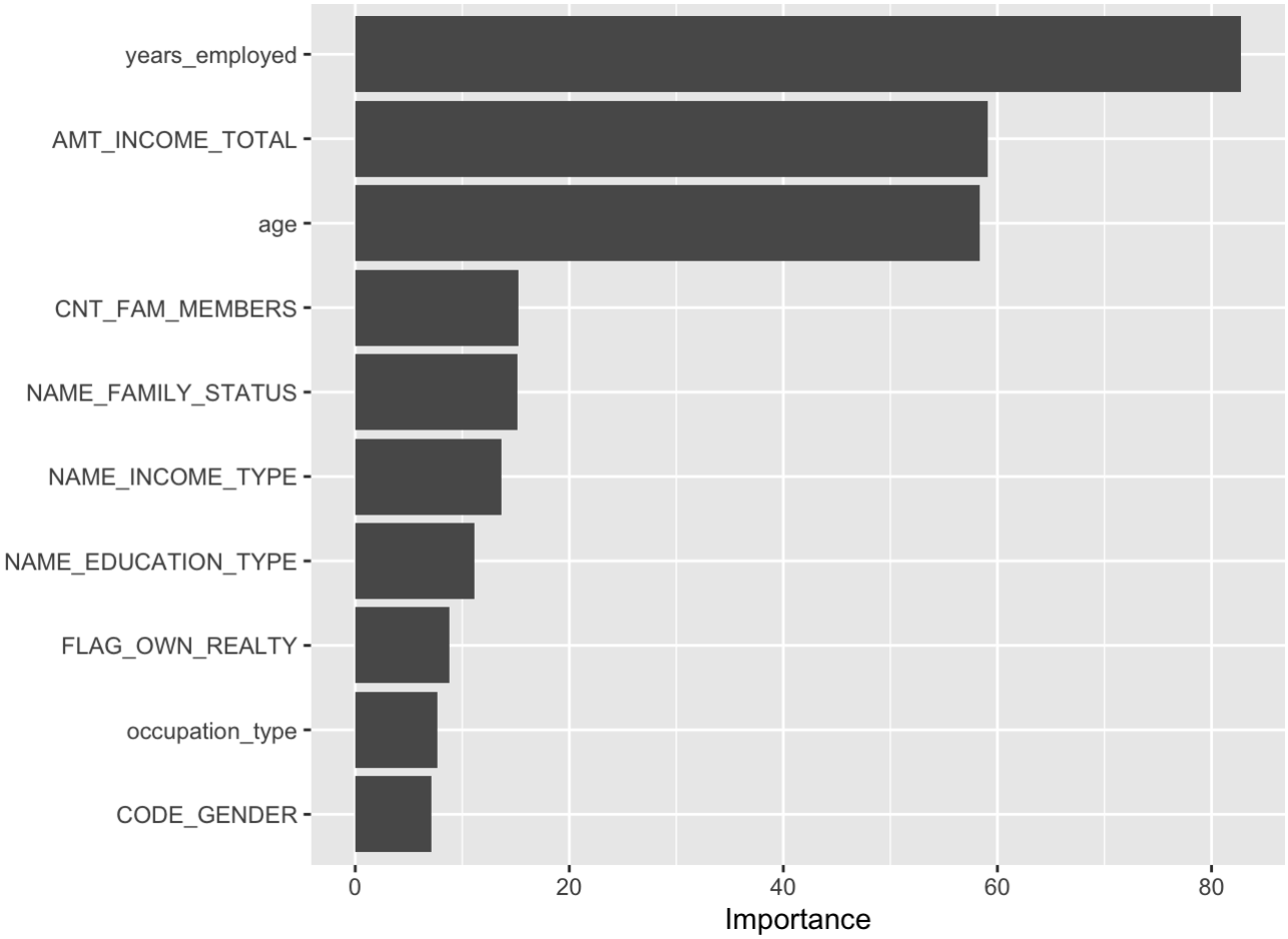
```
cd.rf.tune.param
```

mtry	.config
<int>	<chr>
11	Preprocessor1_Model2

1 row

The importance of variables is as follows,

```
# important variables
rf.vip <- cd.workflow.rf.final %>%
  fit(data=card.train) %>%
  pull_workflow_fit() %>%
  vip()
rf.vip
```



