

# Report on Default of Credit Card Clients Dataset

Masayoshi Sato

2021/6/26

## Introduction

Finance is one of fields where machine learning is commonly used. It deals with a huge amount of data and is surrounded by a lot of uncertainties. To predict future, ranging from stock market prices to individual repayment of a debt, using pre-existing data is a crucial part of finance. In this paper, we will deal with a problem many credit card companies are facing. Can we predict whether a credit client will pay their debt or fail to pay? Traditionally, finding a credible borrower have been a kind of know-how, or skill and experience nurtured by experts. Instead, we try to form machine learning models. using a dataset which is open to public.

The dataset we use, “Default of Credit Card Clients Dataset” is stored in Kaggle website. It was collected in Taiwan in 2005. It has 24 variables, such as age, education, and payment condition. Outcome has two results, “0” non-default, “1” default. Each data was anonymously collected and labeled with individual IDs.

Our goal is to form a classification model which predicts the most accurate outcome, default or not. We need to bear it in mind that its distribution of these outcomes is imbalanced. Namely, the number of default clients are small compared to non- default clients. To address the issue, we will use not only accuracy but also other criteria, balanced accuracy.

We will use three machine learning models, logistic regression, decision tree, and random forest. If necessary, we will tune their parameters to find the best solution. Our procedures are as follows :

1. Data exploration and data cleansing
2. Splitting the dataset into train\_set, validation\_set, and test\_set
3. Applying models, logistic regression, decision tree, and random forest
4. Considering models performance, and evaluating

This paper is written as a final assignment in “HarvardX PH125.9x Data Science: Capstone.”

## Packages and Dataset

In this paper, we use following R packages, “tidyverse<sup>1</sup>”, “DataExplorer<sup>2</sup>”, “gridExtra<sup>3</sup>”, “rpart<sup>4</sup>”, “caret<sup>5</sup>”, and “ranger<sup>6</sup>”.

We use a dataset stored in Kaggle<sup>7</sup> website. In the description, it says, “This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005.” It is CSV file.

---

<sup>1</sup><https://cran.r-project.org/web/packages/tidyverse/index.html>

<sup>2</sup><https://cran.r-project.org/web/packages/DataExplorer/index.html>

<sup>3</sup><https://cran.r-project.org/web/packages/gridExtra/index.html>

<sup>4</sup><https://cran.r-project.org/web/packages/rpart/index.html>

<sup>5</sup><https://cran.r-project.org/web/packages/caret/index.html>

<sup>6</sup><https://cran.r-project.org/web/packages/ranger/index.html>

<sup>7</sup><https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset>

Kaggle requires registration to download the dataset. For the sake of convenience, the file is stored in my GitHub repository<sup>8</sup>.

## Data Exploration

First, we look at the downloaded dataset. Here are variables and their data type.

```
## spec_tbl_df [30,000 x 25] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##   $ ID                : num [1:30000] 1 2 3 4 5 6 7 8 9 10 ...
##   $ LIMIT_BAL         : num [1:30000] 20000 120000 90000 50000 50000 50000 50000 100000 140000 ...
##   $ SEX               : num [1:30000] 2 2 2 2 1 1 1 2 2 1 ...
##   $ EDUCATION         : num [1:30000] 2 2 2 2 2 1 1 2 3 3 ...
##   $ MARRIAGE          : num [1:30000] 1 2 2 1 1 2 2 2 1 2 ...
##   $ AGE               : num [1:30000] 24 26 34 37 57 37 29 23 28 35 ...
##   $ PAY_0             : num [1:30000] 2 -1 0 0 -1 0 0 0 0 -2 ...
##   $ PAY_2             : num [1:30000] 2 2 0 0 0 0 0 -1 0 -2 ...
##   $ PAY_3             : num [1:30000] -1 0 0 0 -1 0 0 -1 2 -2 ...
##   $ PAY_4             : num [1:30000] -1 0 0 0 0 0 0 0 0 -2 ...
##   $ PAY_5             : num [1:30000] -2 0 0 0 0 0 0 0 0 -1 ...
##   $ PAY_6             : num [1:30000] -2 2 0 0 0 0 0 -1 0 -1 ...
##   $ BILL_AMT1         : num [1:30000] 3913 2682 29239 46990 8617 ...
##   $ BILL_AMT2         : num [1:30000] 3102 1725 14027 48233 5670 ...
##   $ BILL_AMT3         : num [1:30000] 689 2682 13559 49291 35835 ...
##   $ BILL_AMT4         : num [1:30000] 0 3272 14331 28314 20940 ...
##   $ BILL_AMT5         : num [1:30000] 0 3455 14948 28959 19146 ...
##   $ BILL_AMT6         : num [1:30000] 0 3261 15549 29547 19131 ...
##   $ PAY_AMT1          : num [1:30000] 0 0 1518 2000 2000 ...
##   $ PAY_AMT2          : num [1:30000] 689 1000 1500 2019 36681 ...
##   $ PAY_AMT3          : num [1:30000] 0 1000 1000 1200 10000 657 38000 0 432 0 ...
##   $ PAY_AMT4          : num [1:30000] 0 1000 1000 1100 9000 ...
##   $ PAY_AMT5          : num [1:30000] 0 0 1000 1069 689 ...
##   $ PAY_AMT6          : num [1:30000] 0 2000 5000 1000 679 ...
##   $ default.payment.next.month: num [1:30000] 1 1 0 0 0 0 0 0 0 0 ...
##   - attr(*, "spec")=
##     .. cols(
##       .. ID = col_double(),
##       .. LIMIT_BAL = col_double(),
##       .. SEX = col_double(),
##       .. EDUCATION = col_double(),
##       .. MARRIAGE = col_double(),
##       .. AGE = col_double(),
##       .. PAY_0 = col_double(),
##       .. PAY_2 = col_double(),
##       .. PAY_3 = col_double(),
##       .. PAY_4 = col_double(),
##       .. PAY_5 = col_double(),
##       .. PAY_6 = col_double(),
##       .. BILL_AMT1 = col_double(),
##       .. BILL_AMT2 = col_double(),
##       .. BILL_AMT3 = col_double(),
##       .. BILL_AMT4 = col_double(),
##       .. BILL_AMT5 = col_double(),
```

<sup>8</sup>[https://github.com/masa951125/Final\\_project/raw/main/UCI\\_Credit\\_Card.csv](https://github.com/masa951125/Final_project/raw/main/UCI_Credit_Card.csv)

```
## .. BILL_AMT6 = col_double(),
## .. PAY_AMT1 = col_double(),
## .. PAY_AMT2 = col_double(),
## .. PAY_AMT3 = col_double(),
## .. PAY_AMT4 = col_double(),
## .. PAY_AMT5 = col_double(),
## .. PAY_AMT6 = col_double(),
## .. default.payment.next.month = col_double()
## .. )
```

It has 30000 rows and 25 columns. “Default.payment.next.month” is our target variable. Some variables’ values are limited number of integers. Some are rather wide range of numbers. “SEX”, “EDUCATION”, “MARRIAGE”, “PAY\_0” -“PAY\_6”, and “default.payment.next.month” look like categorical data. Other features seem to be numerical.

We know there are no NAs, Nulls in the dataset.

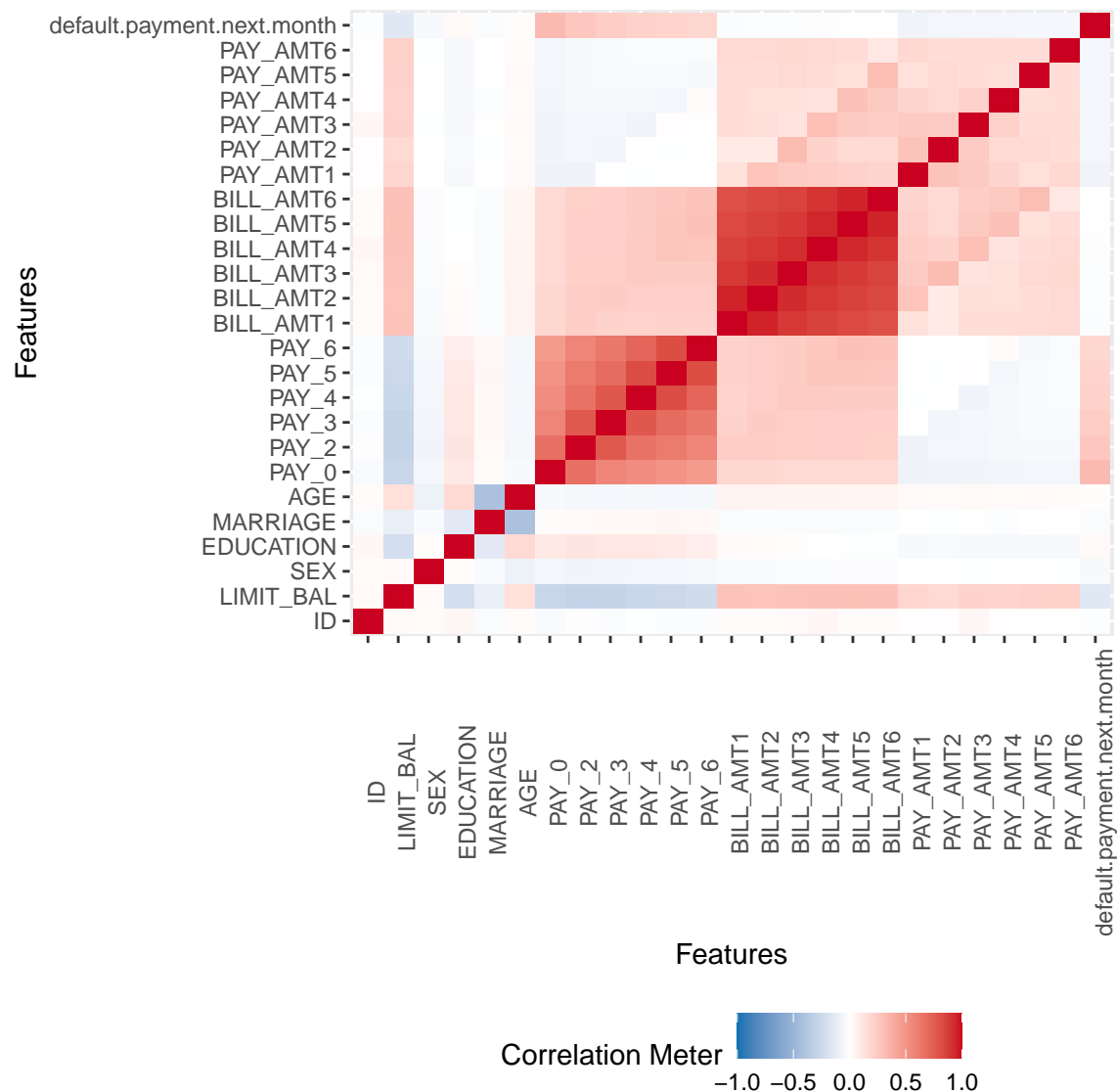
```
#Number of NAs
sum(is.na(original_default))
```

```
## [1] 0
```

```
#Number of Nulls
sum(is.null(original_default))
```

```
## [1] 0
```

How these predictors are correlated? We use “plot\_correlation” function to investigate this.



