HarvardX Data Science Capstone Project Submission: Credit Card

SIU LUNG DAVID LAW

1/7/2020

Section 1: Overview and Executive Summary

The purpose of this project is to create a machine learning system to predict default based on a data-set on Taiwan credit card clients from April 2005 to September 2005. The data set comes from the UCI Machine Learning Repository Irvine, CA: University of California, School of Information and Computer Science (https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients#). The data-set is originally contained in an excel worksheet with 30000 entries and 25 data-fields with the official explanation as following:

Columns	Field	Keys
1	ID	ID of each entry
2	LIMIT_BAL	Amount of given credit (NT dollar) including both the individual consumer credit and his/her family (supplementary) credit.
3	SEX	1 = male; 2 = female
4	EDUCATION	1 = graduate school; 2 = university; 3 = high school; 4 = others
5	MARRIAGE	1 = married; 2 = single; 3 = others
6	AGE	Age in year
7-12*	PAY_0, PAY_2-PAY_6	History of past payment: PAY_0 = repayment status in Sept 2005; PAY_2 = repayment status in August 2005… PAY_6 = repayment status of April 2005 with -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months;.. ; 8 = payment delay for eight months; 9 = payment delay for nine months and above
13-18	BILL_AMT1 - BILL_AMT6	Amount of bill statement (NT dollar). BILL_AMT1 = bill amount in Sept 2005; BILL_AMT2 = bill amount in Aug 2005; …; BILL_AMT6 = bill amount in April 2005.
19-24	PAY_AMT1 - PAY-AMT6	Amount of previous payment (NT dollar). PAY_AMT1 = amount paid in September, 2005; PAY_AMT2 = amount paid in August, 2005; …; PAY_AMT6 = amount paid in April, 2005
25	default	1 = default payment in the next month, 0 = no default payment in the next month

- For some unexplained reason, the field name for Sept 2005 payment status is called PAY_0 rather than the more logical PAY_1.

In this project, we will examine the data in this data-set in details by looking at their statistics and also using visualization techniques. Also, we will propose six different models and compare their accuracy. Among these six prediction models, with one of them is a proprietary model created by me and another one is a ensemble model based on that proprietary model. The rest of the models are machine learning models from the caret package.

A key issue we have encountered in our research is there is no formal definition of default provided along with the data-set. Default can mean many things. One conventional definition is failure to pay credit card bill for six consecutive months. However, based on our data observation, this appears not the data-set default does not follow this definition. Also, it is unclear if default is only related to the payment of credit card bill. For example, it is unclear if a person is considered as default if he has no credit card bill or pays all his credit card bill on time but files bankruptcy due to his other liabilities. Without this clear definition, it becomes very difficult to connect what we are modeling here with reality.

Still, we manage to make the following findings:

- The data may need further cleaning
  - There are many entries without proper documentation. For example, there are significant number of PAY_0 entries with unexplained status 0 or -2.
  - The logic of many entries are inconsistent to the official explanation. For example, there are entries where the BILL_AMT1 is 0 but somehow default in the next month. In another example, we find there are abnormal movement in the payment status data. We see there are significantly more people in "two month payment delay" in August 2005 than "one month payment delay" in July 2005. This should be impossible as for a person to be two month delay in payment, he has to be one month delay in the previous month first.
  - All columns seem to be significantly dependent on each other, even for demography data such as SEX and MARRIAGE. And this is not what we would expect.
- PAY_0 is clearly the most important features for predicting default
- Ignoring all the above deficiencies, we manage to produce 6 models to predict default with most of them have accuracy above 77.8% (the level where assuming there is no default for everyone):

Models	Accuracy on train data	Accuracy on test data
Proprietary	82.1%	81.7%
Naive Bayesian	80.3%	80.2%
KNN (k=60)	77.7%	78.1%
Random Forest (mtry=3)	81.7%	82.3%
Logistic	73%	82.1%
Ensemble	85.6%	82.1%

In the following section, we will detail the methodology we use to arrive at these conclusions. We will only show the visualization output but not the codes in this report, as they are very long. Interested users can review the codes in the separate R code files. Also, it takes many hours or even days for my computer (iMac: Retina 5K, 27-inch, 2019; Processor: 3 GHz 6-Core Intel Core i5; Memory: 8 GB 2667 MHz DDR4) to generate the models. Readers of this report may keep this in mind if they want to run the codes in the R code files. If I have more computing power, I may be able to run a more extensive parameter calibration of the Random Forest and KNN models, resulting in more accurate models.

Section 2: Methods and Analysis

Our analysis will follow the steps below, and this section and the R codes file will be indexed in the same manner.

1. Preprocessing - This includes downloading file and install packages, importing file into data frame, changing classes of certain columns and participation file into train and test set.
2. Basic Investigation - We inspect the files and data entries to identify abnormalities and patterns.
3. Visualization - We further explore the relationship between different parameters and default.
4. Modeling - The six models mentioned above will be created and explained here.
5. Testing - These six models will then be tested using the test set.

Sub 2.1 Preprocessing

In this section, we will make the necessary steps we need to perform such as downloading data before doing the analysis.

Sub 2.1.A Downloading Data and Packages

As first step, we are going to download the file and relevant packages

```
#1.A Downloading Data and Packages
# Note: this process could take a couple of minutes
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(devtools)) install.packages("devtools", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(dplyr)
library(dslabs)
library(readxl)
library(ggpubr)

dl <- tempfile()
download.file("https://archive.ics.uci.edu/ml/machine-learning-databases/00350/default%20of%20credit%20
```

Sub 2.1.B Importing Data into data frame

The next step is to put these data into a data frame

```
#1.B Importing Data into data frame
xlsx_example <- read_excel(dl, range = cell_rows(c(2, NA)))
df = as.data.frame(xlsx_example)
```

Sub 2.1.C Changing columns from numeric to factors

In the imported data frame, all the data are numeric including SEX, MARRIAGE, EDUCATION, default status etc. However, these should be categorical rather than numerical. Therefore, we should change their classes to factors with the following codes.

```
#1.C Changing columns from numeric to factors
sapply(df, class)
```

```
##                        ID                 LIMIT_BAL
##                 "numeric"                 "numeric"
##                       SEX                 EDUCATION
##                 "numeric"                 "numeric"
##                  MARRIAGE                       AGE
##                 "numeric"                 "numeric"
##                     PAY_0                     PAY_2
##                 "numeric"                 "numeric"
##                     PAY_3                     PAY_4
##                 "numeric"                 "numeric"
##                     PAY_5                     PAY_6
##                 "numeric"                 "numeric"
##                 BILL_AMT1                 BILL_AMT2
##                 "numeric"                 "numeric"
##                 BILL_AMT3                 BILL_AMT4
##                 "numeric"                 "numeric"
##                 BILL_AMT5                 BILL_AMT6
##                 "numeric"                 "numeric"
##                  PAY_AMT1                  PAY_AMT2
```

```
##                 "numeric"                  "numeric"
##                  PAY_AMT3                   PAY_AMT4
##                 "numeric"                  "numeric"
##                  PAY_AMT5                   PAY_AMT6
##                 "numeric"                  "numeric"
## default payment next month
##                 "numeric"
```

```
df$ID<-as.factor(df$ID)
df$SEX<-as.factor(df$SEX)
df$EDUCATION<-as.factor(df$EDUCATION)
df$MARRIAGE<-as.factor(df$MARRIAGE)
df$PAY_0<-as.factor(df$PAY_0)
df$PAY_2<-as.factor(df$PAY_2)
df$PAY_3<-as.factor(df$PAY_3)
df$PAY_4<-as.factor(df$PAY_4)
df$PAY_5<-as.factor(df$PAY_5)
df$PAY_6<-as.factor(df$PAY_6)
names(df)[25]<-"default"
df$default<-as.factor(df$default)
```

After these changes, the following shows the classes of each columns

```
sapply(df, class)
```

```
##        ID LIMIT_BAL       SEX EDUCATION  MARRIAGE       AGE     PAY_0     PAY_2
##  "factor" "numeric"  "factor"  "factor"  "factor" "numeric"  "factor"  "factor"
##     PAY_3     PAY_4     PAY_5     PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4
##  "factor"  "factor"  "factor"  "factor" "numeric" "numeric" "numeric" "numeric"
## BILL_AMT5 BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##   default
##  "factor"
```

Sub 2.1.D Partitioning the date-set into test set and train set

The last step is to partition the data-set into test set and train set.

```
#1.D partitioning the dateset into test set and train set
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = df$default, times = 1, p = 0.1, list = FALSE)
train <- df[-test_index,]
test <- df[test_index,]
```

And we have completed the preprocessing.

Sub 2.2 Basic Investigation

Now we can start making some preliminary investigation of data.