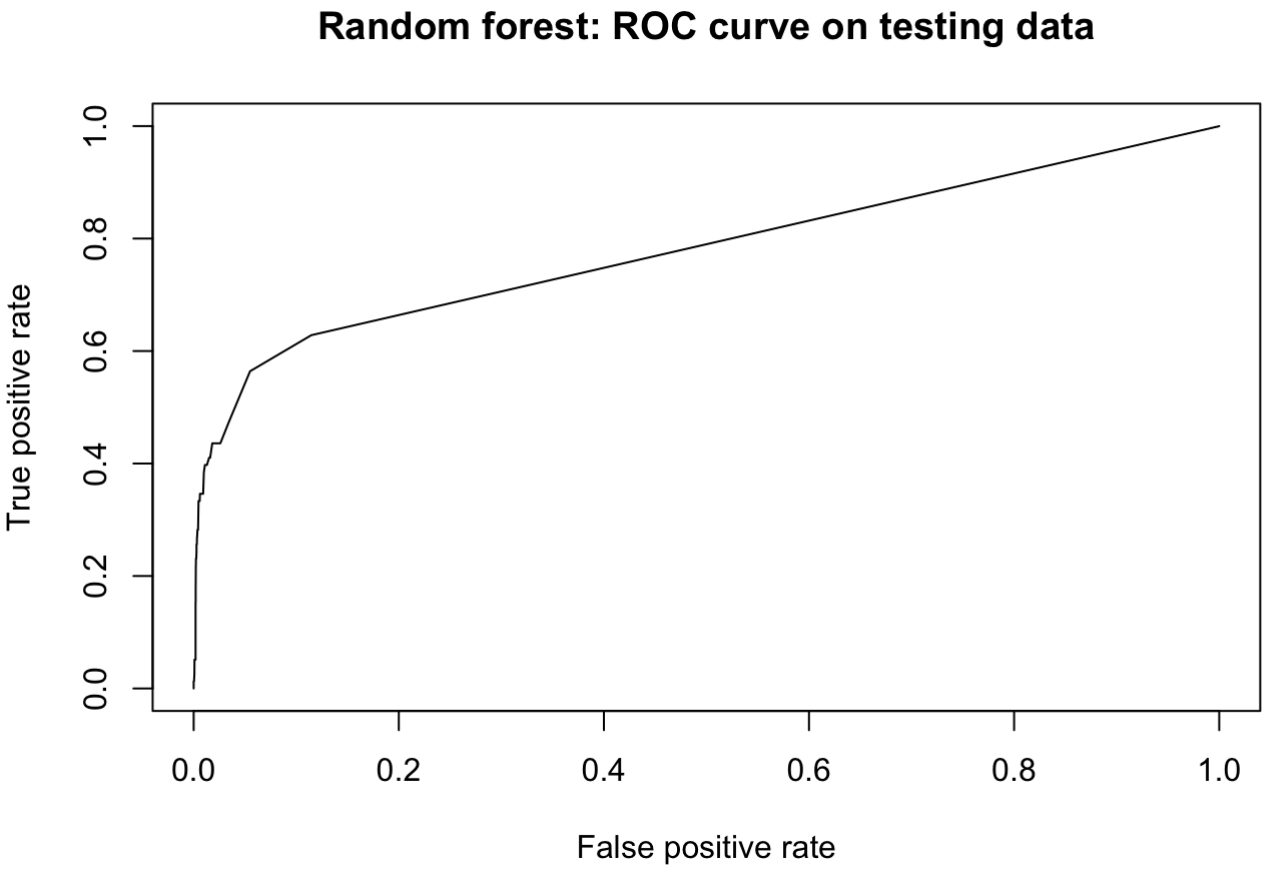
Again, the top three variables in terms of importance (i.e. contribution to the model) are years_employed, age and AMT_INCOME_TOTAL, consistent with the results from bagged decision tree. Similar to the case of bagging, the importance values for other variables are significantly smaller.

```
# fit final model to train data and evaluate on test data
cd.rf.eval <- cd.workflow.rf.final %>%
  last_fit(split=card.split)

# AUC value for random forest
rf.metrics <- cd.rf.eval %>%
  collect_metrics()
rf.metrics
```

.metric	.estimator	.estimate	.config
<chr>	<chr>	<dbl>	<chr>
accuracy	binary	0.9904969	Preprocessor1_Model1
roc_auc	binary	0.7802066	Preprocessor1_Model1