Takeaways from this are;

1. Outcome (default.payment.next.month) has a strong positive correlation with PAY (positive), and PAY\_AMT(negative), but a weak correlation with BILL\_AMT.
2. Overall, LIMIT\_BAL has a relatively strong correlation with other factors (except SEX).
3. EDUCATION, MARRIAGE, AGE have relatively strong correlation with one another.
4. EDUCATION and AGE have a relatively weak correlation with PAY and BILL\_AMT respectively.
5. PAY and BILL\_AMT, BILL\_AMT and PAY\_AMT have strong correlation.

We will look into individual features further.

## 1 Outcome

First, we look into the target variable, "default.payment.next.month". The data description says, "Default payment, 1=yes, 0=no."<sup>9</sup> We show the proportion of "0", "1".

<sup>9</sup><https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset>

```
##
##      0      1
## 0.7788 0.2212
```

It is imbalanced. This means that if we predict all the outcome as “0”, we will get 77.9% accuracy. We should be aware of this fact considering model’s evaluation.

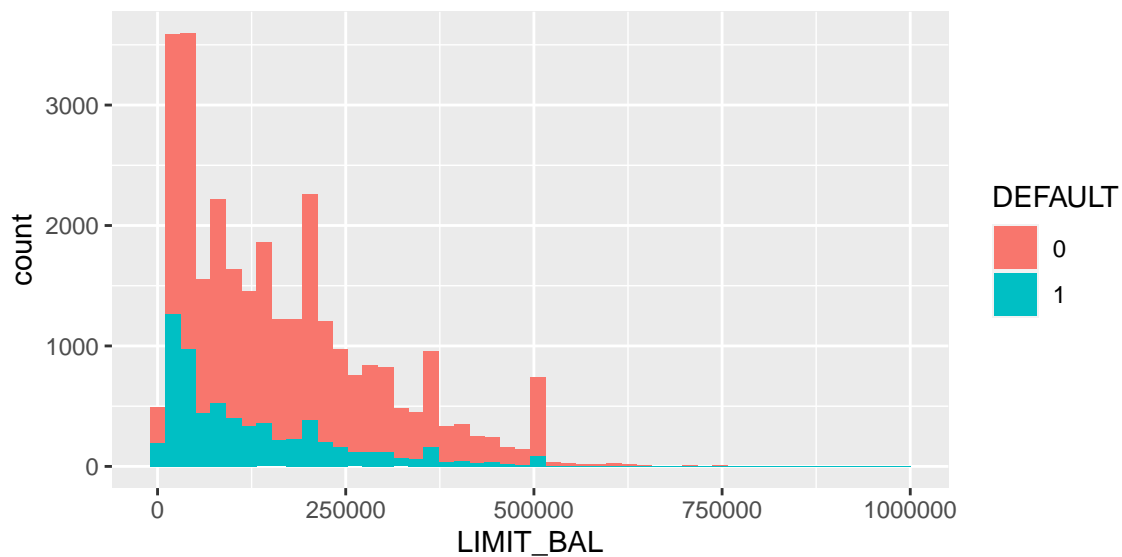
We change the name, “default.payment.next.month”, to “DEFAULT” for the sake of convenience. Also, we change this numeric variable into factor.

## 2 “LIMIT\_BAL”

This is an “amount of given credit in NT dollars (includes individual and family/supplementary credit)”<sup>10</sup> NT stands for “New Taiwan”. It is numerical data.

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 10000   50000  140000  167484  240000 1000000
```

We draw its distribution filling the proportion of default.



Distribution is skewed right. Default clients seem to be gathered around lower range of LIMIT\_BAL values.

## 3 “SEX”

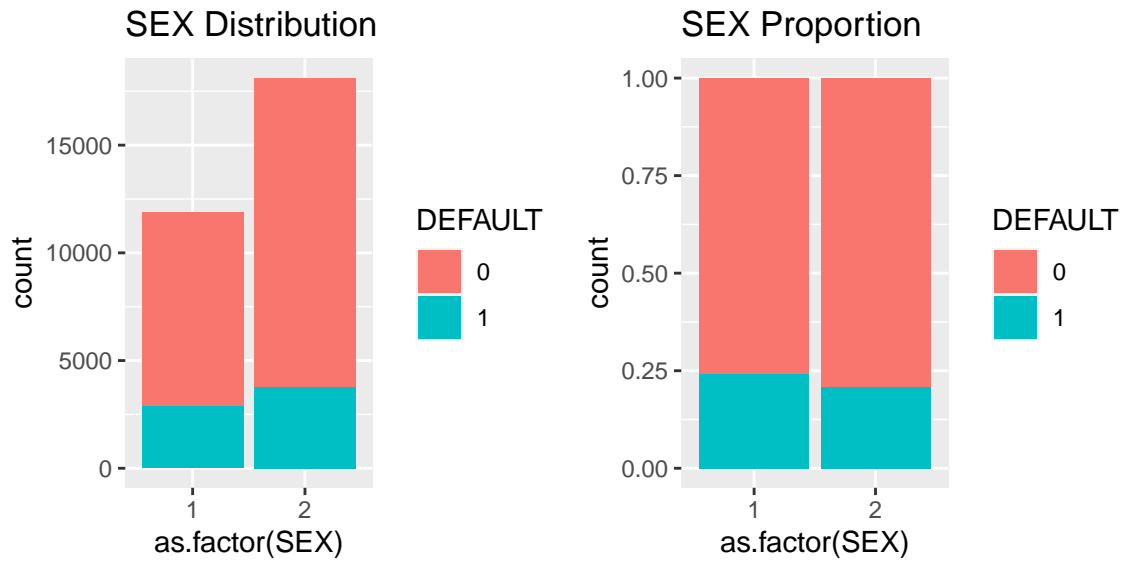
The values “1”, “2” correspond to male and female respectively<sup>11</sup>. Male is approximately 40% and female is 60%.