Sub 2.2.A Calculating Basic Statistics

The following are the codes to run the basic statistics and the default rate of the whole train set. We also define the variables total_default, total_n_default, total_pop and default_prop to denote total number of defaults, total number of non-defaults, total number of population and the default ratio respectively.

```
#2.A Calculating Basic Statistics
summary(train)
```

```
##        ID          LIMIT_BAL       SEX       EDUCATION MARRIAGE       AGE
##  1       :     1   Min.   : 10000   1:10644   0:   12   0:   46   Min.   :21.00
##  2       :     1   1st Qu.: 50000   2:16355   1: 9522   1:12261   1st Qu.:28.00
##  3       :     1   Median :140000             2:12640   2:14407   Median :34.00
##  4       :     1   Mean   :167271             3: 4425   3:  285   Mean   :35.46
##  5       :     1   3rd Qu.:240000             4:  111             3rd Qu.:41.00
##  6       :     1   Max.   :800000             5:  244             Max.   :79.00
##  (Other):26993                                6:   45
##      PAY_0            PAY_2            PAY_3            PAY_4
##  0      :13204   0      :14129   0      :14200   0      :14844
##  -1     : 5128   -1     : 5434   -1     : 5323   -1     : 5112
##  1      : 3328   2      : 3536   -2     : 3694   -2     : 3921
##  -2     : 2493   -2     : 3426   2      : 3429   2      : 2810
##  2      : 2427   3      :  303   3      :  215   3      :  158
##  3      :  290   4      :   90   4      :   67   4      :   62
##  (Other):  129   (Other):   81   (Other):   71   (Other):   92
##      PAY_5            PAY_6          BILL_AMT1          BILL_AMT2
##  0      :15296   0      :14679   Min.   :-165580   Min.   :-69777
##  -1     : 4973   -1     : 5170   1st Qu.:   3519   1st Qu.:  2943
##  -2     : 4098   -2     : 4417   Median :  22303   Median : 21055
##  2      : 2329   2      : 2456   Mean   :  51150   Mean   : 49138
##  3      :  158   3      :  162   3rd Qu.:  67054   3rd Qu.: 63970
##  4      :   73   4      :   43   Max.   : 746814   Max.   :671563
##  (Other):   72   (Other):   72
##    BILL_AMT3         BILL_AMT4          BILL_AMT5         BILL_AMT6
##  Min.   :-157264   Min.   :-170000   Min.   :-81334   Min.   :-339603
##  1st Qu.:   2620   1st Qu.:   2329   1st Qu.:  1748   1st Qu.:   1240
##  Median :  20030   Median :  18992   Median : 18049   Median :  16928
##  Mean   :  46956   Mean   :  43226   Mean   : 40204   Mean   :  38772
##  3rd Qu.:  60036   3rd Qu.:  54509   3rd Qu.: 50144   3rd Qu.:  49117
##  Max.   :1664089   Max.   : 706864   Max.   :823540   Max.   : 699944
##
##     PAY_AMT1         PAY_AMT2            PAY_AMT3            PAY_AMT4
##  Min.   :     0   Min.   :      0.0   Min.   :     0.0   Min.   :     0
##  1st Qu.:   990   1st Qu.:    827.5   1st Qu.:   393.5   1st Qu.:   291
##  Median :  2100   Median :   2004.0   Median :  1800.0   Median :  1500
##  Mean   :  5654   Mean   :   5832.1   Mean   :  5178.8   Mean   :  4780
##  3rd Qu.:  5004   3rd Qu.:   5000.0   3rd Qu.:  4507.5   3rd Qu.:  4003
##  Max.   :873552   Max.   :1684259.0   Max.   :889043.0   Max.   :621000
##
##     PAY_AMT5          PAY_AMT6         default
##  Min.   :     0   Min.   :     0.0   0:21027
##  1st Qu.:   258   1st Qu.:   115.5   1: 5972
##  Median :  1500   Median :  1500.0
##  Mean   :  4786   Mean   :  5230.6
##  3rd Qu.:  4058   3rd Qu.:  4000.0
##  Max.   :426529   Max.   :528666.0
##
```

```r
nrow(train)
```

```
## [1] 26999
```

```r
total_default<-sum(train$default==1) # total default number
total_n_default<-sum(train$default==0) #total non-default number
```

```
total_pop<-total_default+total_n_default #total population
default_prop<-total_default/(total_default+total_n_default) #ratio of default
default_prop
```

```
## [1] 0.2211934
```

From the above, we observe the following:

- There are 26999 rows of data, and among these there are 5972 defaults.
- The default rate for the train set is 22.1%.
- There are many data with status that are undocumented from the data provider:
  - EDUCATION status as 0, 4, 5, 6
  - MARRIAGE status as 0
  - PAY_X status as 0 and -2, and the numbers of data with such status are very significant

Also, as a relatively minor point, there is no column with name "PAY_1" and there are only "PAY_0" and "PAY_2". According to official data description, PAY_0 is supposed to represent payment status of SEP 2005 while PAY_2 represents payment status of AUG 2005 and there is no gap month in between.

Sub 2.2.B Identifying and Investigating Undocumented Data

The following codes allow us to make some basic investigation of undocumented data

The first is to evaluate the EDUCATION status against LIMIT_BAL.

```
#2.B Identifying and Investigating Undocumented Data
train%>% group_by(EDUCATION) %>%
  summarize(n=n(), Q_1=quantile(LIMIT_BAL,0.25), median = median(LIMIT_BAL), mean = mean(LIMIT_BAL),
            Q_3=quantile(LIMIT_BAL,0.75)) #No clear pattern for the undocuemnted Education
```

```
## # A tibble: 7 x 6
##   EDUCATION     n    Q_1 median    mean    Q_3
##   <fct>     <int>  <dbl>  <dbl>   <dbl>  <dbl>
## 1 0            12 190000 215000 218333. 260000
## 2 1          9522 100000 200000 213255. 300000
## 3 2         12640  50000 110000 146782. 210000
## 4 3          4425  50000  80000 126054. 180000
## 5 4           111 150000 200000 221802. 280000
## 6 5           244  67500 150000 160557. 220000
## 7 6            45  30000 100000 133556. 200000
```

The above table shows different EDUCATION status against LIMIT_BAL. We would expect generally speaking higher education would imply higher credit rating and thus higher LIMIT_BAL. The above table confirms this understanding, as LIMIT_BAL for EDUCATION status 1 is better than 2, and 2 is better than 3. EDUCATION status 4 is "considered as"others" according to the official definition, and apparently this does not necessarily mean lower education than status 3, which is high school. Also, EDUCATION status 0, 5 and 6 are undocumented. From the stat above, it is not clear roughly where do these refer to.

Next, we evaluate MARRIAGE against LIMIT_BAL:

```
#2.B Identifying and Investigating Undocumented Data
train%>% group_by(MARRIAGE) %>%
  summarize(n=n(), Q_1=quantile(LIMIT_BAL,0.25), median = median(LIMIT_BAL), mean = mean(LIMIT_BAL), Q_3
```

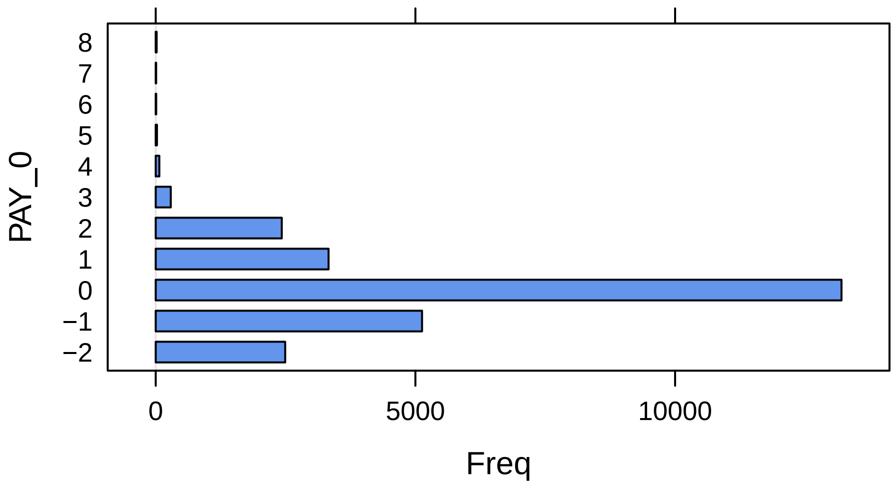
```
## # A tibble: 4 x 6
##   MARRIAGE     n   Q_1 median    mean    Q_3
##   <fct>    <int> <dbl>  <dbl>   <dbl>  <dbl>
## 1 0           46 72500 115000 136522. 200000
## 2 1        12261 70000 160000 182028. 260000
```

```
## 3 2         14407 50000 130000 156211. 220000
## 4 3           285 30000  50000  96491. 130000
```

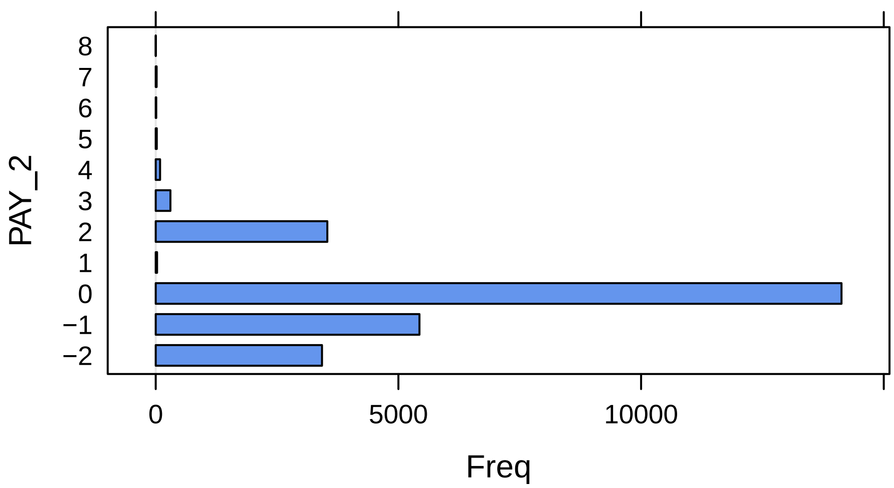
The above table shows different MARRIAGE status against LIMIT_BAL. According to the documentation, 1 represents married, 2 represents single, 3 represents others and there is no explanation of 0. Status 0 seems to be different to all these three status. Given the number of such entries are not significant, we are not too worried about this.

Next we will investigate the pay status with the following charts. Please note status 0 and -2 are not documented:

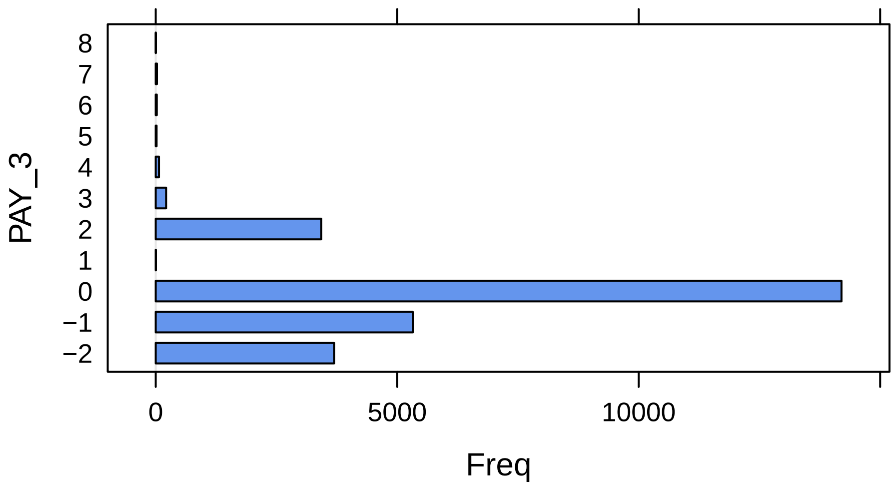
```r
#2.B Identifying and Investigating Undocumented Data
barchart(train$PAY_0, ylab="PAY_0", col ="cornflowerblue")
```



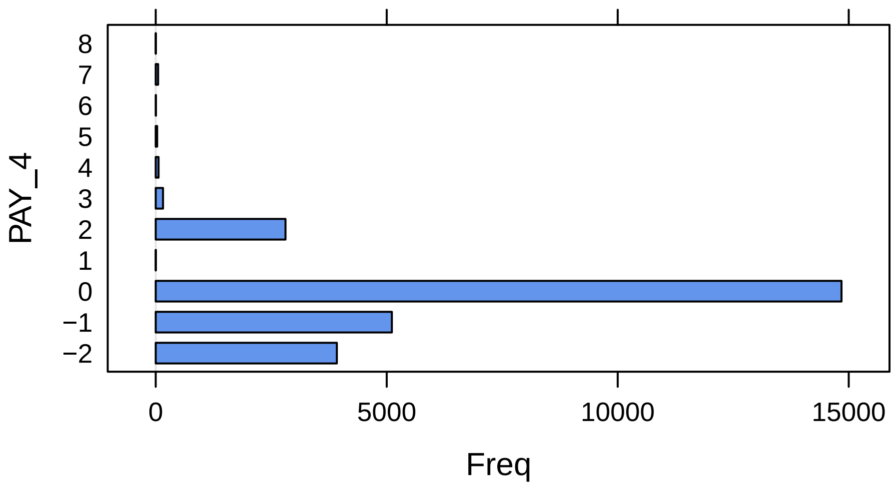

```r
barchart(train$PAY_2, ylab="PAY_2", col ="cornflowerblue")
```




```r
barchart(train$PAY_3, ylab="PAY_3", col ="cornflowerblue")
```




```r
barchart(train$PAY_4, ylab="PAY_4", col ="cornflowerblue")
```



From these graphs, it appears the distribution of different status for PAY_0, PAY_2, PAY_3 and PAY_4 are similar except that PAY_0 seems to have significantly more status 1 compared to others. In fact, the status 1 figures in PAY_2, PAY_3, PAY_4 and PAY_5 appear to be abnormal. As we may recall, status 2 in PAY_2 represents one has payment delay for 2 months in Aug 2005. For this to happen, there must be one month delay in July 2005 (i.e. status 1 in PAY_3). However, we see that status 1 in PAY_3 is much fewer than status 2 in PAY_2, and this is illogical. The same abnormality also occurs between PAY_3 and PAY_4.