2 rows

```
# plot ROC
## get the probabilities from rf model
rf.predict <- cd.rf.eval %>% collect_predictions()
## prepare an object for ROC curve
rf.prediction <- prediction(rf.predict$.pred_1, card.test$label)
rf.roc.obj <- performance(rf.prediction, measure="tpr", x.measure="fpr")
## plot ROC curve
plot(rf.roc.obj, main="Random forest: ROC curve on testing data")
```




```
# confusion matrix
rf.cm <- cd.rf.eval %>%
  collect_predictions() %>%
  dplyr::select(.pred_class, label) %>%
  rename(predicted_label = .pred_class, actual_label = label) %>%
  table()
rownames(rf.cm) <- c("Good", "Bad")
colnames(rf.cm) <- c("Good", "Bad")
rf.cm
```

```
##                actual_label
## predicted_label Good  Bad
##            Good 8639   66
##            Bad    17   12
```

Lastly, we check the performance of the random forest model on testing data. The AUC value for random forest is the highest so far among all models. The confusion matrix also suggests a significant improvement from the baseline logistic model with two predictors: random forest picks up almost all of the “good” accounts

correctly; it also correctly picks up 12 “bad” accounts.

(4) Tree: Summary

At the end of the tree-based method section, we make a summary of the three models.

```
# combine AUC values for three methods
auc_sum <- list(tree.metrics[2,], bag.metrics[2,], rf.metrics[2,]) %>%
  map_dfr(bind_rows) %>%
  dplyr::select(AUC=.estimate)
# display performance metrics of all three methods
all_tree_metrics <- cbind(method=c("Tree", "Bagging", "Random Forest"), auc_sum)
all_tree_metrics
```

method <chr>	**AUC** <dbl>
Tree	0.5769979
Bagging	0.6841231
Random Forest	0.7802066

3 rows

The random forest model has the highest AUC value out of the three tree models in our case. Indeed, it leverages the power of several decision trees and it reduces the collinearity between trees. However, it should be noted that although random forest model enjoys a high predictive accuracy, it is less interpretable.

4. Model Comparison

Given our data, Random Forest has the highest auc, beating the industrial standard practice, stepwise logistic regression, and Radial SVM and Lasso logistic regression coming in between.

We would like to evaluate the models against the stepwise logistic regression (banks’ practice) according to model complexity (computational time), predictive accuracy, and interpretability.

```
tab <- matrix(c("Slow- especially for large banks with more user data","Medium ","Fas
t","Medium- linear SVM is more interpretable than radial kernel SVM","Poor-black box,
not suitable for banks to be compliant","Good- classic logit model, which is suitable
for banks to interpret based on individual information collected","High","Very high",
"Moderately high","Poor- kernel includes all features","Good","Moderately good","Mode
rately-change categorical levels to several dummy variables","Extremely","Moderately-
change categorical levels to several dummy variables","No","No","Yes","Extremely Good
- only depend on borderline data","Good","Poor- may be affected by high leverage poin
ts or outliers, but not so much given the banks’ applicant number is tremendous"), nc
ol=3, byrow=TRUE)
colnames(tab) <- c("SVM",   "Random forest",    "Lasso")
rownames(tab) <- c("Computational speed","Interpretability" ,"Predictive accuracy" ,
"Dealing with irrelevant features","Natural handling categorical features","Standardi
zation required" ,"Robustness to outliers"
)
tab <- as.table(tab)
tab
```

```
##                                     SVM
## Computational speed                 Slow- especially for large banks with more u
ser data
## Interpretability                    Medium- linear SVM is more interpretable tha
n radial kernel SVM
## Predictive accuracy                 High
## Dealing with irrelevant features    Poor- kernel includes all features
## Natural handling categorical features Moderately-change categorical levels to seve
ral dummy variables
## Standardization required            No
## Robustness to outliers              Extremely Good- only depend on borderline da
ta
##                                     Random forest
## Computational speed                 Medium
## Interpretability                    Poor-black box, not suitable for banks to be
compliant
## Predictive accuracy                 Very high
## Dealing with irrelevant features    Good
## Natural handling categorical features Extremely
## Standardization required            No
## Robustness to outliers              Good
##                                     Lasso
## Computational speed                 Fast
## Interpretability                    Good- classic logit model, which is suitable
for banks to interpret based on individual information collected
## Predictive accuracy                 Moderately high
## Dealing with irrelevant features    Moderately good
## Natural handling categorical features Moderately-change categorical levels to seve
ral dummy variables
## Standardization required            Yes
## Robustness to outliers              Poor- may be affected by high leverage point
s or outliers, but not so much given the banks' applicant number is tremendous
```

(Random Forest Vs. Stepwise) However, is the lost in interpretability worth it for the gain in predictive accuracy? There is indeed a noticeable amount of increase in auc, while the interpretability is completely lost as it is a black box. In the end, Banks would like to utilize this machine learning model as a guide to better decision making and have higher certainty in whether an applicant will repay its debt. Interpretability is a necessity, as the general public, perhaps more importantly the regulatory, must be able to completely understand the underlying algorithms. Therefore Random Forest might not be suitable for this specific task.