```
##
##      1      2
## 0.3962667 0.6037333
```

<sup>10</sup><https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset> NT stands for “New Taiwan”.

<sup>11</sup><https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset>

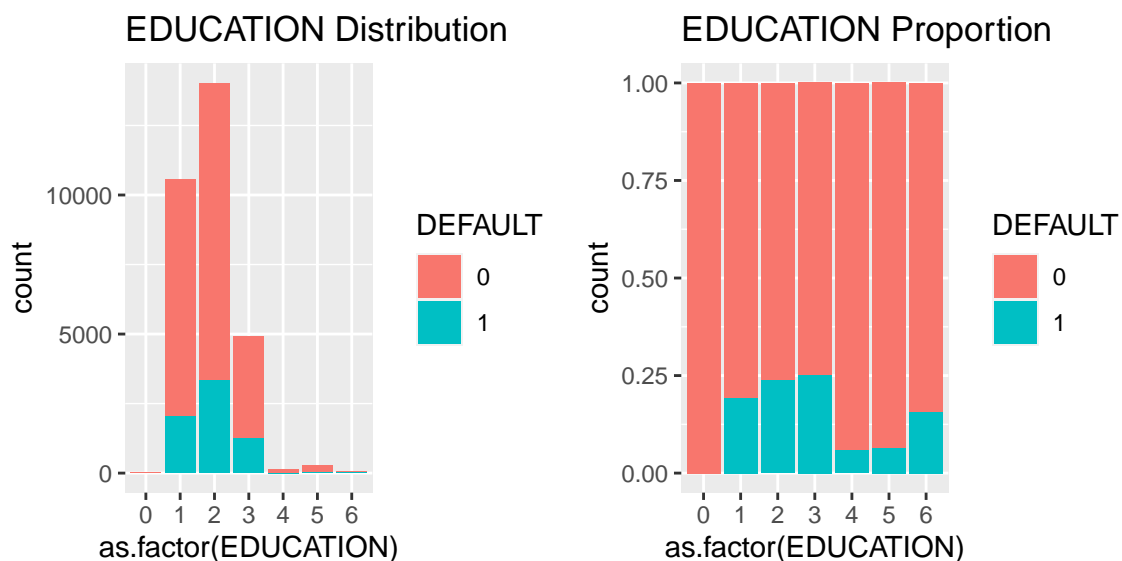
We draw its distribution and its proportion in terms of default rates.



There seemed to be little difference between genders. This categorical variable seems to be irrelevant to the outcome.

#### 4 “EDUCATION”

In this variable, values are “1”, “2”, “3”, “4”, “5”, “6”. They are categorical values. The numbers have meanings as follows ; 1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown<sup>12</sup>. We plot its distribution and stacked bar graph.



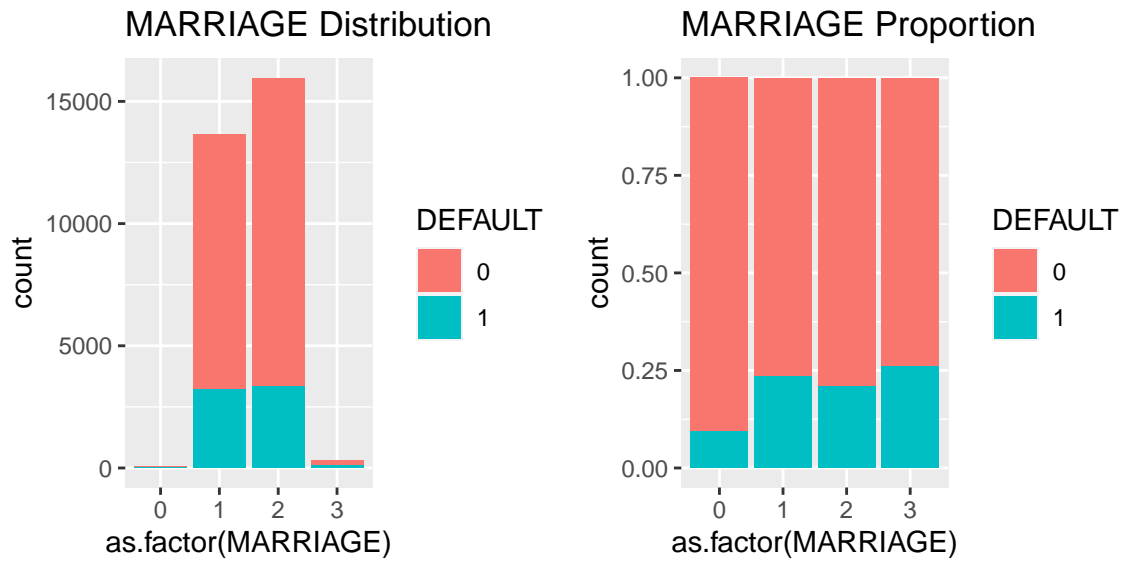
People whose final education is high school have relatively high default rate. On the other hand, people whose final education is graduate school have low default rate.

<sup>12</sup><https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset>

## 5 “MARRIAGE”

This is also categorical data. The value number means clients’ marital status<sup>13</sup>. 1=married, 2=single, 3=others. But there are 54 individuals whose values are 0.

We draw its distribution graph and stacked bar graph.



Value 0 is negligible. Comparing married and single people, married people are a little more likely to default.

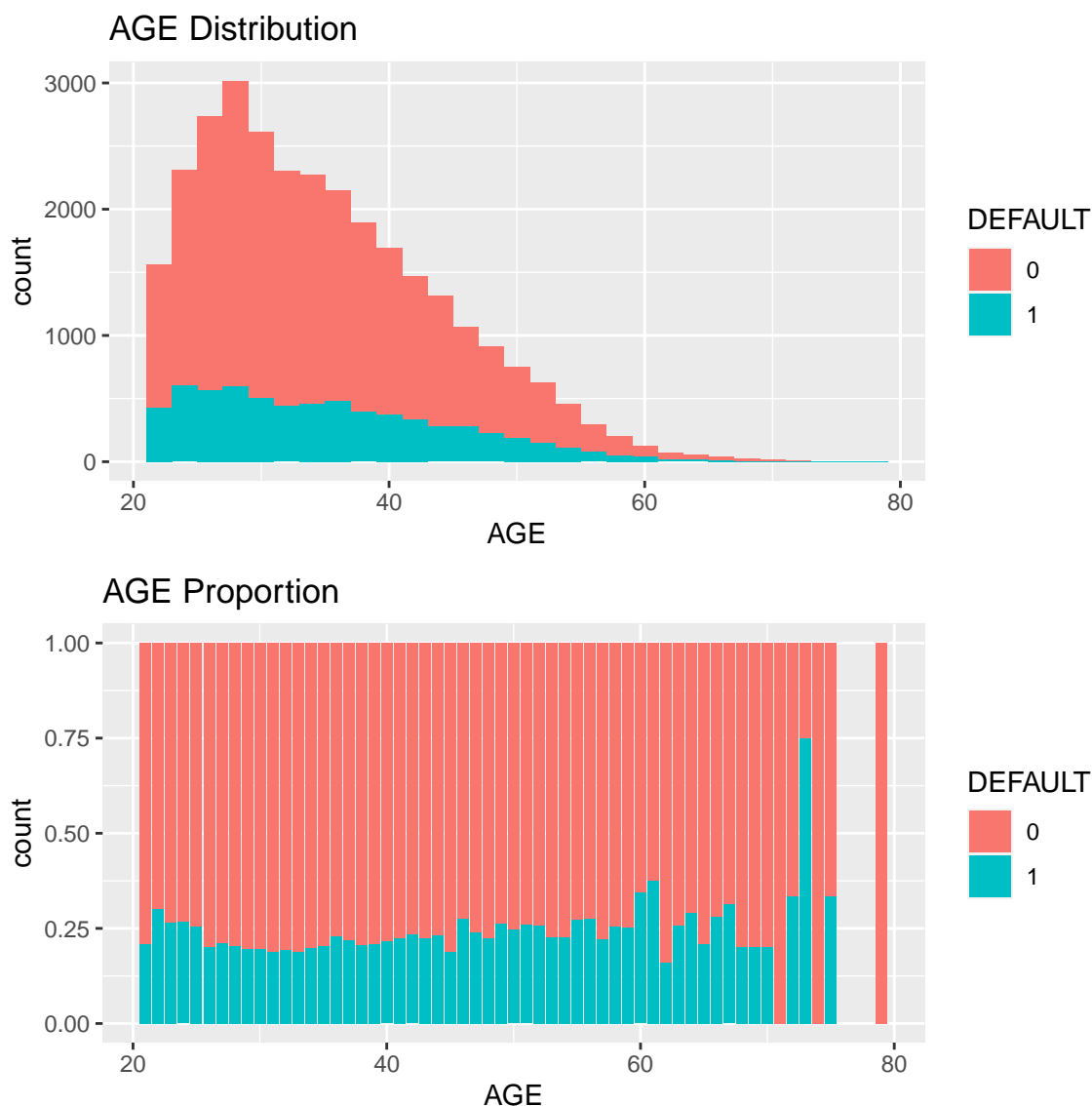
## 6 “AGE”

As its name indicates, it is numerical data.

##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	21.00	28.00	34.00	35.49	41.00	79.00

We plots its distribution filling the bar with each age bin’s default rate.

<sup>13</sup><https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset>



The number of clients are decreasing as they get older. Regarding default clients, younger people in their early 20s and older age groups around 60s show higher proportions of default than other age groups.

## 7 “PAY”

Variable from PAY\_0, PAY\_2 ~PAY\_6 have the same values, as is explained in the description<sup>14</sup>. They are categorical data. PAY\_0 means repayment status in September, 2005. Then go back in time by a month until April, 2005. Values are ;

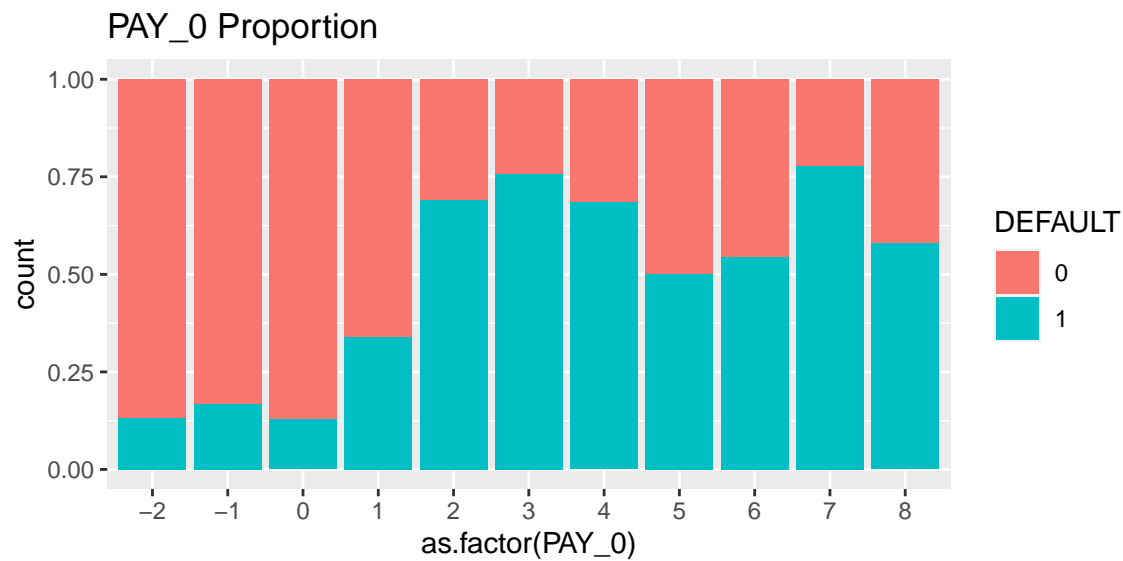
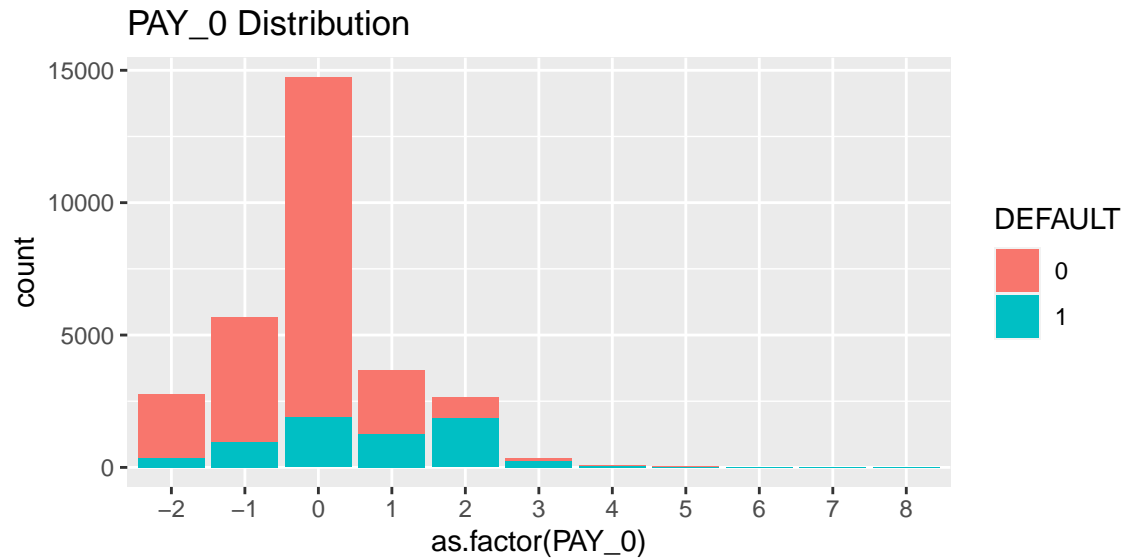
```
## [1] 2 -1 0 -2 1 3 4 8 7 5 6
```

They mean; “-1”= pay duly, “1”= payment delay for 1 month, “2” = payment delay for 2 months, ... 9 = payment delay for 9 months and above. “0” and “-2” are not defined.

We draw PAY\_0 distribution and stacked bar graph.

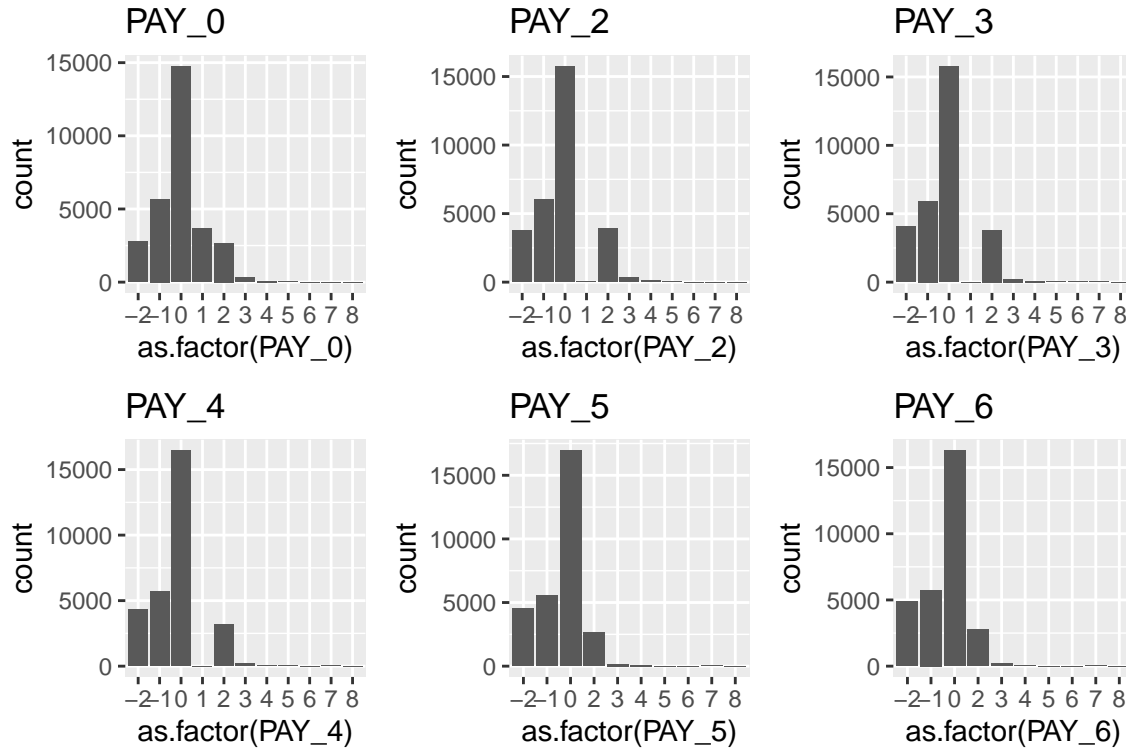
<sup>14</sup><https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset>





We are not sure what “-2” and “0” mean. Non-default clients distribution is centered around 0. Default clients one is left skewed. From these two graphs we understand that as payment delay becomes longer, clients are more likely to come to a default.

PAY\_2 ~ PAY\_6 's distributions are almost as the same as PAY\_0. We show their distribution.



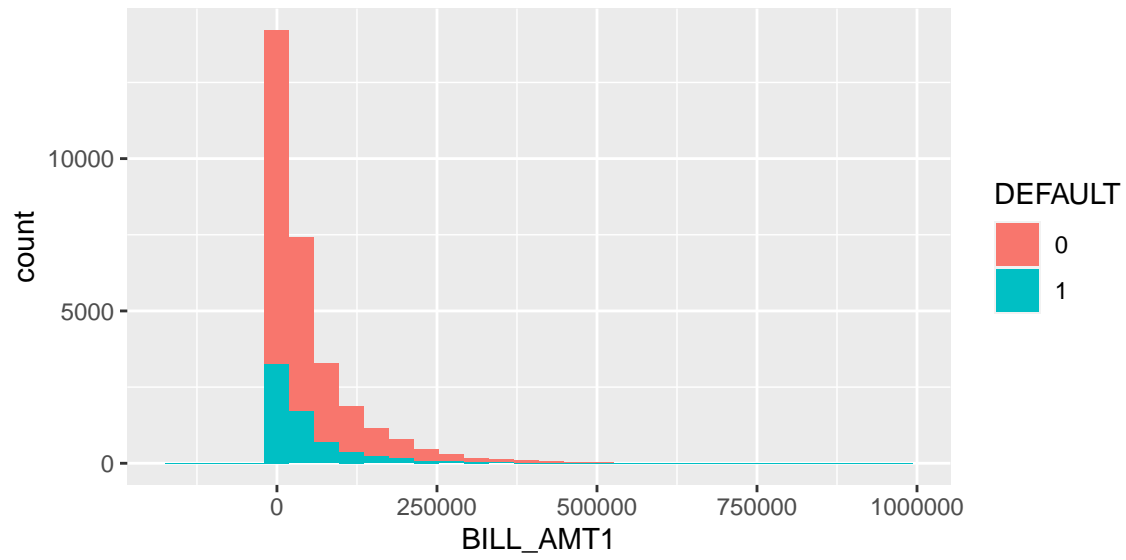
## 8 “BILL\_AMT”

This variable means an amount of bill statement. BILL\_AMT1 is a record in September, 2005<sup>15</sup>. Then go back in time by a month until April, 2005. Likewise previous variables, we plot BILL\_AMT distribution, filling each bar with its default rate. This is numerical variable.

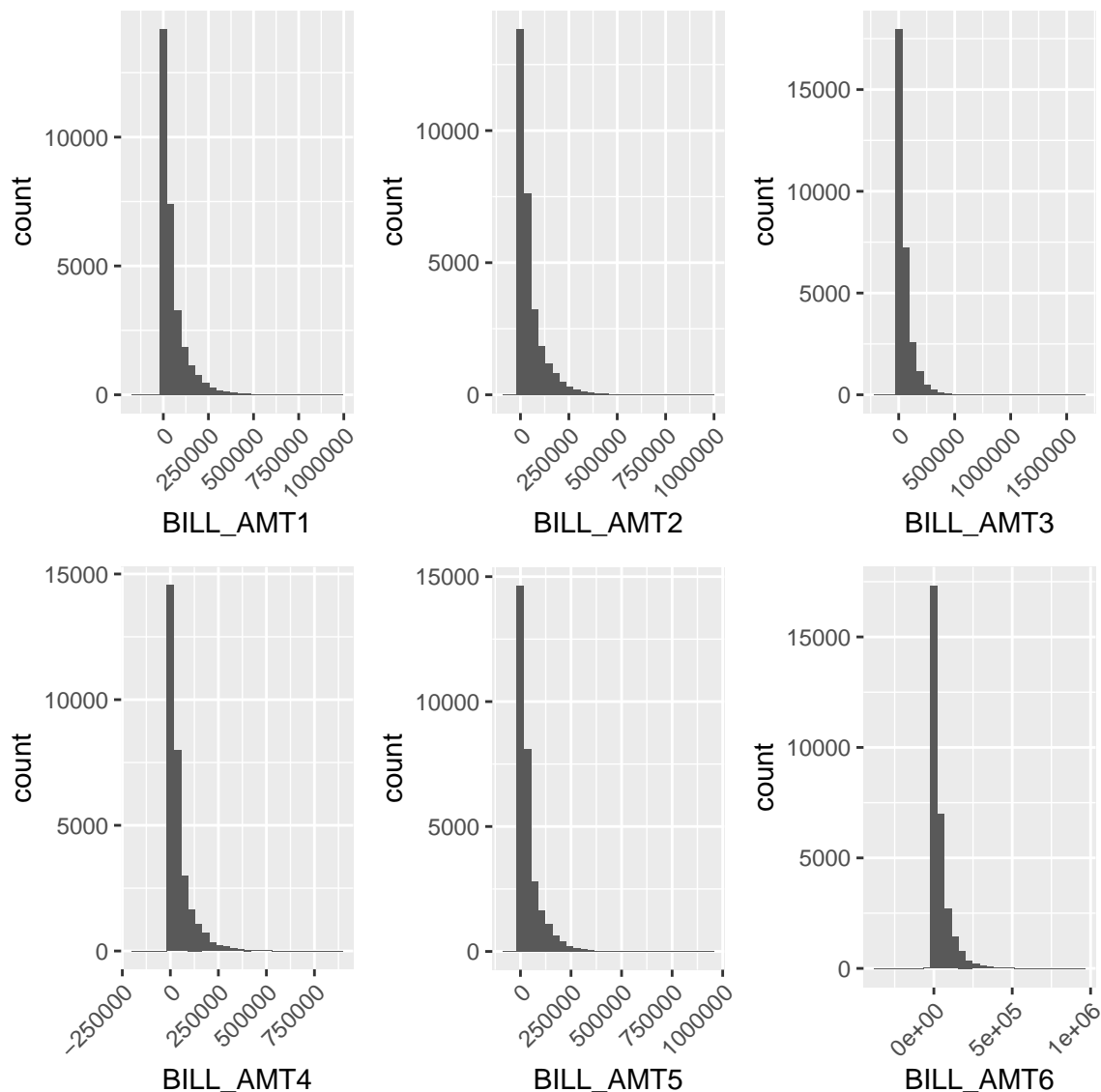
##	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4
##	Min. : -165580	Min. : -69777	Min. : -157264	Min. : -170000
##	1st Qu.: 3559	1st Qu.: 2985	1st Qu.: 2666	1st Qu.: 2327
##	Median : 22382	Median : 21200	Median : 20089	Median : 19052
##	Mean : 51223	Mean : 49179	Mean : 47013	Mean : 43263
##	3rd Qu.: 67091	3rd Qu.: 64006	3rd Qu.: 60165	3rd Qu.: 54506
##	Max. : 964511	Max. : 983931	Max. : 1664089	Max. : 891586
##	BILL_AMT5	BILL_AMT6		
##	Min. : -81334	Min. : -339603		
##	1st Qu.: 1763	1st Qu.: 1256		
##	Median : 18105	Median : 17071		
##	Mean : 40311	Mean : 38872		
##	3rd Qu.: 50191	3rd Qu.: 49198		
##	Max. : 927171	Max. : 961664		

Maximum values are fluctuating, but medians and means are decreasing from BILL\_AMT1 to BILL\_AMT6.

<sup>15</sup><https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset>



It is right skewed. From BILL\_AMT1 to BILL\_AMT6, their distributions are almost the same as are shown in following plots.



## 9 “PAY\_AMT”

This variable means an amount of previous payment. PAY\_AMT1 is an amount of previous payment in September, 2005<sup>16</sup>. Likewise BILL\_AMT, PAY\_AMT goes back in time by a month from August to April, 2005 which is PAY\_AMT6. This is numerical data.

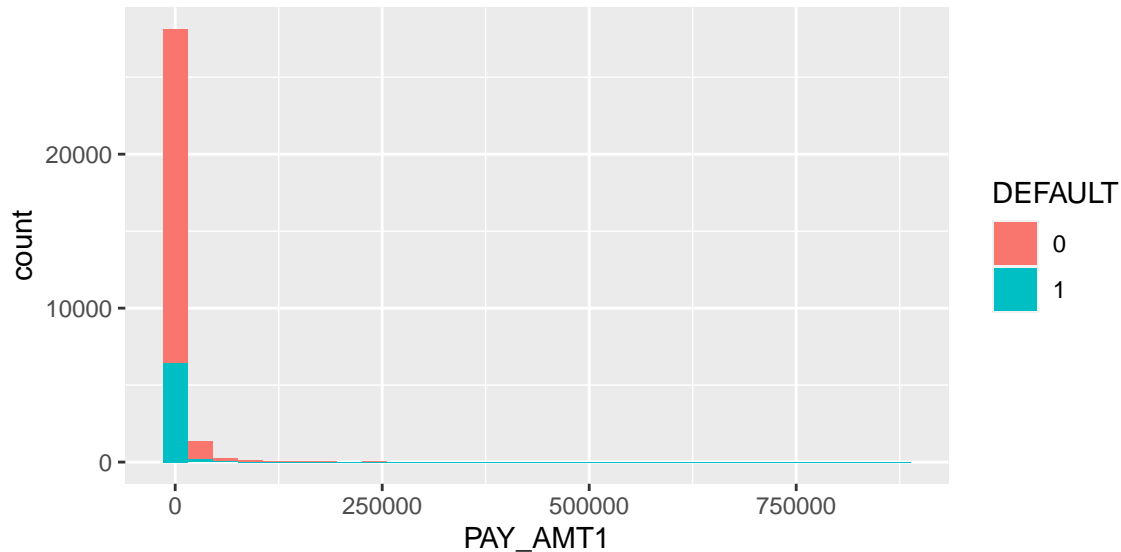
##	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4
##	Min. : 0	Min. : 0	Min. : 0	Min. : 0
##	1st Qu.: 1000	1st Qu.: 833	1st Qu.: 390	1st Qu.: 296
##	Median : 2100	Median : 2009	Median : 1800	Median : 1500
##	Mean : 5664	Mean : 5921	Mean : 5226	Mean : 4826
##	3rd Qu.: 5006	3rd Qu.: 5000	3rd Qu.: 4505	3rd Qu.: 4013
##	Max. : 873552	Max. : 1684259	Max. : 896040	Max. : 621000
##	PAY_AMT5	PAY_AMT6		

<sup>16</sup><https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset>