Next, we investigate the transition of status between PAY_0 and PAY_2.

```r
table(train$PAY_0, train$PAY_2)
```

```
##
##         -2    -1     0     1     2     3     4     5     6     7     8
##   -2  2318   172     0     0     3     0     0     0     0     0     0
##   -1     0  4184   548     0   345    44     4     3     0     0     0
##   0      0   416 12788     0     0     0     0     0     0     0     0
##   1   1108   552     2    28  1496   102    30     6     2     1     1
##   2      0   110   791     0  1449    65    12     0     0     0     0
##   3      0     0     0     0   243    40     6     1     0     0     0
##   4      0     0     0     0     0    52    15     3     0     0     0
##   5      0     0     0     0     0     0    23     0     1     0     0
##   6      0     0     0     0     0     0     0     9     0     0     0
```

```
## 7       0     0     0     0     0     0     0     0     8     0     0
## 8       0     0     0     0     0     0     0     0     0    18     0
```

From this table, nearly all PAY_0 status -2 either come from PAY_2 status of -2 or -1. As -1 means one has duly paid, then -2 seems to represent even a better status as the source of -2 are better than -1. So, we presume -2 represents something like "duly pay for a long time". On the other hand, comparing the transition from PAY_2 status of 0 and -1, we find that although both these can transit to PAY_0 status of -1, 0, 1 ,2, we find PAY_2 status of 0 generally transit to a "worse" status of PAY_0. Therefore, we conjecture status 0 represents something that is worse than duly pay (status -1) but is better than delay in payment, which is something like making the "min pay".

This table also shows the abnormality of lack of status 1 for PAY_2. We also see significant of PAY_2 status 2 transit to status 1 or 2 in PAY_0. This should be impossible for one to have a 2 month payment delay in AUG 2005 to transit to 1 month payment delay or 2 month delay in Sept 2005.

We can make another transition table between PAY_2 and PAY_3 and the same pattern emerges.

```
table(train$PAY_2, train$PAY_3)
```

```
##
##          -2    -1     0     1     2     3     4     5     6     7     8
##    -2  3063   359     1     0     3     0     0     0     0     0     0
##    -1   354  4200   521     0   345    14     0     0     0     0     0
##    0    215   483 12659     0   744    24     3     1     0     0     0
##    1     12     9     1     4     2     0     0     0     0     0     0
##    2     50   272  1018     0  2067    80    18     4     2    23     2
##    3      0     0     0     0   268    23     9     1     1     1     0
##    4      0     0     0     0     0    74    15     1     0     0     0
##    5      0     0     0     0     0     0    22     0     0     0     0
##    6      0     0     0     0     0     0     0    11     0     0     0
##    7      0     0     0     0     0     0     0     0    19     0     0
##    8      0     0     0     0     0     0     0     0     0     1     0
```

Next, we find a few entries that are very interesting for further investigation:

```
train%>%filter(ID%in%c(110,122, 143, 149,574))
```

```
##    ID LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_0 PAY_2 PAY_3 PAY_4 PAY_5 PAY_6
## 1 110    360000   1         2        1  35     1    -2    -2    -2    -2    -2
## 2 122    450000   1         1        1  40     1    -2    -2    -2    -2    -2
## 3 143     50000   1         2        2  23     1     2     2     2     0     0
## 4 149     80000   2         2        1  23     1     2     3     2     0     0
## 5 574    160000   2         2        2  60    -2    -1    -1     0    -1    -1
##   BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6 PAY_AMT1 PAY_AMT2
## 1      -103      -103      -103      -103      -103      -103        0        0
## 2         0         0         0         0         0         0        0        0
## 3     10131     10833     20583     19996     19879     18065     1000    10000
## 4      9168     10522     10205      9898     10123     12034     1650        0
## 5      3128      5156      1089       489      3177      1009     5156     1089
##   PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6 default
## 1        0        0        0        0       0
## 2        0        0        0        0       1
## 3      400      700      800      600       0
## 4        0      379     2091        1       0
## 5        0     3177     1009        0       1
```

From the above, we can make a few interesting observations:

- ID 110 seems to suggest BILL_AMT_x corresponds to PAY_AMT_x+1

- ID 143 and 149 Pay status can jump from 0 to 2, which as said before should be against the explanation of documentation
- ID 149 Pay status can decrease from 3 to 2 and 2 to 1, which is against the explanation in the documentation
- ID 122 suggests one can default even if BILL_AMT1 =0, and this is counter-intuitive.
- ID 574 seems to suggest PAY_4 relates to BILL_AMT3 and PAY_AMT4

Based on the findings of ID 574, we run a statistics table among PAY_4, PAY_AMT3/BILL_AMT4

```
train%>%filter(BILL_AMT4!=0)%>%mutate(ratio=PAY_AMT3/BILL_AMT4)%>%
group_by(PAY_3)%>% summarise(median=median(ratio),mean=mean(ratio),
Q_10=quantile(ratio,0.1),Q_25=quantile(ratio,0.25),Q_75=quantile(ratio,0.75))
```

```
## # A tibble: 11 x 6
##    PAY_3 median      mean    Q_10   Q_25    Q_75
##    <fct>  <dbl>     <dbl>   <dbl>  <dbl>   <dbl>
##  1 -2    1      -39.5     0       1      1.00
##  2 -1    1        0.797   0       0.761  1
##  3 0     0.0468   0.0446  0.0288  0.0359 0.0929
##  4 1     0.0914   0.125   0.00807 0.0198 0.196
##  5 2     0.0406  -0.00404 0       0      0.0857
##  6 3     0        0.0233  0       0      0.0369
##  7 4     0        0.0129  0       0      0.00143
##  8 5     0        0.00341 0       0      0
##  9 6     0        0.00408 0       0      0
## 10 7     0        0.00414 0       0      0
## 11 8     0.0321   0.0321  0.00642 0.0161 0.0482
```

Previously, we conjecture status 0 probably represents one has made some "min pay". The above data seems to support that as the 0.1 quantile and 0.25 quantile of Payment Amount/Bill Amount ratio for status 0 is much higher than status 1.

Sub 2.2.C Check Independence

Finally, we want to check if the data are independent to each other. While we expect the default status or payment status should be dependent on other variables, we expect some variables such as SEX and MARRIAGE should be independent to each other. However, we find that all are dependent on each other, including these demographic variables. This is again counter-intuitive.

```
chisq.test(train$SEX, train$default)$p.value
```

```
## [1] 9.44661e-12
```

```
chisq.test(train$SEX, train$EDUCATION)$p.value
```

```
## [1] 9.970003e-06
```

```
chisq.test(train$SEX, train$MARRIAGE)$p.value
```

```
## [1] 3.968583e-07
```

```
chisq.test(train$SEX, train$AGE)$p.value
```

```
## [1] 6.491683e-37
```

```
chisq.test(train$SEX, train$PAY_0)$p.value
```

```
## [1] 5.882783e-20
```

```r
chisq.test(train$SEX, train$LIMIT_BAL)$p.value
```

```
## [1] 5.109402e-93
```

```r
chisq.test(train$EDUCATION, train$PAY_0)$p.value
```

```
## [1] 1.474267e-192
```

```r
chisq.test(train$EDUCATION, train$MARRIAGE)$p.value
```

```
## [1] 2.206001e-209
```

```r
chisq.test(train$PAY_0, train$PAY_2)$p.value
```

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

Sub 2.3 Visualization

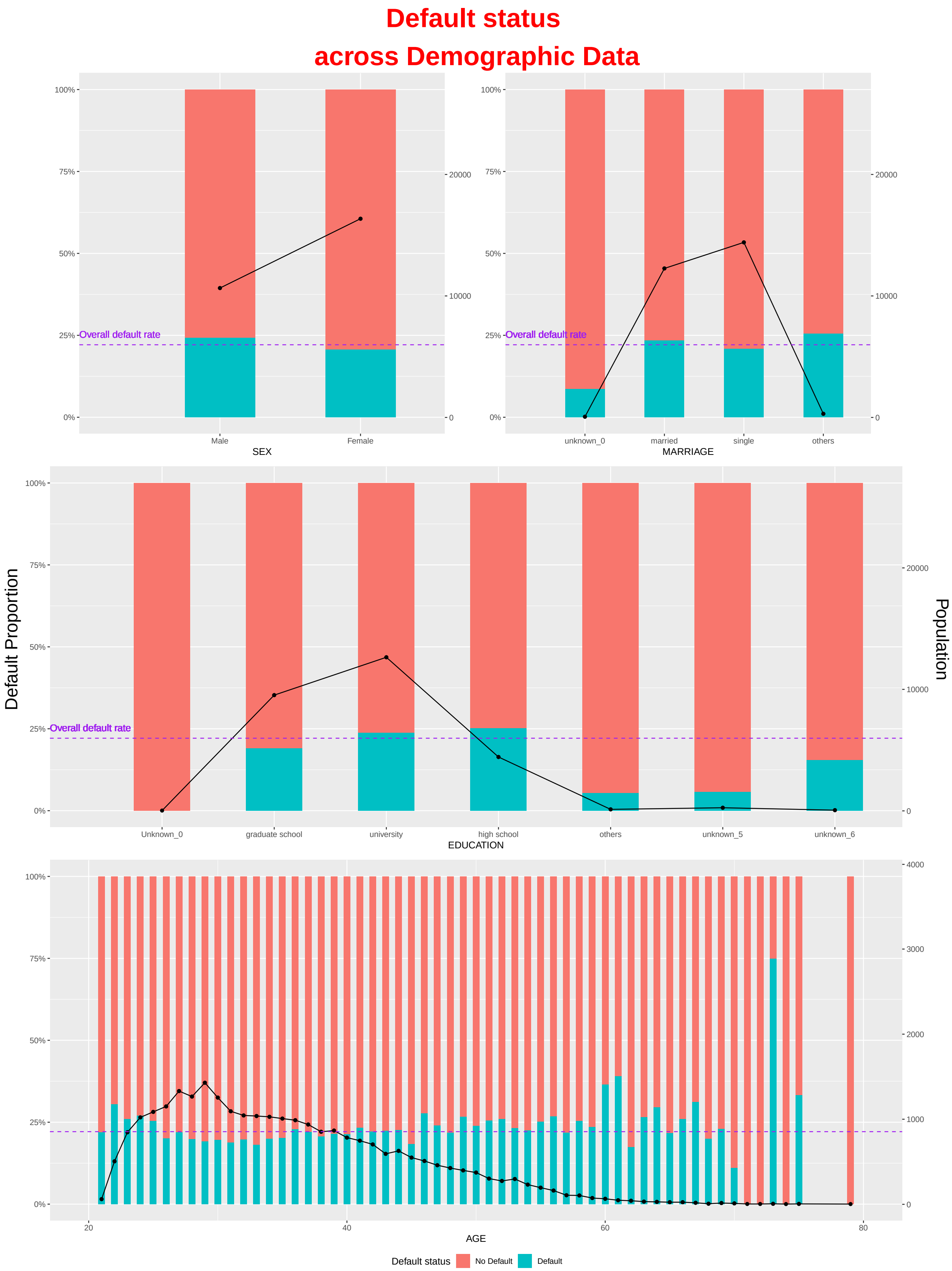
This section requires ggpubr library. The codes for generating the charts are very long, and so they will not be displayed in this pdf report. Interested readers can refer to the codes in the R codes file.