(Radial SVMs Vs. Step-wise) As a high-dimensional model,the high auc is an implication that there are some underlying nonlinear high-dimensional relationships. The benefit of using Radial Kernel SVM is that the interpretability is preserved. The hyperplane which separates the good and bad accounts can be written down exactly, providing a neat representation of how each new account will be quantified for a decision, which have the benefit of displaying to the regulatories how the underlying algorithms work.

Compared with logistic regression in general, SVMs use hinge loss as opposed to the log loss used in the logistic regression. Theoretically, this creates “sparsity” in the sense that not all observations are influential in classifying a data point, in other words, it is local at boundary. This is a benefit as most of the time in credit analysis, class imbalance persists. Having more local methods will mean that “bad” accounts are being treated more closely than a global model.

However, the training time for a model grows exponentially fast with sample size. For example, in our example of 26203 training observations, the model ran for an hour for it to complete training. For a leading Chinese bank, it has around 700 million credit card users in total just as of last year. The time it would take for the

model to finish training could hinder the applicability of the model. Therefore is it not clear whether SVMs will be better than stepwise logistic regression.

(Lasso Vs. Step-wise) It can be seen that the Lasso model predicts the highest AUC across all the logistic models, including stepwise logistic regression. This was expected as the minimum MSE on test set using cross-validation was found for Lasso model across all the models

The chosen model in lasso contains almost all the 15 predictor variables. The weightage of each predictor variable for every component is given by scores, with gender and car ownership given the highest score. This suggests that all the predictor variables are contributing in predicting the response variable, which makes sense for the procedure of information collection and validation for credit card application.

As a conclusion, while the more sophisticated models have a higher predictive accuracy, we believe that interpretability of the model should still remain for the credit analyst and also whoever would need to review the result to be able to understand exactly why the model gives an output. Hence, SVMs and Tree methods failed to suffice this criteria, while Lasso serves as a reasonable alternative to the existing stepwise models.

References

[1. https://www.sciencedirect.com/science/article/abs/pii/S0167923613002625 (https://www.sciencedirect.com/science/article/abs/pii/S0167923613002625)]

[2. https://mp.weixin.qq.com/s/upjzuPg5AMIDsGxlpqnoCg (https://mp.weixin.qq.com/s/upjzuPg5AMIDsGxlpqnoCg)]

[3. https://www.kaggle.com/rikdifos/credit-card-approval-prediction (https://www.kaggle.com/rikdifos/credit-card-approval-prediction)]

[4. https://www.researchgate.net/post/How-do-I-calculate-the-best-cutoff-for-ROC-curves (https://www.researchgate.net/post/How-do-I-calculate-the-best-cutoff-for-ROC-curves)]

[5. https://economictimes.indiatimes.com/industry/banking/finance/women-getting-more-credit-conscious-also-default-less-than-men-credit-bureaus/articleshow/74516703.cms?from=mdr (https://economictimes.indiatimes.com/industry/banking/finance/women-getting-more-credit-conscious-also-default-less-than-men-credit-bureaus/articleshow/74516703.cms?from=mdr)]

[6. ISLR Chapter 4.3]

[7. https://stackoverflow.com/questions/29015282/predict-function-error-for-probabilities-in-glmnet (https://stackoverflow.com/questions/29015282/predict-function-error-for-probabilities-in-glmnet)]

[8. https://bradleyboehmke.github.io/HOML/regularized-regression.html#lm-features (https://bradleyboehmke.github.io/HOML/regularized-regression.html#lm-features)]

[9. https://www.tidymodels.org/learn/work/tune-svm/ (https://www.tidymodels.org/learn/work/tune-svm/)]

[10. https://dzone.com/articles/variable-importance-and-how-it-is-calculated (https://dzone.com/articles/variable-importance-and-how-it-is-calculated)]

[11. https://www.tidymodels.org/start/tuning/ (https://www.tidymodels.org/start/tuning/)]