```
## Min.   :    0.0   Min.   :    0.0
## 1st Qu.:  252.5   1st Qu.:   117.8
## Median : 1500.0   Median :  1500.0
## Mean   :  4799.4   Mean    :  5215.5
## 3rd Qu.: 4031.5   3rd Qu.:  4000.0
## Max.   :426529.0   Max.    :528666.0
```

Their maximum values are fluctuating, but medians are gradually decreasing from PAY\_AMT1 to PAY\_AMT6.

Here is PAY\_AMT1's plot.

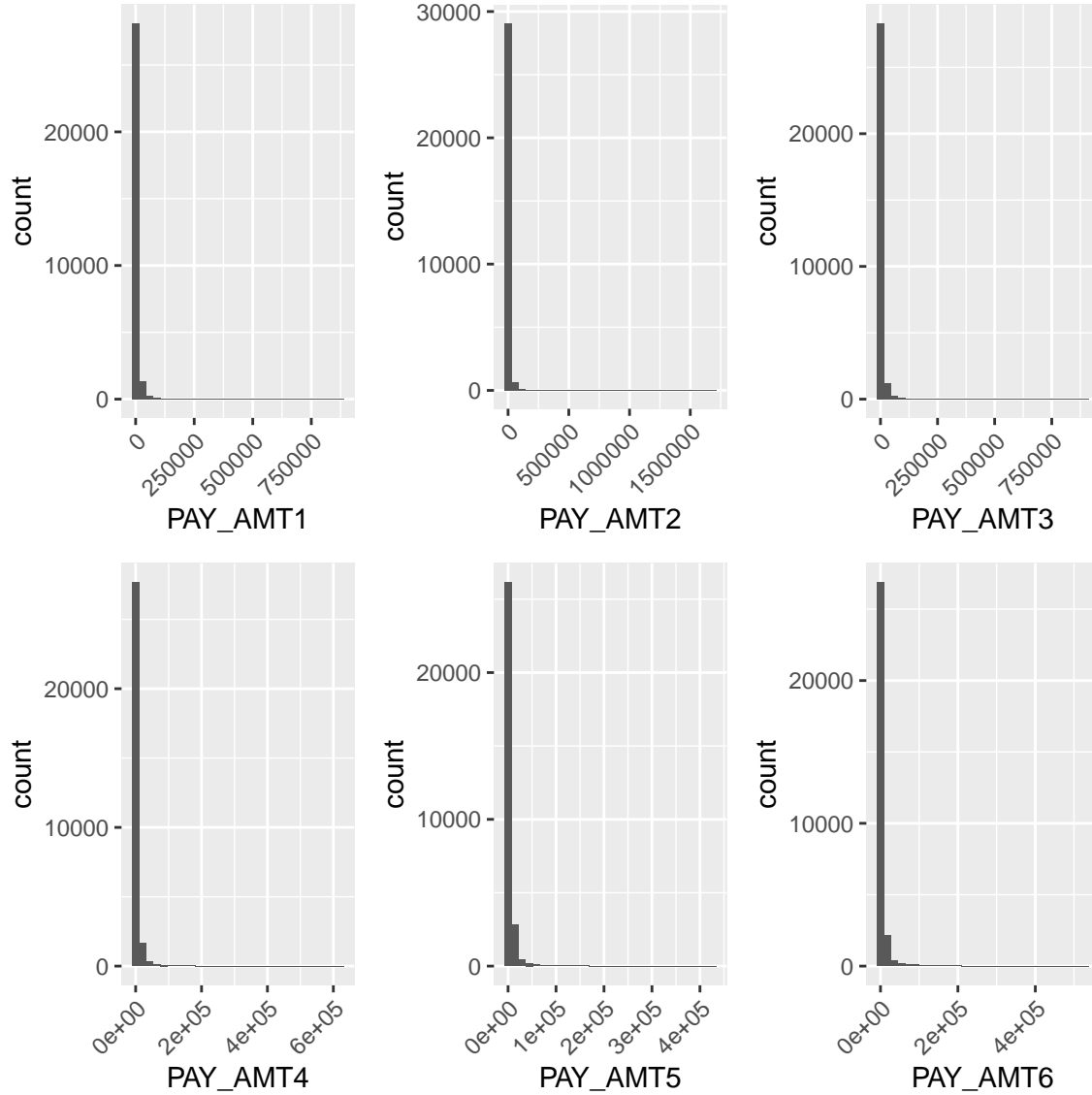


As we have seen a distribution summary, it is right skewed significantly. Maximum amount is more than 850,000 NT dollars, but its mean and median are 5664 and 2100 respectively. Also, about 17% of clients' values are 0.