Sub 2.3.A Charting Demographic Data vs Default

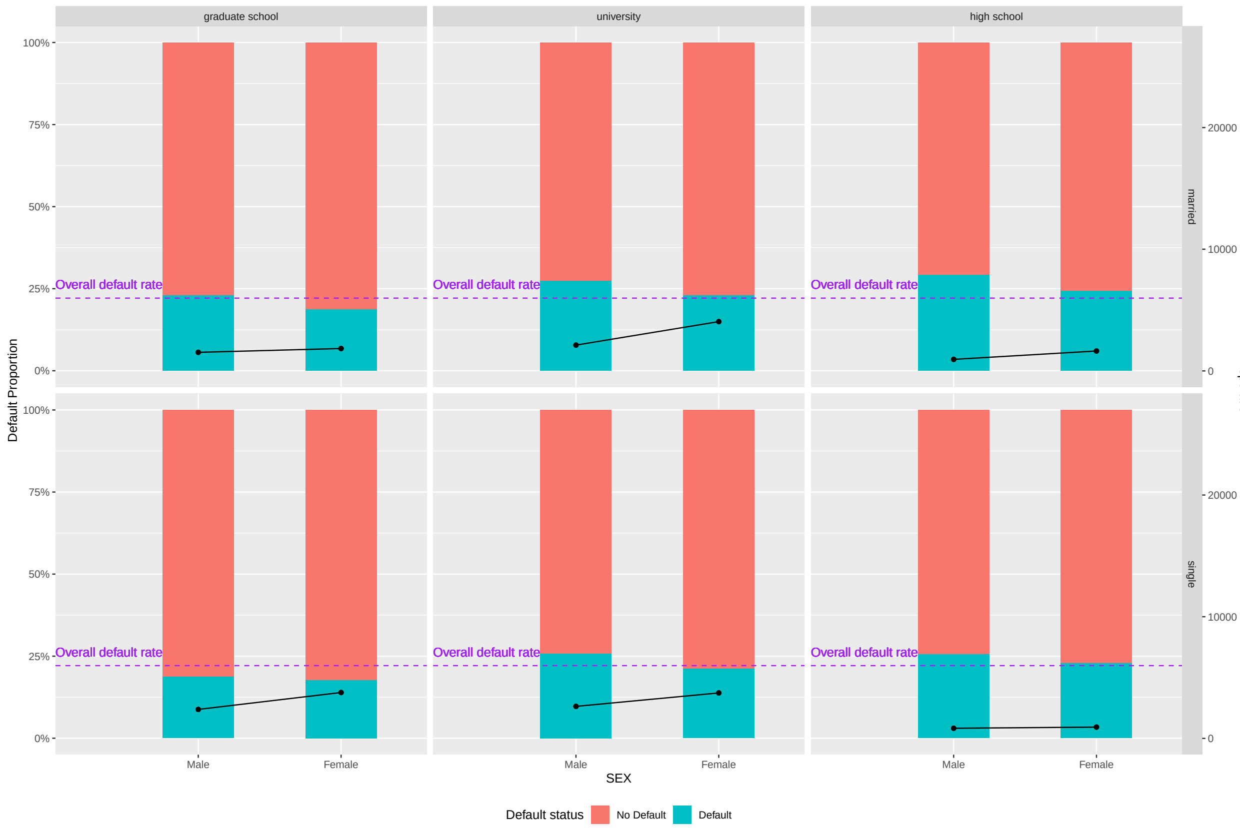
Firstly, we will start with charts visualizing demographic data against default. In the charts below, the blue part of the column charts represent default ratio, and the pink part represents non-default ratio. As one can only be either default or non-default, the total of these two add up to 100%. The black line represents the population in that category, and should be read against the axis on the right. For example, in the SEX diagram, we see there are roughly 10000 Male and 15000 Female. The blue line represents the default rate of the whole training set.

One of the purposes of these charts are to spot variables that can be used in the creation of proprietary model. Ideally, we should look for categories where firstly, the default rate significantly differs from the general population default rate and secondly, there are significant populations in these categories. It appears none of these variables or categories would satisfy these criteria. For example, even though EDUCATION = others default rate significantly differs from the general default rate, the population size of that category is very small. On the other hand, SEX = FEMALE appears to have a large population, but the default rate does not differ much from the overall default rate. Therefore, we are not going to use any of these variables in the proprietary model.

The second facet grid chart below allows us to perform a similar analysis, but on combination of these demographic data.

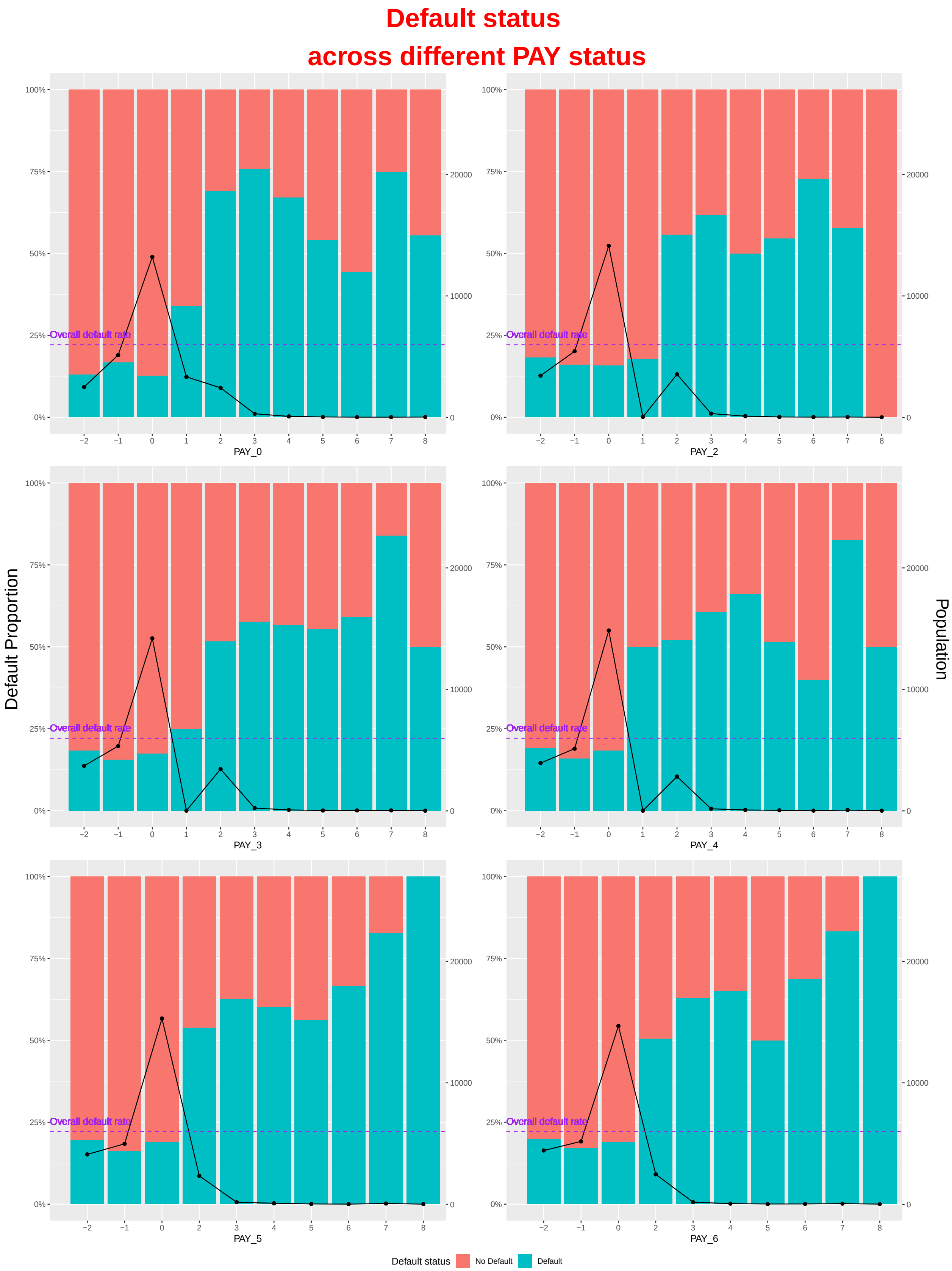


Default status
across Demographic Data
Default Proportion
Population
100%
75%
50%
25%
0%
20000
10000
0
Overall default rate
Male
Female
SEX
unknown_0
married
single
others
MARRIAGE
Unknown_0
graduate school
university
high school
others
unknown_5
unknown_6
EDUCATION
4000
3000
2000
1000
20
40
60
80
AGE
Default status
No Default
Default