```
#proportion of clients whose PAY_AMT is 0
mean(original_default$PAY_AMT1==0)
```

```
## [1] 0.1749667
```

From PAY\_AMT1 to PAY\_AMT6, their distributions follow the same trend as PAY\_AMT1.



## Data Preparation

Before going further, we will arrange our dataset to make it easier to investigate. First, we remove ID column, as it is irrelevant to the outcome.

Currently our data is composed of numerical values. Categorical values need to be changed to a factor. As we saw in data exploration, SEX, EDUCATION, MARRIAGE, PAY\_0~PAY\_6 are categorical values.

Regarding numerical values, “LIMIT\_BAL” ranges from 10000 to 1000000, but “AGE” from 21 to 79 as we saw in the previous section. When doing regression, these wide varieties of ranges of variables cause inaccuracy. Thus we standardize these variables, using following formula.

$$Transformed.Values = \frac{Values - Mean}{Standard.Deviation}$$

We get variables whose means are 0, and standard deviations are 1 using function “scale”.

Then, we split the data into two. As we have relatively a large amount of data, 30000, the proportion we use is 80% and 20%. The smaller dataset will be used when evaluating a model at the final stage. We call it `test_set`.

We are going to use decision tree, and random forest later. They can be tuned to produce improved results by finding hyperparameters. If we tune the model using hyperparameters again and again, this might cause overfitting. To avoid this, we split the larger dataset again and produce two datasets, “`train_set`” and “`validation_set`”. Split proportion is the same as before, 80% and 20% respectively.

The three datasets have the almost same outcome ratio.

```
## # A tibble: 2 x 4
##   outcome train_set validation_set test_set
##   <dbl> <table>    <table>    <table>
## 1      0 0.7788311 0.7787961    0.7787035
## 2      1 0.2211689 0.2212039    0.2212965
```

## Model Analysis

### 1 Evaluation Metrics

Generally speaking, overall accuracy is used to evaluate a model. But this dataset has imbalanced proportion of outcomes as we explored. What consequences bring about? Guessing all responses are 0, we calculate its confusion matrix.

```
##           Reference
## Prediction      0      1
##           0 3739 1062
##           1      0      0
```

From this confusion matrix, we know even such a guess produces fairly good results.

- Accuracy (True positive + True Negative / Total) 0.7788
- Sensitivity (True Positive / True Positive+ False Negative) 1.0

In contrast,

- Specificity (True Negative / True Negative + False Positive) 0
- Balanced accuracy (arithmetic mean of sensitivity and specificity) 0.5

As such, the credit company falsely give credit to a lot of clients who will fail to repay a debt. The loss for the company would be huge. Therefore, our goal is to pursue more accurate accuracy than guessing all responses are 0, taking account of improving specificity. We will use both *accuracy* and *balanced accuracy* to evaluate each model.

## 2 Logistic Regression

Firstly, we choose logistic regression. Logistic regression is commonly used in binary classification problems (outcome is 0 and 1, or True and False). As it uses logistic transformed odds, it produces outcome ranging from 0 to 1.

We have 24 features in the dataset. There are many ways to deal with such many predictors. First, we will form a logistic regression model with fewer variables. In the data exploration, we understand predictors from PAY\_0 to PAY\_6 seem to be important. But PAY variables, BILL\_AMT variables and PAY\_AMT variables have strong correlation. In addition to this, BILL\_AMTs seem to have weak correlation with DEFAULT, target variable.

From these, we narrow down variables. We leave out BILL\_AMT2 to BILL\_AMT6, PAY\_AMT2 to PAY\_AMT6, PAY\_2 to PAY\_6 as well. Then we predict responses using glm function. The result;

```
##
## Call:
## glm(formula = DEFAULT ~ LIMIT_BAL + SEX + EDUCATION + MARRIAGE +
##      AGE + PAY_0 + PAY_AMT1 + BILL_AMT1, family = binomial(link = "logit"),
##      data = train_set)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.8600  -0.6012  -0.5244  -0.3403   3.2785
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -13.92484   105.25873  -0.132  0.89475
## LIMIT_BAL    -0.34333    0.02639 -13.008 < 2e-16 ***
## SEX2         -0.16184    0.03975  -4.071 4.68e-05 ***
## EDUCATION1    10.88972   105.25677   0.103  0.91760
## EDUCATION2    10.94220   105.25678   0.104  0.91720
## EDUCATION3    10.87972   105.25678   0.103  0.91767
## EDUCATION4    10.09138   105.25780   0.096  0.92362
## EDUCATION5     9.46368   105.25730   0.090  0.92836
## EDUCATION6    10.36704   105.25800   0.098  0.92154
## MARRIAGE1      1.57053    0.63924   2.457  0.01402 *
## MARRIAGE2      1.42360    0.63942   2.226  0.02599 *
## MARRIAGE3      1.62229    0.66105   2.454  0.01412 *
## AGE            0.01767    0.02247   0.787  0.43151
## PAY_0-1        0.18717    0.08471   2.210  0.02713 *
## PAY_00        -0.41514    0.08494  -4.887 1.02e-06 ***
## PAY_01         0.87807    0.08659  10.141 < 2e-16 ***
## PAY_02         2.21948    0.09468  23.441 < 2e-16 ***
## PAY_03         2.47682    0.17905  13.833 < 2e-16 ***
## PAY_04         1.99336    0.31677   6.293 3.12e-10 ***
## PAY_05         0.67991    0.63420   1.072  0.28369
## PAY_06         1.77298    0.73779   2.403  0.01626 *
## PAY_07         2.17427    0.84403   2.576  0.00999 **
## PAY_08         1.39957    0.56652   2.470  0.01349 *
## PAY_AMT1      -0.18223    0.03911  -4.659 3.17e-06 ***
## BILL_AMT1      0.13577    0.02525   5.378 7.54e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```



```
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 20288 on 19197 degrees of freedom
## Residual deviance: 17045 on 19173 degrees of freedom
## AIC: 17095
##
## Number of Fisher Scoring iterations: 11
```