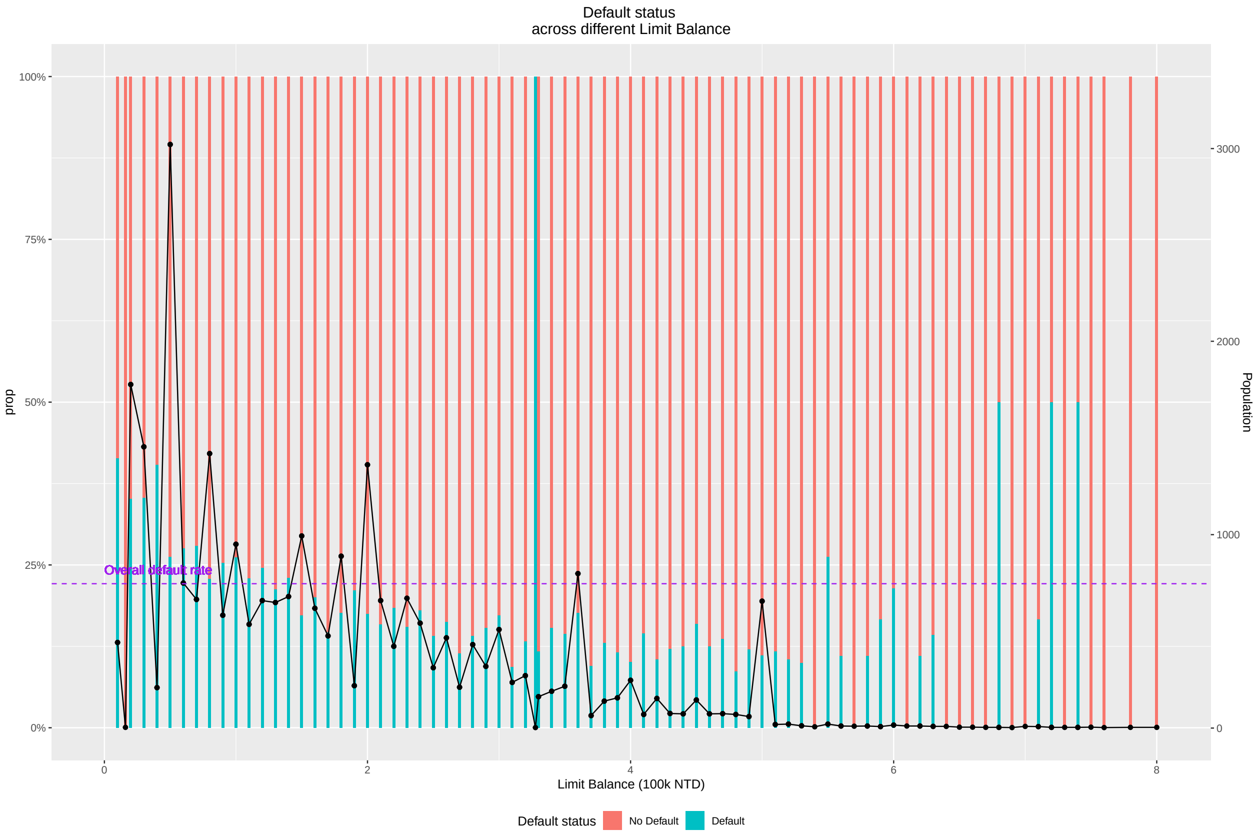
Sub 2.3.B Charting PAY status vs Default

We repeat the same exercise but on PAY status. These fields appear to be promising for building the proprietary model, as status -2, -1, 0, 1, 2 of PAY__0, status 2 of PAY__2 to PAY__6 all satisfy the criteria mentioned above.


Default status
across different PAY status
Default Proportion
Population
Overall default rate
PAY_0
PAY_2
PAY_3
PAY_4
PAY_5
PAY_6
100%
75%
50%
25%
0%
20000
10000
0
Default status
No Default
Default

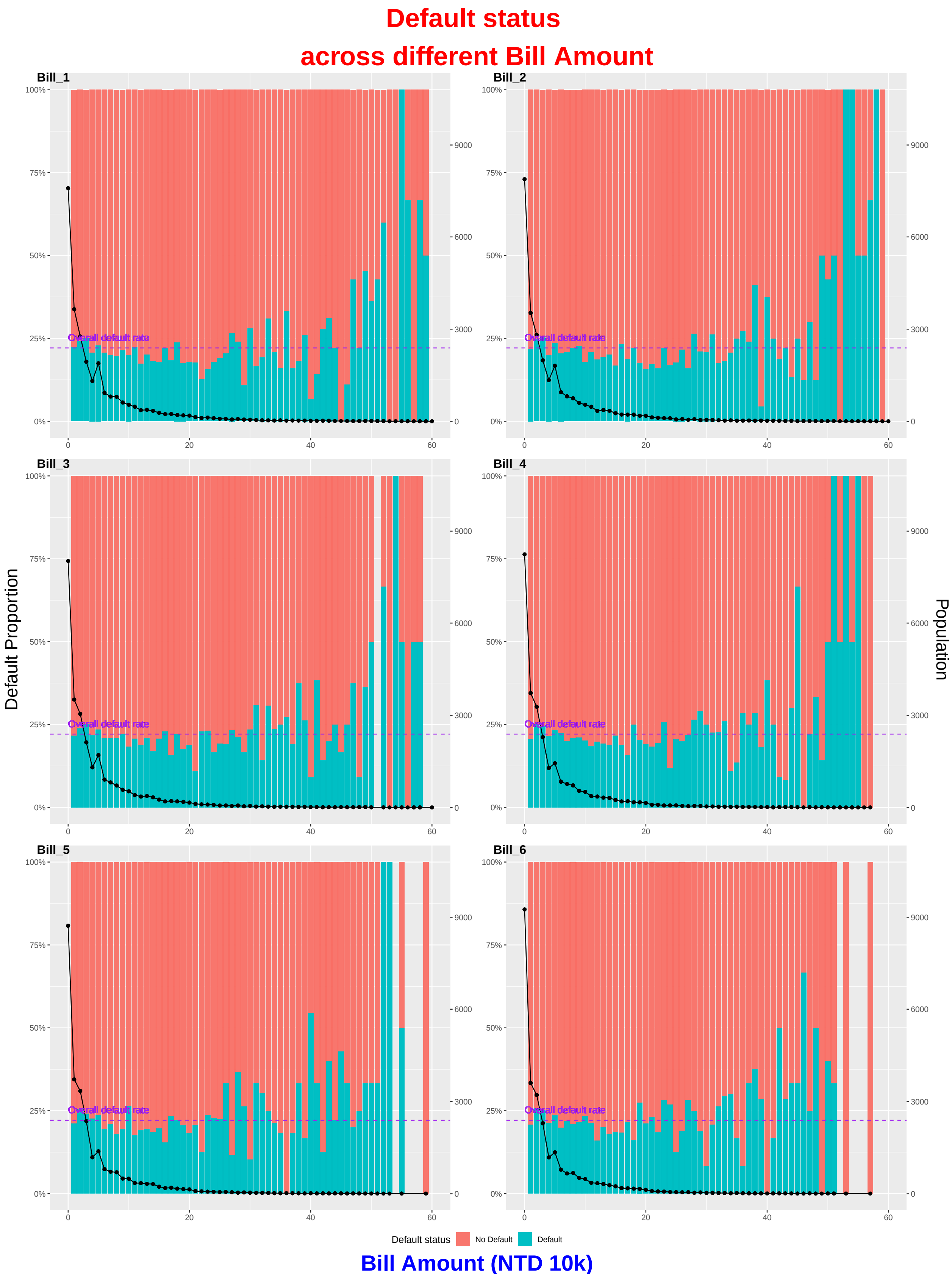
Sub 2.3.C Charting LIMIT_BAL vs Default

We repeat the same analysis between LIMIT_BAL and default. This field may potentially be useful in building our proprietary model, but it appears they are not as strong as the PAY status.



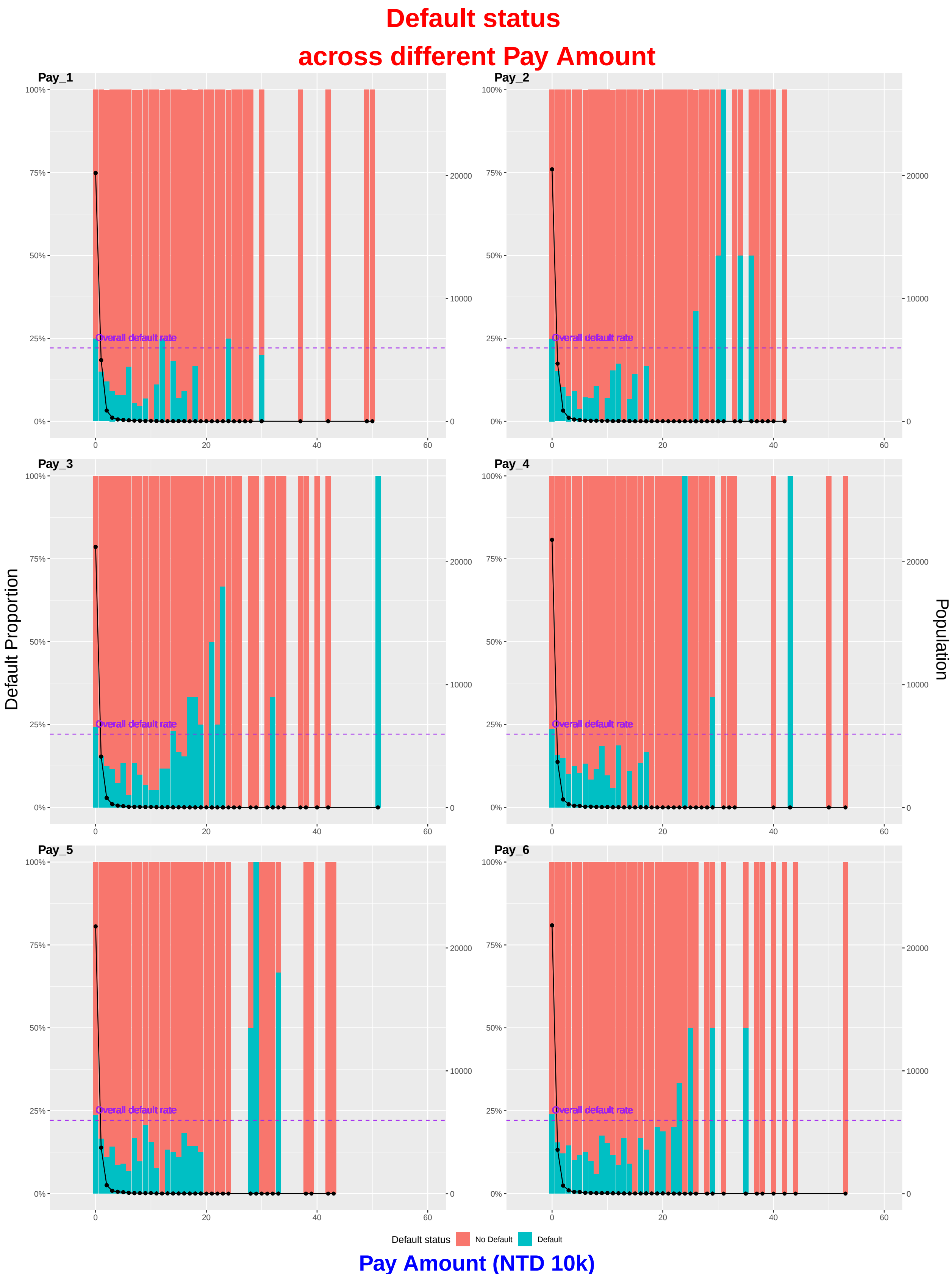
Sub 2.3.D Charting BILL_AMT vs Default

We repeat the same analysis between BILL_AMT and default. These fields may potentially be useful in building our proprietary model, but it appears they are not as strong as the PAY status.