From this, we understand that;

- Important variables :  
LIMIT\_BAL, SEX(whether a client is a female or not), PAY\_0( whether its values are 0, 1, 2, 3, 4),  
, BILL\_AMT1,PAY\_AMT1

Then fit the model to the validation set and show its confusion matrix.

```
glm_fewer_prob <- predict(glm_fewer_md1, validation_set,type="response")
glm_fewer_pred <- ifelse(glm_fewer_prob >0.5,1,0)
confusionMatrix(as.factor(glm_fewer_pred), validation_set$DEFAULT)$table
```

```
##           Reference
## Prediction    0    1
##           0 3603  740
##           1  136  322
```

Other statistic metrics are;

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
glm fewer features	0.817538	0.9636266	0.3032015	0.6334141

One of other ways is step-wise regression. This method is to repeatedly examine statistical significance of each variable in a regression model<sup>17</sup>. There are three methods; forward selection, backward elimination, forward and backward elimination. Forward selection starts from no variable and add a variable step by step. Backward elimination starts from full model and subtract variables one by one. Forward and backward combines the two. One of the most convenient features of this method is we can get logistic regression results without knowing details of variables in the dataset.

We use forward and backward step wise regression, as it is commonly used. In the step-wise regression, we make two models, null model with no predictors and full model with all predictors at first. Then we use “step” function to conduct step-wise regression. The result;

```
##
## Call:
## glm(formula = DEFAULT ~ PAY_0 + PAY_4 + LIMIT_BAL + PAY_6 + PAY_AMT2 +
##      BILL_AMT3 + PAY_AMT1 + PAY_3 + MARRIAGE + EDUCATION + SEX +
##      PAY_AMT5 + PAY_5 + PAY_AMT3 + BILL_AMT5, family = binomial(link = "logit"),
##      data = train_set)
##
## Deviance Residuals:
```

<sup>17</sup>It uses Akaike information criterion (AIC) .[https://en.wikipedia.org/wiki/Akaike\\_information\\_criterion](https://en.wikipedia.org/wiki/Akaike_information_criterion)

```

##      Min      1Q   Median      3Q      Max
## -2.3707  -0.6010  -0.5041  -0.3009   3.4687
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -15.98503   286.37745  -0.056  0.955487
## PAY_0-1        0.39057    0.10989   3.554  0.000379 ***
## PAY_00       -0.24892    0.10957  -2.272  0.023098 *
## PAY_01        0.77280    0.10051   7.689  1.48e-14 ***
## PAY_02        1.99325    0.11628  17.142 < 2e-16 ***
## PAY_03        1.90918    0.20210   9.447 < 2e-16 ***
## PAY_04        1.60586    0.33553   4.786  1.70e-06 ***
## PAY_05        0.26727    0.67139   0.398  0.690564
## PAY_06        1.59286    0.84034   1.895  0.058027 .
## PAY_07        0.96314    1.25174   0.769  0.441634
## PAY_08       -12.20222   371.73655  -0.033  0.973814
## PAY_4-1       -0.21537    0.13474  -1.598  0.109960
## PAY_40       -0.14822    0.15003  -0.988  0.323175
## PAY_41        0.92374  1076.95006   0.001  0.999316
## PAY_42        0.12951    0.16021   0.808  0.418855
## PAY_43        0.13295    0.30917   0.430  0.667174
## PAY_44        0.50734    0.53958   0.940  0.347087
## PAY_45       -2.37446    1.04205  -2.279  0.022689 *
## PAY_46      -29.76168   497.36889  -0.060  0.952284
## PAY_47       -7.03294  6762.09451  -0.001  0.999170
## PAY_48     -35.13222  6819.44800  -0.005  0.995890
## LIMIT_BAL    -0.25510    0.02807  -9.088 < 2e-16 ***
## PAY_6-1      -0.21039    0.10068  -2.090  0.036646 *
## PAY_60       -0.42701    0.10790  -3.957  7.58e-05 ***
## PAY_62       -0.07930    0.12554  -0.632  0.527636
## PAY_63        0.77912    0.30280   2.573  0.010081 *
## PAY_64        0.06776    0.55390   0.122  0.902637
## PAY_65       -0.17413    0.90140  -0.193  0.846818
## PAY_66        0.50944    0.95489   0.534  0.593684
## PAY_67     -11.83717   348.38903  -0.034  0.972896
## PAY_68       24.98387  6829.59209   0.004  0.997081
## PAY_AMT2     -0.27007    0.06028  -4.480  7.47e-06 ***
## BILL_AMT3     0.28795    0.05843   4.928  8.29e-07 ***
## PAY_AMT1     -0.14662    0.03934  -3.727  0.000194 ***
## PAY_3-1       0.09293    0.11794   0.788  0.430741
## PAY_30        0.15585    0.12940   1.204  0.228411
## PAY_31     -13.48725   616.91616  -0.022  0.982558
## PAY_32        0.50002    0.13127   3.809  0.000140 ***
## PAY_33        0.27342    0.23206   1.178  0.238701
## PAY_34       -0.05606    0.46813  -0.120  0.904670
## PAY_35        0.49636    0.98075   0.506  0.612785
## PAY_36       15.31057   371.73748   0.041  0.967147
## PAY_37       -0.61752    1.30540  -0.473  0.636180
## PAY_38     -5.77319  6819.44810  -0.001  0.999325
## MARRIAGE1     1.75261    0.66165   2.649  0.008077 **
## MARRIAGE2     1.60347    0.66191   2.422  0.015415 *
## MARRIAGE3     1.89062    0.68302   2.768  0.005639 **
## EDUCATION1    12.78941   286.37668   0.045  0.964379
## EDUCATION2    12.83888   286.37668   0.045  0.964241

```

```

## EDUCATION3      12.79347   286.37668    0.045 0.964368
## EDUCATION4      12.08256   286.37706    0.042 0.966346
## EDUCATION5      11.54511   286.37687    0.040 0.967842
## EDUCATION6      12.45988   286.37710    0.044 0.965296
## SEX2            -0.14270    0.04010   -3.558 0.000373 ***
## PAY_AMT5        -0.08804    0.03253   -2.706 0.006809 **
## PAY_5-1         -0.03317    0.12973   -0.256 0.798219
## PAY_50           0.06647    0.14304    0.465 0.642155
## PAY_52           0.41577    0.15993    2.600 0.009329 **
## PAY_53           0.08700    0.30026    0.290 0.772008
## PAY_54          -0.26936    0.58798   -0.458 0.646874
## PAY_55           1.20656    1.07849    1.119 0.263249
## PAY_56           26.86203   655.12680    0.041 0.967294
## PAY_57           20.58158  6762.07341    0.003 0.997571
## PAY_58           29.61922  7061.38645    0.004 0.996653
## PAY_AMT3        -0.08144    0.04312   -1.889 0.058898 .
## BILL_AMT5       -0.09897    0.05642   -1.754 0.079381 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 20288  on 19197  degrees of freedom
## Residual deviance: 16612  on 19132  degrees of freedom
## AIC: 16744
##
## Number of Fisher Scoring iterations: 13

```

From this, we understand that;

- Used variables:  
LIMIT\_BAL, MARRIAGE, EDUCATION, SEX, PAY\_0, PAY\_3, PAY\_4, PAY\_5, PAY\_6, PAY\_AMT1, PAY\_AMT2, PAY\_AMT3, PAY\_AMT5, BILL\_AMT3, BILL\_AMT5
- Important variables :  
PAY\_0( whether its values are -1, 1, 2, 3, 4), PAY\_3( whether its values are 2), PAY\_6( whether its values are 0), LIMIT\_BAL, BILL\_AMT3, PAY\_AMT1, PAY\_AMT2, SEX(whether a client is a female or not)

Then fit the model to the validation set and make a confusion matrix.

```

##           Reference
## Prediction    0    1
##           0 3567  714
##           1  172  348

```

Other statistic metrics are;

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
glm step wise	0.8154551	0.9539984	0.3276836	0.640841

Comparing the two models;

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
glm fewer features	0.8175380	0.9636266	0.3032015	0.6334141
glm step wise	0.8154551	0.9539984	0.3276836	0.6408410

The model with fewer features shows better accuracy than the step-wise model. But its specificity is worse than the step-wise model, in consequence produces worse balanced accuracy. We understand from these regression the selection of variables are crucial in determining the result. We look into other models.

### 3 Decision Tree

A decision tree is a familiar and intuitive method. For example, if you feel sick, a physician may ask you questions following a tree chart. You are asked whether it is yes or no at each node of the chart. After following the nodes, the doctor gets your health outcome.

Decision tree algorithm recursively divides the dataset into partitions with similar values for the outcome. A metric is used to choose the partitions. One of decision tree R functions “rpart” uses Gini index.

$$\text{Gini}(j) = \sum_{k=1}^K \hat{p}_{j,k}(1 - \hat{p}_{j,k})$$

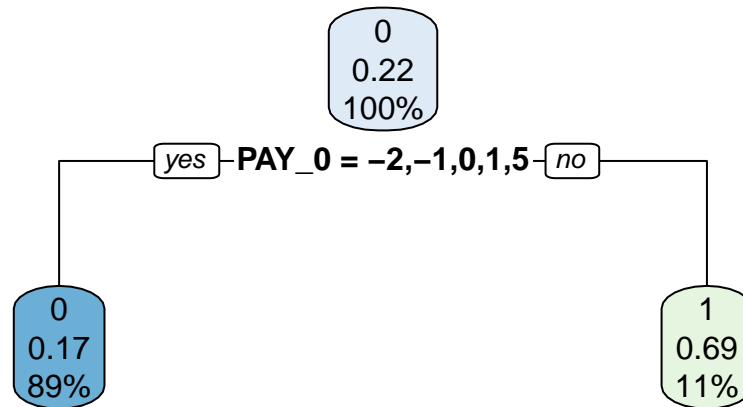
$\hat{p}_{j,k}$  as the proportion of observations in partition  $j$  that are of class  $k$ .<sup>18</sup> If there are lots of classes in the partition, Gini index, also called Gini impurity, increases up to 1. On the other hand, if there are few classes in the partition it decreases down to 0. If you predict perfectly, the index is 0.

We use “rpart” function to make a model, then predict the outcome.

```
## n= 19198
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
## 1) root 19198 4246 0 (0.7788311 0.2211689)
##    2) PAY_0=-2,-1,0,1,5 17180 2849 0 (0.8341676 0.1658324) *
##    3) PAY_0=2,3,4,6,7,8 2018 621 1 (0.3077304 0.6922696) *
```

We draw a decision tree based on this model.