Default status
across different Bill Amount
Bill_1
Bill_2
Bill_3
Bill_4
Bill_5
Bill_6
Overall default rate
Default Proportion
Population
Default status
No Default
Default
Bill Amount (NTD 10k)

Sub 2.3.E Charting PAY_AMT vs Default

The charts below shows that majority of the population pays less than NTD5000, and people in that category have a default rate that is slightly higher than the overall default rate. The rest who pays more than that will have a lower default rate than the average.


Default status
across different Pay Amount
Pay_1
Pay_2
Pay_3
Pay_4
Pay_5
Pay_6
Overall default rate
Default Proportion
Population
Default status
No Default
Default
Pay Amount (NTD 10k)

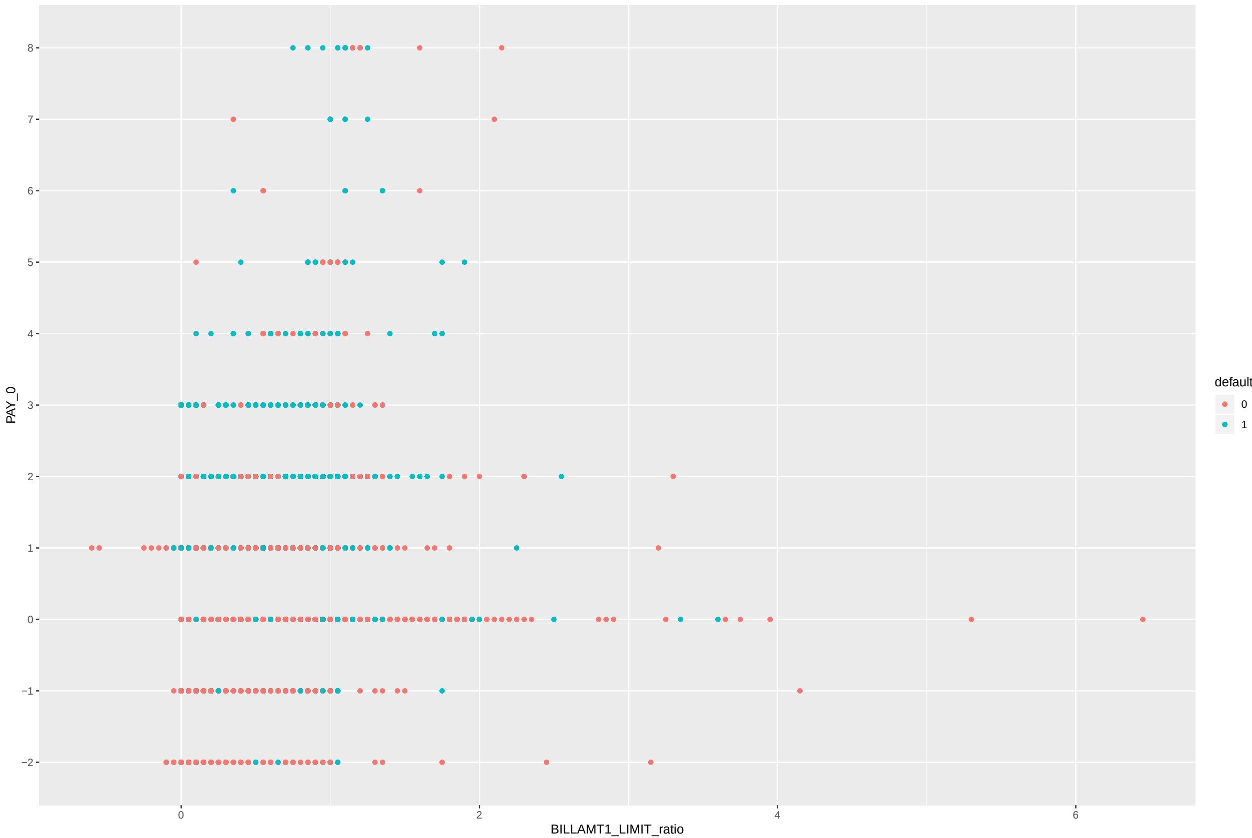
Sub 2.3.F Charting BILL_AMT LIMIT_BAL ratio vs Default

Next, we explore the relationship between BILL_AMT/LIMIT BAL and default rate. Intuitively, the LIMIT_BAL should proxy the maximum amount of money one can afford. If one has a bill that is very large compared to the LIMIT_BAL, then he is more likely to have a failure to pay resulting in default. The following charts shows there is quite a strong relationship between this ratio and the default rate.


Default status
across different Bill Amount divided by Limit Balance
Bill_1/Limit
Bill_2/Limit
Bill_3/Limit
Bill_4/Limit
Bill_5/Limit
Bill_6/Limit
Overall default rate
Default Proportion
Population
100%
75%
50%
25%
0%
20000
10000
0
0%
50%
100%
150%
200%
Default status
No Default
Default
Bill Amount/Credit Balance

Sub 2.3.G Charting BILL_AMT LIMIT_BAL ratio vs PAY status

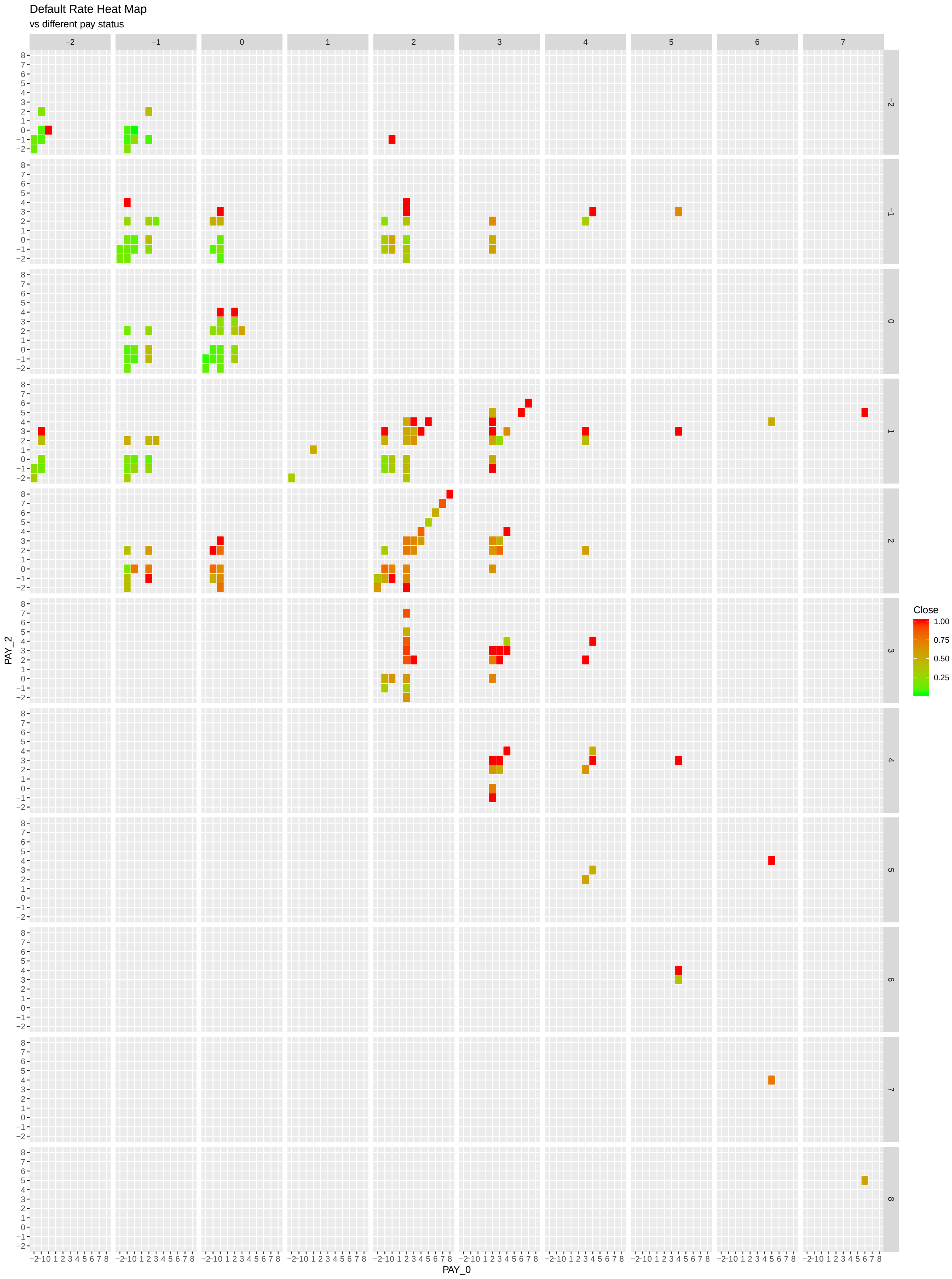
Next, we will explore the relationship among BILL_AMT LIMIT_BAL ratio, PAY status and default rate. It appears that majority of the default (blue dots) appear on the "up" side rather than the "right' side. This means the PAY status are much stronger predictor of default than the ratio.



Sub 2.3.H Charting different PAY status

Next, we use a facet heat map on PAY_0, PAY_2, PAY_3 and PAY_4 to visualize how these status affect the default rate. In the chart, each small block is a heat map with PAY_3 as x-axis and PAY_4 as y-axis. The big x-axis and y-axis are PAY_0 and PAY_1. From this heat map, we can see majority of the defaults happen when PAY_0 = 2 or 3. Also, we see some mainly "green" default in the upper left corner, representing small amount of default can happen when PAY_0 = -1 or -2. In those areas, majority of the default are in the block PAY_2 = 2. Moreover, in those blocks, we see that the darkest color are usually in the small pieces where PAY_3 or PAY_4 = 2.

From this heat map, therefore, we can infer that among all these PAY status, PAY_0 has the highest predictive power. PAY_2, PAY_3 and PAY_4 still have predictive power, but are much weaker.