<sup>18</sup>Irizarry, Rafael A, “Introduction to Data Science: Data Analysis and Prediction Algorithms with R, 31.10.4 Classification (decision) trees”. *Internet archive*, <https://rafalab.github.io/dsbook/examples-of-algorithms.html#classification-decision-trees>



This model uses only PAY\_O to make partitions. This means the credit company need to check a client's PAY\_0. If the value is -2, -1, 0, 1, 5, they are likely to repay, otherwise, they are likely to default on the debt.

Then predict the outcome, and show its confusion matrix.

```
##           Reference
## Prediction    0    1
##           0 3597  736
##           1  142  326
```

Other statistic metrics;

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
rpart	0.8171214	0.9620219	0.306968	0.634495

Compared with the previous logistic regression models, its accuracy has improved but its balanced accuracy become worse.

We try to improve this result using other function “train” in caret package. When we use “train” , it conducts “cross validation” and finds optimal parameters in the decision tree algorithm.

One of the cross validation methods is k-fold cross validation. It splits the dataset into k “folds” randomly. For each group, it takes the group as a test dataset, and takes other groups as a training dataset. Then it fits a model on the training set and evaluates it on the test set. This procedure is repeated until every K-fold serve as the test set. Finally it summarizes the performance of the model as the average of these procedures.

“ModelLookup” function in caret package tells us what parameters in rpart we can tune.

```
modelLookup("rpart")
```

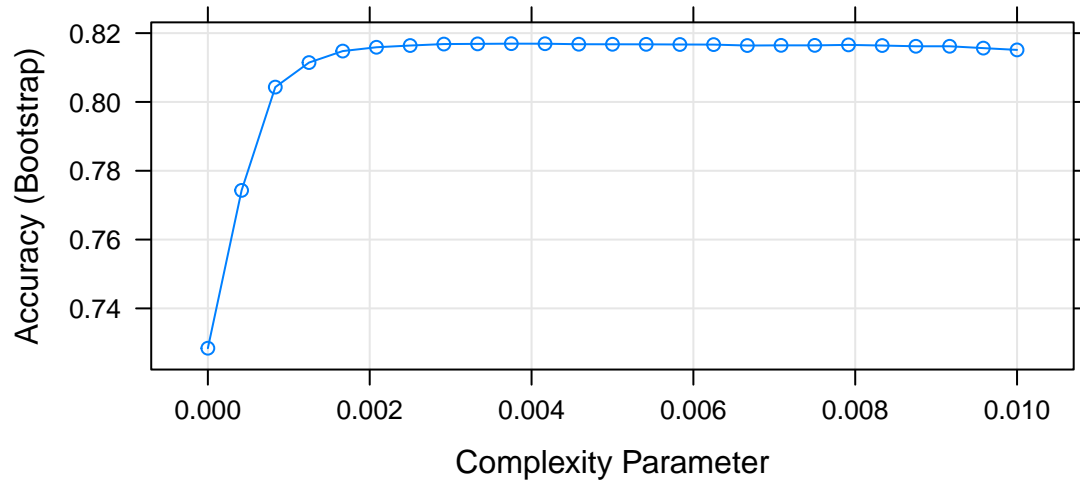
```
##  model parameter          label forReg forClass probModel
## 1  rpart          cp Complexity Parameter    TRUE    TRUE    TRUE
```

Cp stands for complexity parameter. It determines the complexity of the model. As its value gets smaller, the algorithm produces more trees. In the previous decision tree model we fit, cp is 0.01. We want to know whether cp values less than 0.01 will improve the performance.

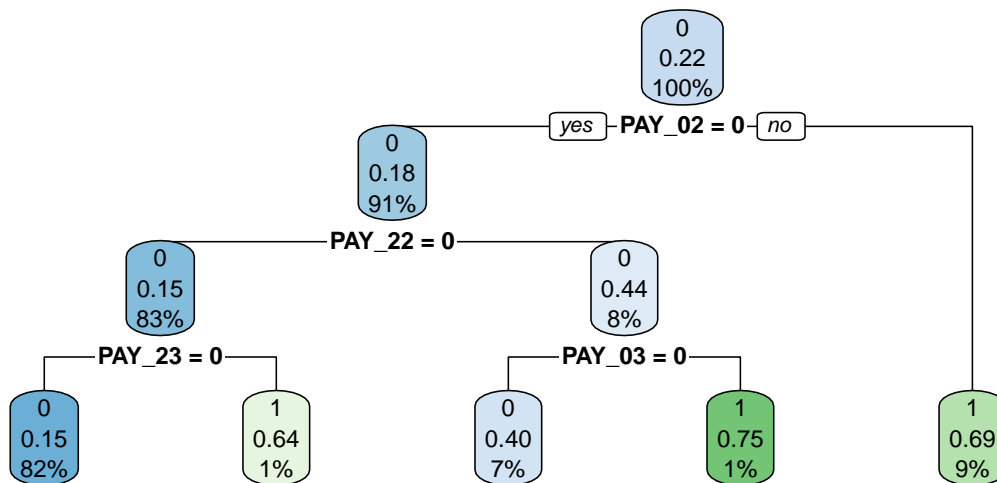
We train the dataset using “train” function. This function conducts 10 folds cross validation as a default. As we do not specify the number, it will conduct 10 folds cross validation.

```
## CART
##
## 19198 samples
##    23 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 19198, 19198, 19198, 19198, 19198, 19198, ...
## Resampling results across tuning parameters:
##
##    cp            Accuracy    Kappa
## 0.0000000000    0.7284228    0.2235331
## 0.0004166667    0.7742936    0.2859917
## 0.0008333333    0.8043130    0.3325194
## 0.0012500000    0.8114391    0.3445758
## 0.0016666667    0.8147804    0.3520765
## 0.0020833333    0.8158908    0.3523277
## 0.0025000000    0.8163979    0.3484179
## 0.0029166667    0.8168234    0.3458840
## 0.0033333333    0.8168740    0.3431768
## 0.0037500000    0.8169363    0.3425090
## 0.0041666667    0.8169238    0.3412075
## 0.0045833333    0.8167714    0.3405336
## 0.0050000000    0.8167598    0.3402173
## 0.0054166667    0.8167432    0.3408068
## 0.0058333333    0.8166985    0.3413150
## 0.0062500000    0.8166645    0.3414413
## 0.0066666667    0.8163935    0.3386679
## 0.0070833333    0.8164501    0.3381807
## 0.0075000000    0.8164443    0.3386962
## 0.0079166667    0.8165811    0.3373467
## 0.0083333333    0.8163949    0.3357974
## 0.0087500000    0.8161803    0.3344378
## 0.0091666667    0.8161803    0.3344378
## 0.0095833333    0.8156400    0.3303031
## 0.0100000000    0.8151237    0.3273365
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.00375.
```

Plot cp.



Then draw a decision tree.



As PAY variables are factored and turned into a dummy variable, a node “PAY\_02”, for example, is asking whether a client’s PAY\_0 is “-1” or not. “-1” is the second level in the PAY\_0 original column.

Fit the model to the validation set and show its confusion matrix.

```

##           Reference
## Prediction    0    1
##           0 3587  730
##           1  152  332

```

Other statistic metrics;

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
rpart tuned	0.8162883	0.9593474	0.3126177	0.6359826

Comparing two decision models, the tuned model has improved in terms of specificity and balanced accuracy. But both performed worse than logistic regression using fewer predictors in terms of balanced accuracy.

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
rpart	0.8171214	0.9620219	0.3069680	0.6344950
rpart tuned	0.8162883	0.9593474	0.3126177	0.6359826

#### 4 Random Forest

Decision tree algorithm is easy to understand as it follow human decision making process. However, it “can easily over-train” and “is not very flexible.”<sup>19</sup> To improve this, we introduce random forest algorithm.

Random forest is one of the most popular machine learning algorithms. As its name implies, it makes multiple trees “**randomly** different” from a dataset, then produce a final prediction based on the average prediction of “combination of trees”, namely “**forest**”.<sup>20</sup>

In this paper, we use “ranger” function. And show the model details;

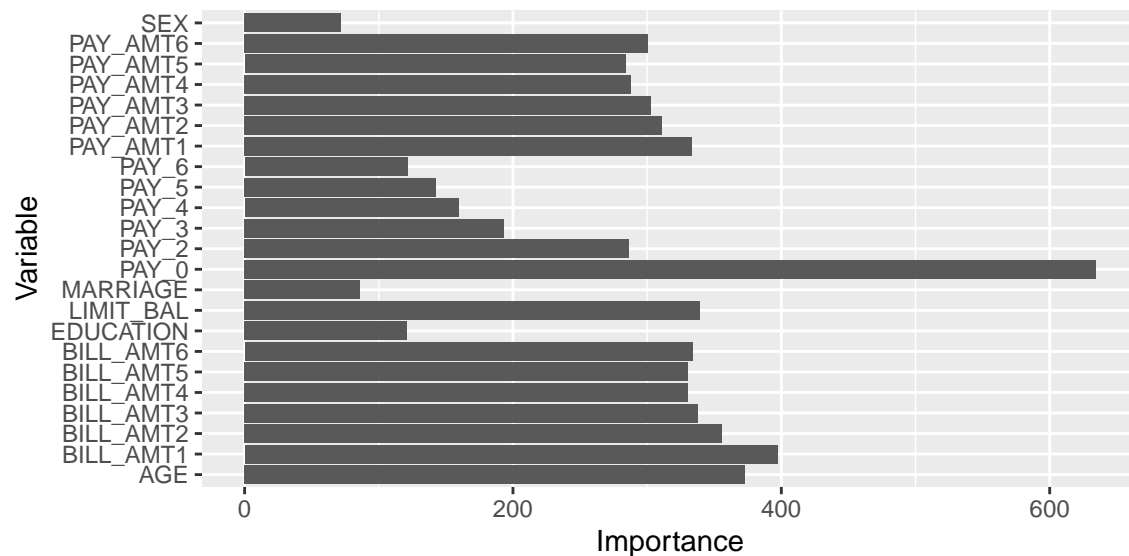
```
## Ranger result
##
## Call:
## ranger(formula = DEFAULT ~ ., data = train_set, importance = "impurity",      probability = F)
##
## Type:                                Classification
## Number of trees:                      500
## Sample size:                          19198
## Number of independent variables:      23
## Mtry:                                  4
## Target node size:                      1
## Variable importance mode:              impurity
## Splitrule:                             gini
## OOB prediction error:                  18.30 %
```

Plot the variables’ importance;

<sup>19</sup>Irizarry, Rafael A, “Introduction to Data Science: Data Analysis and Prediction Algorithms with R, 31.10.4 Classification (decision) trees”. *Internet archive*, <<https://rafalab.github.io/dsbook/examples-of-algorithms.html#classification-decision-trees>>

<sup>20</sup>Irizarry, Rafael A, “Introduction to Data Science: Data Analysis and Prediction Algorithms with R, 31.11 Random forests”. *Internet archive*, <https://rafalab.github.io/dsbook/examples-of-algorithms.html#random-forests>





As we saw in the decision tree model, PAY\_0 is considerably important compared with other features. As for other features, BILL\_AMT1, PAY\_AMT1, and LIMIT\_BAL, and AGE are important.

Then fit the model to the validation set and show its confusion matrix.

```
##           Reference
## Prediction      0      1
##           0 3564  698
##           1  175  364
```

Other statistic metrics;.

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
random forest	0.8181629	0.953196	0.3427495	0.6479728

The results has been much improved, especially specificity, compared to other models. Can we improve the result further? Using “modelLookup”, we can sse parameters which can be tuned.

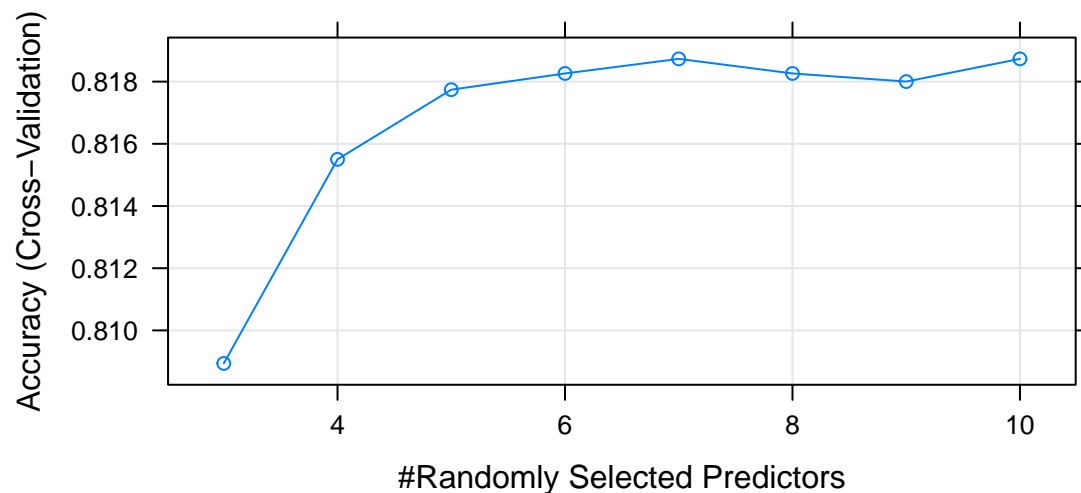
```
##   model      parameter                                label forReg forClass probModel
## 1 ranger      mtry #Randomly Selected Predictors      TRUE      TRUE      TRUE
## 2 ranger    splitrule                                Splitting Rule  TRUE      TRUE      TRUE
## 3 ranger min.node.size                                Minimal Node Size TRUE      TRUE      TRUE
```

Mtry is a number of variables to possibly split at in each node. Splitrule is a splitting rule. Min.node.size is a minimal node size. In the previous random forest model, its mtry is 4, we used Gini index as a splitting rule, and minimal node size is 1. In the following model, we compare mtry ranging from 3 to 10 in terms of accuracy.

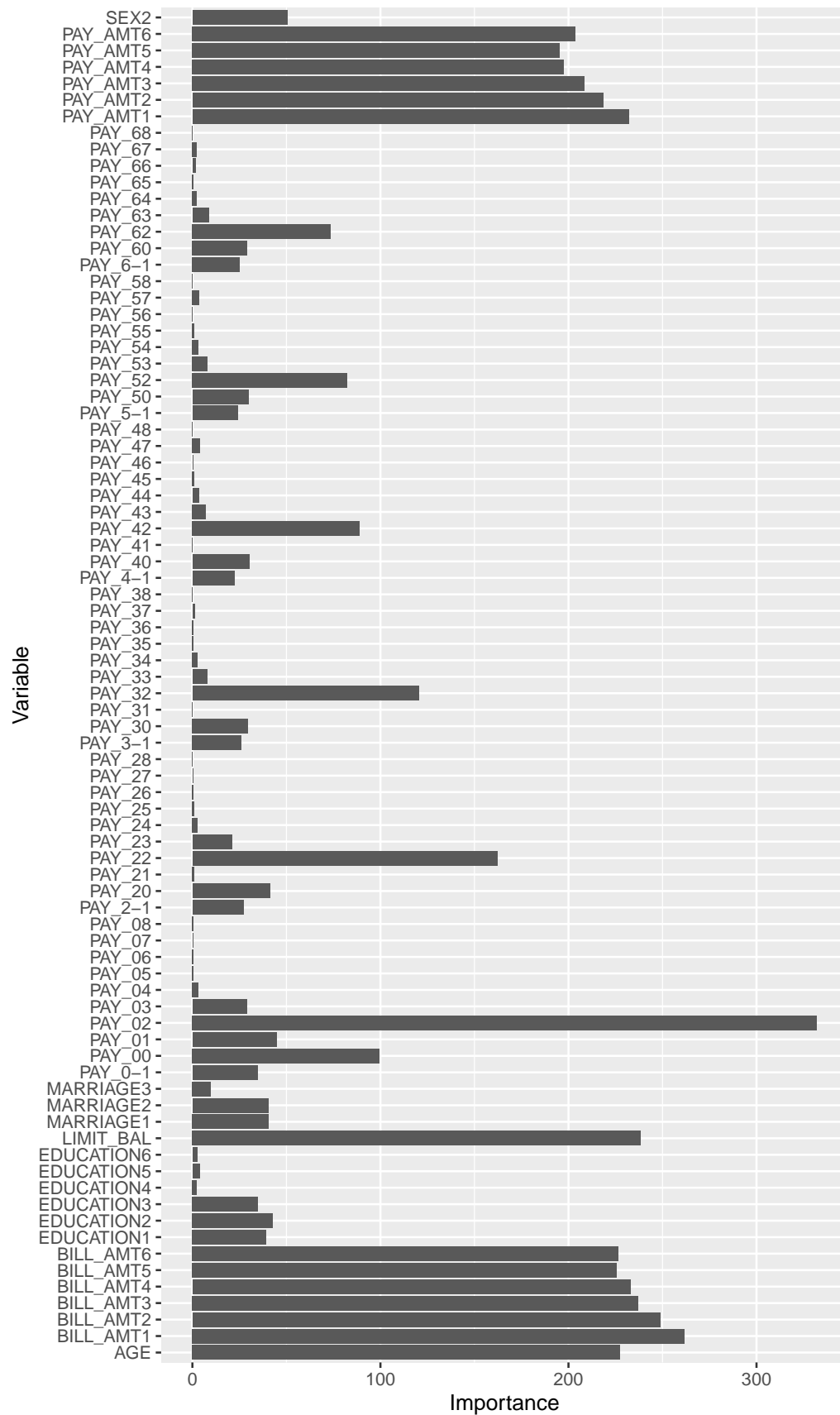
```
## Random Forest
##
## 19198 samples
##   23 predictor
##   2 classes: '0', '1'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17278, 17279, 17278, 17278, 17279, 17278, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##   3     0.8089386 0.2799217
##   4     0.8155016 0.3355440
##   5     0.8177415 0.3582670
##   6     0.8182623 0.3620446
##   7     0.8187315 0.3660563
##   8     0.8182623 0.3655969
##   9     0.8180020 0.3658315
##  10     0.8187312 0.3700717
##
## Tuning parameter 'splitrule' was held constant at a value of gini
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 7, splitrule = gini
## and min.node.size = 1.
```

Plot mtry and accuracy.



Plot the variables' importance;



Contrary to the previous model, not PAY\_0 but PAY\_02's importance is outstanding.

Then fit the model to the validation set and show confusion matrix;

```
##           Reference
## Prediction    0    1
##           0 3578  725
##           1  161  337
```

Other statistic metrics;

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
random forest tuned	0.8154551	0.9569404	0.3173258	0.6371331

Comparing these two random forest models, the results of the tuned model gets worse than the default model in terms of both accuracy and balanced accuracy. There occurred over fitting in the tuned model.

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
random forest	0.8181629	0.9531960	0.3427495	0.6479728
random forest tuned	0.8154551	0.9569404	0.3173258	0.6371331

## Evaluation

We look at the table in which the results of our 6 models are shown.

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
glm step wise	0.8154551	0.9539984	0.3276836	0.6408410
glm fewer features	0.8175380	0.9636266	0.3032015	0.6334141
rpart	0.8171214	0.9620219	0.3069680	0.6344950
rpart tuned	0.8162883	0.9593474	0.3126177	0.6359826
random forest	0.8181629	0.9531960	0.3427495	0.6479728
random forest tuned	0.8154551	0.9569404	0.3173258	0.6371331

Among them, the best performance in terms of both accuracy and balanced accuracy is produced by “**random forest default model**”. As we mentioned before, **PAY\_0** is the most important variables compared with others. Then follows **BILL\_AMT1**, **PAY\_AMT1**, and **LIMIT\_BAL**, and **AGE**.

We fit this model to the test set and extract final results.

Final statistic metrics are as follows;

method	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
final random forest	0.8226962	0.9621228	0.3320783	0.6471006

Fitting the model to the test set, we get **accuracy 0.8226962** and **balanced accuracy 0.6471006**. This results outperformed considerably our first guess (guessing all outcomes are 0), accuracy 0.7788 and balanced accuracy 0.5.

## Conclusion

### Summary of Our Research

In this paper, we used an open data, “Default of Credit Card Clients Dataset” and explored its information seeking for insights which contributed to the model making. We used three methods, logistic regression, decision tree, and random forest. In each method, we tried to improve the results by changing variables and tuning hyperparameters.

Random forest algorithm showed the best performance. Fitting it to the test set, we got accuracy 0.8226962 and balanced accuracy 0.6471006. But when tuned, the model showed over-fitted results. The logistic regression models outperform the decision tree models. As well as the random forest models, the default decision tree model showed better performance than the tuned model.

### Future Study

As we use rather simple methods in the paper, we recognized our limitations in terms of dealing with the variables. Basically, we formed logistic regression model assuming that independent variables are not highly correlated with each other. We eliminated some of the variables according to some insights from our exploration. But we need to use more rigorous methods regarding the variables. Tweaking the variables or regularization should be taken into consideration. Moreover, the variables, PAY, PAY\_AMT, and BILL\_AMT, are a kind of historical data over a half year. If we can look into these historical changes further, we might be able to acquire more accurate results.

From a business point of view, overall economic condition may play a very important role in predicting people’s financial behavior. A case in point, many people could not afford to pay their loan in the late 2000s due to the financial crisis at that time. On the other hand, when economy is booming, people repay the debt easily. Our dataset did not have such information, but if we want to make a more credible and practical model, we need to consider what kind of data is offered.