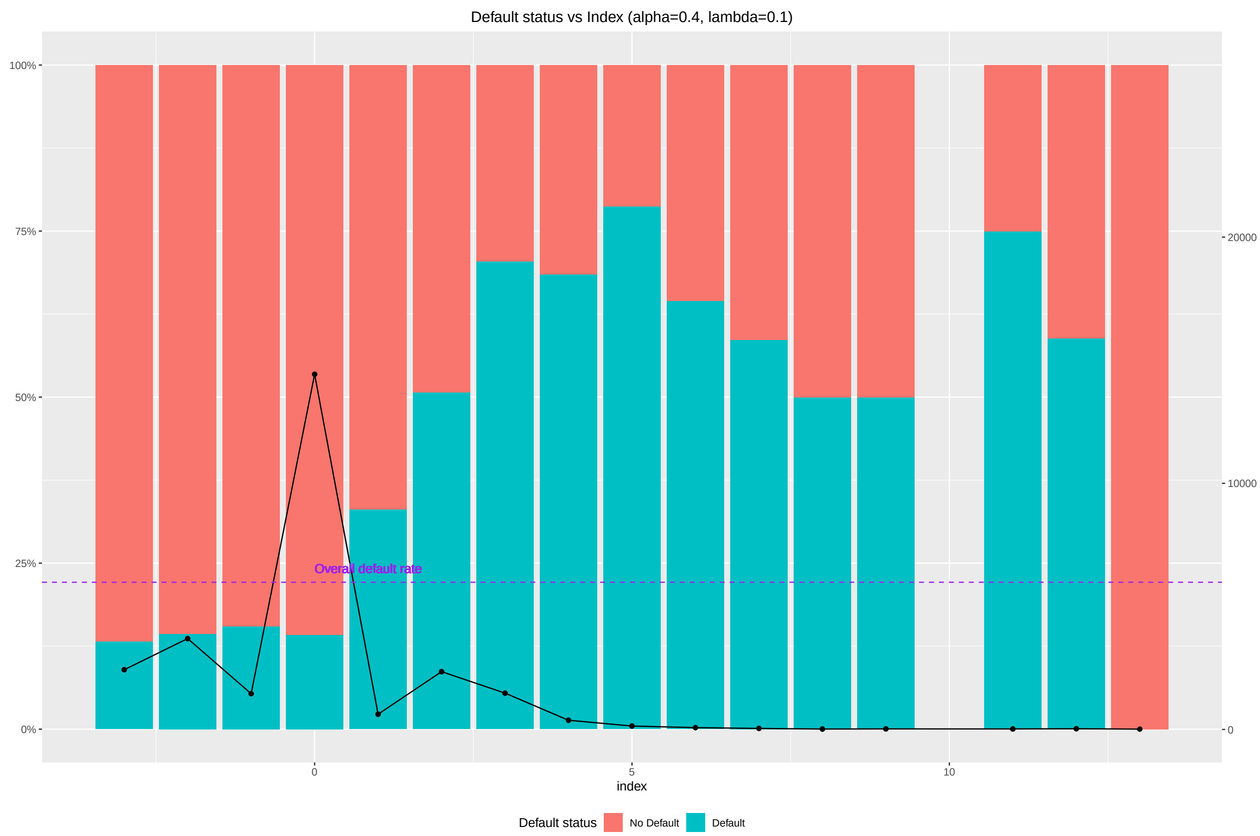

Default Rate Heat Map
vs different pay status
PAY_2
PAY_0
Close
1.00
0.75
0.50
0.25

Sub 2.3.I Charting Index vs default

Based on the visualization above, we propose an index I as

$$\begin{aligned} I = PAY_0 + \alpha PAY_2 + \alpha^2 PAY_3 + \alpha^3 PAY_4 + \alpha^4 PAY_5 + \alpha^5 PAY_6 + \\ \lambda(BILL_1 + \alpha BILL_2 + \alpha^2 BILL_3 + \alpha^3 BILL_4 + \alpha^4 BILL_5 + \alpha^5 BILL_6)/LIMIT \end{aligned} \tag{1}$$

Then we plot I against default assuming $\alpha = 0.4$ and $\lambda = 0.1$. The chart below seems to show this index have a strong positive relationship with default rate. We will use this I to create our proprietary model.



Sub 2.4 Modeling

In the sub-section, we will demonstrate a few machine learning models to predict default. Running of these models take hours or even days, and so readers should keep that in mind before trying these codes. And due to the very long time required in running the models, I have not completed a full search of best parameters should be used for KNN and Random Forest.

It is very interesting to note that many of these machine learning models considers PAY_0 as the most important modeling parameter.

Sub 2.4.A Proprietary Model

The first model to present here is the proprietary model mentioned above. From the above, we know that I of Equation (1)appears to be a strong indicator of default. However, it is not very clear what is the best parameter for α and λ. In the codes below, we will see how to maximize the model accuracy by trying different values of α and λ and see what threshold the Index has to be above in order to consider it as default.

```
#4.A Proprietary Model
Prop_Model<- function(alpha, threshold, lambda){
```

```r
p<-train %>%
  mutate(index1=(as.numeric(paste(PAY_0))+alpha*as.numeric(paste(PAY_2))+
  alpha^2*as.numeric(paste(PAY_3))+ alpha^3*as.numeric(paste(PAY_4))+
  alpha^4*as.numeric(paste(PAY_5))+alpha^5*as.numeric(paste(PAY_6))),
  index2=(BILL_AMT1+alpha*BILL_AMT2+alpha^2*BILL_AMT3+alpha^3*BILL_AMT4+
  alpha^4*BILL_AMT5+alpha^5*BILL_AMT6)/LIMIT_BAL,
  p=ifelse(index1+index2*lambda>=threshold, 1,0)) %>%
pull(p)
c(alpha, threshold, lambda, confusionMatrix(data=as.factor(p), reference=train$default)$overall["Accura
}

Prop_Model_Prediction<- function(dataset, alpha, threshold, lambda){
  p<-dataset%>%
    mutate(index1=(as.numeric(paste(PAY_0))+alpha*as.numeric(paste(PAY_2))+
    alpha^2*as.numeric(paste(PAY_3))+alpha^3*as.numeric(paste(PAY_4))+
    alpha^4*as.numeric(paste(PAY_5))+alpha^5*as.numeric(paste(PAY_6))),
    index2=(BILL_AMT1+alpha*BILL_AMT2+alpha^2*BILL_AMT3+alpha^3*BILL_AMT4+
    alpha^4*BILL_AMT5+alpha^5*BILL_AMT6)/LIMIT_BAL,
    p=ifelse(index1+index2*lambda>=threshold, 1,0)) %>%
    pull(p)
  as.factor(p)
#  confusionMatrix(data=as.factor(p), reference=dataset$default)
}

v1<-rep(seq(0,1,0.1),121)
v2<-rep(rep(seq(0,5,0.5),each=11), time=11)
v3<-rep(seq(0,1,0.1), each = 121)

P_Model_Calibration<-mapply(Prop_Model,v1, v2, v3)
P_Model_Calibration[, which(P_Model_Calibration[4,] == max(P_Model_Calibration[4,]), arr.ind = TRUE)]
```

```
##                                  Accuracy
## 0.2000000 1.5000000 0.0000000 0.8205489
```

We find that the optimal accuracy can be achieved by

$$\begin{aligned} \alpha &= 0.2 \\ Threshold &= 1.5 \\ \lambda &= 0 \end{aligned} \tag{2}$$

It turns out the bill limit ratio is not needed in the model. The accuracy of this model and also its confusion matrix, based on training data, are shown below:

```r
P_train<-Prop_Model_Prediction(train, 0.2,1.5,0)
confusionMatrix(data=P_train, reference=train$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 20061  3879
##          1   966  2093
##
##                Accuracy : 0.8205
##                  95% CI : (0.8159, 0.8251)
```

```
##     No Information Rate : 0.7788
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.369
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9541
##             Specificity : 0.3505
##          Pos Pred Value : 0.8380
##          Neg Pred Value : 0.6842
##              Prevalence : 0.7788
##          Detection Rate : 0.7430
##    Detection Prevalence : 0.8867
##       Balanced Accuracy : 0.6523
##
##        'Positive' Class : 0
##
```

Sub 2.4.B Naive Bayesian

The second model we propose is the Naive Bayesian. We will use the train function in the caret package to calibrate the model.

The accuracy of the model, based on training data, is shown below. Also, we will see below that this model considers PAY_0 as the most important modeling parameter.

```
model_nb
```

```
## Naive Bayes
##
## 26999 samples
##    23 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 24300, 24299, 24300, 24299, 24299, 24299, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.7080256  0.3126832
##    TRUE      0.8008075  0.3469814
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' 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 fL = 0, usekernel = TRUE and adjust
##  = 1.
```

```
varImp(model_nb)
```

```
## ROC curve variable importance
##
##   only 20 most important variables shown (out of 23)
```

```
##
##           Importance
## PAY_0        100.000
## PAY_2         72.390
## PAY_3         65.437
## LIMIT_BAL     61.785
## PAY_AMT1      57.648
## PAY_4         56.842
## PAY_AMT2      55.250
## PAY_5         51.477
## PAY_AMT3      50.037
## PAY_6         46.436
## PAY_AMT4      46.242
## PAY_AMT6      42.628
## PAY_AMT5      41.511
## EDUCATION     15.340
## SEX           12.398
## BILL_AMT1      8.794
## MARRIAGE       8.256
## BILL_AMT2      5.032
## BILL_AMT3      3.820
## BILL_AMT4      2.237
```

Sub 2.4.C KNN

The third model we want to try is the KNN. We have tried the k from 1 to 60 and find the accuracy continues to improve till 60. We think it is entirely plausible that the accuracy can continue to increase after 60. However, we stop it here as only run the case $k = 60$ here because the time to run the model calibration is too long. The R codes file have the codes to run from 1 to 60 and interested readers can run it from there.

```r
#4.C knn (takes days to complete)
Sys.time()
```

```
## [1] "2020-01-10 06:49:17 CST"
```

```r
set.seed(7, sample.kind = "Rounding")
#model_knn <- train(train[ ,2:24], train$default, method = "knn", tuneGrid = data.frame(k = seq(1,60)))
model_knn <- train(train[ ,2:24], train$default, method = "knn", tuneGrid = data.frame(k = 60))
Sys.time()
```

```
## [1] "2020-01-10 06:51:37 CST"
```

The accuracy of the model, based on training data, is shown below. Also, we will see below that this model considers PAY_0 as the most important modeling parameter.

```r
model_knn
```

```
## k-Nearest Neighbors
##
## 26999 samples
##    23 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 26999, 26999, 26999, 26999, 26999, 26999, ...
## Resampling results:
```

```
##
##   Accuracy   Kappa
##   0.7767771  0.0733482
##
## Tuning parameter 'k' was held constant at a value of 60
```

```
y_hat_knn_train <- predict(model_knn, train)
varImp(model_knn)
```

```
## ROC curve variable importance
##
##   only 20 most important variables shown (out of 23)
##
##            Importance
## PAY_0         100.000
## PAY_2          72.390
## PAY_3          65.437
## LIMIT_BAL      61.785
## PAY_AMT1       57.648
## PAY_4          56.842
## PAY_AMT2       55.250
## PAY_5          51.477
## PAY_AMT3       50.037
## PAY_6          46.436
## PAY_AMT4       46.242
## PAY_AMT6       42.628
## PAY_AMT5       41.511
## EDUCATION      15.340
## SEX            12.398
## BILL_AMT1       8.794
## MARRIAGE        8.256
## BILL_AMT2       5.032
## BILL_AMT3       3.820
## BILL_AMT4       2.237
```

Sub 2.4.D Random Forest

The model we are going to propose here is random forest. We have tried mtry from 3 to 9 and find the optimal is at 3. Similar to KNN, the time to run random forest is very long and so we stop at 3 and only run the case 3 here. The R codes file have the codes to run from 3 to 9 and interested readers can run it from there.

```
#4.D random forest
Sys.time()
```

```
## [1] "2020-01-10 06:51:51 CST"
```

```
set.seed(9, sample.kind = "Rounding")
model_rf <- train(train[ ,2:24], train$default, method = "rf",
#                   tuneGrid = data.frame(mtry = seq(3, 9, 2)),importance=TRUE)
tuneGrid = data.frame(mtry = 2),importance=TRUE)
model_rf
```

```
## Random Forest
##
## 26999 samples
##    23 predictor
```

```
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 26999, 26999, 26999, 26999, 26999, 26999, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8174875  0.3626835
##
## Tuning parameter 'mtry' was held constant at a value of 2
```

```r
Sys.time()
```

```
## [1] "2020-01-10 08:42:05 CST"
```

The accuracy of the model, based on training data, is shown below. Also, we will see below that this model considers PAY_0 as the most important modeling parameter.

```r
varImp(model_rf)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 23)
##
##           Importance
## PAY_0         100.00
## PAY_2          34.82
## LIMIT_BAL      29.74
## BILL_AMT1      28.11
## PAY_AMT3       28.02
## PAY_6          26.37
## PAY_3          25.72
## PAY_4          24.44
## PAY_AMT6       24.27
## BILL_AMT5      23.47
## PAY_AMT5       23.45
## PAY_AMT2       23.31
## PAY_AMT4       22.81
## BILL_AMT6      22.63
## PAY_5          22.61
## BILL_AMT3      22.57
## BILL_AMT4      21.40
## BILL_AMT2      20.87
## PAY_AMT1       16.11
## AGE            12.56
```

```r
y_hat_rf_train <- predict(model_rf, train)
```

Sub 2.4.E Logistic Regression

The model we present here are the logistic regression.

```r
#4.E glm
Sys.time()
```

```
## [1] "2020-01-10 08:42:09 CST"
```

```r
model_glm = train(train[ ,2:24], train$default,'glm')
Sys.time()
```

```
## [1] "2020-01-10 08:43:20 CST"
```

```r
y_hat_glm_train <- predict(model_glm, test, type = "raw")
```

The accuracy of the model, based on training data, is shown below. Also, we will see below that this model considers PAY_0 as the most important modeling parameter.

```r
model_glm
```

```
## Generalized Linear Model
##
## 26999 samples
##    23 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 26999, 26999, 26999, 26999, 26999, 26999, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8202123  0.3725399
```

```r
varImp(model_glm)
```

```
## glm variable importance
##
##   only 20 most important variables shown (out of 82)
##
##           Overall
## PAY_02     100.00
## PAY_03      63.06
## LIMIT_BAL   55.16
## PAY_01      50.33
## PAY_04      30.40
## PAY_AMT1    26.70
## SEX2        25.81
## PAY_0-1     25.09
## PAY_AMT2    22.12
## PAY_32      18.21
## PAY_05      18.05
## PAY_60      18.02
## MARRIAGE3   16.02
## MARRIAGE1   15.64
## MARRIAGE2   14.15
## PAY_24      14.02
## PAY_63      13.35
## PAY_33      12.29
## PAY_AMT6    11.33
## PAY_2-1     10.58
```

Sub 2.4.F Ensemble

Finally, we present the ensemble model here. We combine the two best models: proprietary and random forest. The ensemble model will only consider one will default if both model predicts he will default.

```
#4.F Ensemble
E_Prediction<-function(threshold, p_1,p_2,p_3){
  if(missing(p_3)){
    p<-ifelse(as.numeric(paste(p_1))+as.numeric(paste(p_2))>threshold, 1,0)
    as.factor(p)}
  else{
  p<-ifelse(as.numeric(paste(p_1))+as.numeric(paste(p_2))+as.numeric(paste(p_3))>threshold, 1,0)
as.factor(p)}
}
```

The accuracy of the model, based on training data, is shown below.

```
y_hat_E_train<-E_Prediction(1,y_hat_rf_train, P_train)
confusionMatrix(data=y_hat_E_train, reference=train$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 21020  3879
##          1     7  2093
##
##                Accuracy : 0.8561
##                  95% CI : (0.8518, 0.8602)
##     No Information Rate : 0.7788
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.456
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9997
##             Specificity : 0.3505
##          Pos Pred Value : 0.8442
##          Neg Pred Value : 0.9967
##              Prevalence : 0.7788
##          Detection Rate : 0.7785
##    Detection Prevalence : 0.9222
##       Balanced Accuracy : 0.6751
##
##        'Positive' Class : 0
##
```

Section 3: Testing

In this section, we will run the testing of the model use the test data set. To put this in context, we need to recognize two important background:

- The overall default rate is around 22 percent. An extremely basic model assuming there is no default at all will give us 78% of accuracy. Therefore, we would require any model to achieve at least 78 percent accuracy in order to be considered as useful.
- In the credit card industry, the profit is only a few percent of the bill if the credit card client does not default, but the loss is 100% of the bill in case of default. Therefore, the ability to detect default rather

than non-default is more important.

In the subsequent sub-sections, we will find out the accuracy and specificity of each model.

Sub 3.1 Proprietary Model

```
#5.A Proprietary Model
P_test<-Prop_Model_Prediction(test, 0.2,1.5,0)
confusionMatrix(data=P_test, reference=test$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2241  452
##          1   96  212
##
##                Accuracy : 0.8174
##                  95% CI : (0.8031, 0.8311)
##     No Information Rate : 0.7787
##     P-Value [Acc > NIR] : 1.049e-07
##
##                   Kappa : 0.3443
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9589
##             Specificity : 0.3193
##          Pos Pred Value : 0.8322
##          Neg Pred Value : 0.6883
##              Prevalence : 0.7787
##          Detection Rate : 0.7468
##    Detection Prevalence : 0.8974
##       Balanced Accuracy : 0.6391
##
##        'Positive' Class : 0
##
```

Sub 3.2 Naive Bayesian

```
#5.B naive bayseian
y_hat_nb <- predict(model_nb, test, type = "raw")
confusionMatrix(data=y_hat_nb, reference=test$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2155  412
##          1  182  252
##
##                Accuracy : 0.8021
##                  95% CI : (0.7874, 0.8162)
##     No Information Rate : 0.7787
##     P-Value [Acc > NIR] : 0.0009927
```

```
##
##                   Kappa : 0.3443
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9221
##             Specificity : 0.3795
##          Pos Pred Value : 0.8395
##          Neg Pred Value : 0.5806
##              Prevalence : 0.7787
##          Detection Rate : 0.7181
##    Detection Prevalence : 0.8554
##       Balanced Accuracy : 0.6508
##
##        'Positive' Class : 0
##
```

Sub 3.3 KNN

```r
#5.C knn
y_hat_knn <- predict(model_knn, test)
confusionMatrix(data = y_hat_knn, reference = test$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2292  619
##          1   45   45
##
##                Accuracy : 0.7787
##                  95% CI : (0.7635, 0.7935)
##     No Information Rate : 0.7787
##     P-Value [Acc > NIR] : 0.5104
##
##                   Kappa : 0.0703
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.98074
##             Specificity : 0.06777
##          Pos Pred Value : 0.78736
##          Neg Pred Value : 0.50000
##              Prevalence : 0.77874
##          Detection Rate : 0.76375
##    Detection Prevalence : 0.97001
##       Balanced Accuracy : 0.52426
##
##        'Positive' Class : 0
##
```

Sub 3.4 Random Forest

```
#5.D random forest
y_hat_rf <- predict(model_rf, test)
confusionMatrix(data = y_hat_rf, reference = test$default)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.220593e-01   3.719115e-01   8.078971e-01   8.355906e-01   7.787404e-01
## AccuracyPValue  McnemarPValue
##   2.649971e-09   5.382420e-46
```

Sub 3.5 Logistic Regression

```
#5.E glm
y_hat_glm <- predict(model_glm, test)
confusionMatrix(data = y_hat_glm, reference = test$default)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.207264e-01   3.604975e-01   8.065256e-01   8.342990e-01   7.787404e-01
## AccuracyPValue  McnemarPValue
##   7.921575e-09   4.861506e-50
```

Sub 3.6 Ensemble

```
#6.F Ensemble
y_hat_E_test<-E_Prediction(1,y_hat_rf, P_test)
confusionMatrix(data=y_hat_E_test, reference=test$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2272  472
##          1   65  192
##
##                Accuracy : 0.8211
##                  95% CI : (0.8069, 0.8346)
##     No Information Rate : 0.7787
##     P-Value [Acc > NIR] : 6.045e-09
##
##                   Kappa : 0.3348
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9722
##             Specificity : 0.2892
##          Pos Pred Value : 0.8280
##          Neg Pred Value : 0.7471
##              Prevalence : 0.7787
##          Detection Rate : 0.7571
##    Detection Prevalence : 0.9144
##       Balanced Accuracy : 0.6307
##
##        'Positive' Class : 0
##
```

Sub 3.7 Summary

Models	Accuracy on train data	Accuracy on test data
Proprietary	82.1%	81.7%
Naive Bayesian	80.3%	80.2%
KNN (k=60)	77.7%	78.1%
Random Forest (mtry=3)	81.7%	82.3%
Logistic	73%	82.1%
Ensemble	85.6%	82.1%

Conclusion and Future Works

As a conclusion, we have delivered the following:

- We have carefully examined the relationship among different data fields and relationship with default rate.
- We have identified several data abnormalities which should be clarified further.
- With these imperfect data, we have reviewed several default prediction system including a proprietary model. Nearly most of them can achieve accuracy and specificity better than simply assuming there is no default.

In order to take this research further, the most important is to clarify the abnormality and clean up the data if necessary. Next is to understand the clear legal definition of default in this data-set. With these, we can further enhance the model by the following:

- Running a more extensive search of optimal parameters for KNN and random forest
- Introducing more relevant parameters in the proprietary models
- Using PCA or other relevant techniques to reduce the dimension of the data

Finally, I want to take this opportunity to thank Professor Rafael Irizarry for organizing this great course. I have learnt tremendously from this. Equally important, I would like to thank all the staff instructors, who have provided tremendous help and insight in the discussion board and make this course even greater.