Credit Card Default Prediction

Loncón Joaquín

March 29, 2022

Introduction

The objective of this project is to build a **Machine Learning Model** that predicts if a **client will default on his card payment in the next month**. To achieve this goal some classification models will be trained and then grouped into one to make a final model (**Ensemble**).

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.

For more information about the data see Lichman, M. (2013). UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.

Prepare

Libraries required

```
if(!require(readr)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(plyr)) install.packages("plyr", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(scales)) install.packages("scales", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(mboost)) install.packages("mboost", repos = "http://cran.us.r-project.org")
if(!require(naivebayes)) install.packages("naivebayes", repos = "http://cran.us.r-project.org")
if(!require(MASS)) install.packages("MASS", repos = "http://cran.us.r-project.org")
if(!require(glmnet)) install.packages("glmnet", repos = "http://cran.us.r-project.org")
if(!require(Matrix)) install.packages("Matrix", repos = "http://cran.us.r-project.org")
if(!require(gbm)) install.packages("gbm", repos = "http://cran.us.r-project.org")

library(readr)
library(caret)
library(scales)
library(dplyr)
library(mboost)
library(naivebayes)
library(MASS)
library(glmnet)
library(Matrix)
library(plyr)
library(gbm)
```

Obtain the data

Load the data

```
Default <- read.csv(
    "../input/default-of-credit-card-clients-dataset/UCI_Credit_Card.csv")
```

Pre-Process

Variable Format Changes

1. Make the variables categorical.
2. Rename the factors in the SEX variable
   - 1 = Male
   - 2 = Female
3. Correct the factors in the EDUCATION variable (the 0,4,5,6 values are others/unknown)
   - 1 = Graduate_school
   - 2 = University
   - 3 = High_school
   - 0= Others (ex 0,4,5,6)
4. Rename the factors in the MARRIAGE variable
   - 1 = Married
   - 2 = Single
   - 3 = Divorced
   - 0 = Others

```
Default$SEX <- factor(Default$SEX,labels = c("Male","Female"))

Default$EDUCATION <- ifelse(Default$EDUCATION%in%c(0,4,5,6), 0, Default$EDUCATION)
Default$EDUCATION <- factor(Default$EDUCATION,
                            labels = c("Others","Graduate_school", "University", "High_school"))

Default$MARRIAGE <- factor(Default$MARRIAGE,
                           labels = c("Others","Married", "Single", "Divorced"))
```

5. Create other variables with the number of month delays
6. Correct the factors of PAY variables ,1 trough 8 means payment delay for x month, so lets call all of them delay
   - -2 = No consumption
   - -1 = Paid in full
   - 0 = The use of revolving credit
   - 1 = Payment delay (ex 1:8)

```r
Default$PAY0_DELAY <- ifelse(Default$PAY_0 %in% c(1:8), Default$PAY_0, 0)
Default$PAY2_DELAY <- ifelse(Default$PAY_2 %in% c(1:8), Default$PAY_2, 0)
Default$PAY3_DELAY <- ifelse(Default$PAY_3 %in% c(1:8), Default$PAY_3, 0)
Default$PAY4_DELAY <- ifelse(Default$PAY_4 %in% c(1:8), Default$PAY_4, 0)
Default$PAY5_DELAY <- ifelse(Default$PAY_5 %in% c(1:8), Default$PAY_5, 0)
Default$PAY6_DELAY <- ifelse(Default$PAY_6 %in% c(1:8), Default$PAY_6, 0)

Default$PAY_0 <-ifelse(Default$PAY_0>0, 1, Default$PAY_0)
Default$PAY_2 <-ifelse(Default$PAY_2>0, 1, Default$PAY_2)
Default$PAY_3 <-ifelse(Default$PAY_3>0, 1, Default$PAY_3)
Default$PAY_4 <-ifelse(Default$PAY_4>0, 1, Default$PAY_4)
Default$PAY_5 <-ifelse(Default$PAY_5>0, 1, Default$PAY_5)
Default$PAY_6 <-ifelse(Default$PAY_6>0, 1, Default$PAY_6)

pay_labels <- c("No_consumption","Full_paid", "Use_of_revolving_credit", "Payment_delay")

Default$PAY_0 <- factor(pay_labels[Default$PAY_0+3])
Default$PAY_2 <- factor(pay_labels[Default$PAY_2+3])
Default$PAY_3 <- factor(pay_labels[Default$PAY_3+3])
Default$PAY_4 <- factor(pay_labels[Default$PAY_4+3])
Default$PAY_5 <- factor(pay_labels[Default$PAY_5+3])
Default$PAY_6 <- factor(pay_labels[Default$PAY_6+3])
```

7. Make the variable of interest (default.payment.next.month) a factor and re-level the variable for later interpretation in the models, this makes that the "Default Next Month" level can be interpreted as $Y = 1$.

```r
Default$default.payment.next.month <- factor(Default$default.payment.next.month,
                                             labels = c("Normal", "Default Next Month"))
Default$default.payment.next.month <- relevel(Default$default.payment.next.month,
                                              "Default Next Month")
```

Missing values

No value is missing in the dataframe, we can see this with the following simple code.

```r
any(is.na(Default))
```

```
## [1] FALSE
```

No observations are duplicated

```r
any(duplicated(Default))
```

```
## [1] FALSE
```

Data partition for validation

Create a partition of the data to later test the final model.

```r
set.seed(1, sample.kind = "Rounding") # just for make the code reproducible

validation_index <- createDataPartition(y = Default$default.payment.next.month, times = 1, p = 0.2, lis

validation <- Default[validation_index,]
df <- Default[-validation_index,]
```

Train & Test sets

To train the models, test, and optimize, generate an index, with 80% of the data for training the model and 20% for testing (without using the validation set, which is going to be used at the end).

```r
train_index <- createDataPartition(y = df$default.payment.next.month, times = 1, p = 0.8, list = FALSE)

train_set <- df[train_index,]
test_set <- df[-train_index,]
```

Analysis

Structure & simple statistics

Let's start the analysis by looking at some summary statistics.

```r
str(df)
```

```
## tibble [23,999 x 31] (S3: tbl_df/tbl/data.frame)
##  $ ID                         : num [1:23999] 1 2 3 4 5 7 8 9 10 11 ...
##  $ LIMIT_BAL                  : num [1:23999] 20000 120000 90000 50000 50000 500000 100000 140000 2000
##  $ SEX                        : Factor w/ 2 levels "Male","Female": 2 2 2 2 1 1 2 2 1 2 ...
##  $ EDUCATION                  : Factor w/ 4 levels "Others","Graduate_school",..: 3 3 3 3 3 2 3 4 4 4
##  $ MARRIAGE                   : Factor w/ 4 levels "Others","Married",..: 2 3 3 2 2 3 3 2 3 3 ...
##  $ AGE                        : num [1:23999] 24 26 34 37 57 29 23 28 35 34 ...
##  $ PAY_0                      : Factor w/ 4 levels "Full_paid","No_consumption",..: 3 1 4 4 1 4 4 4 2
##  $ PAY_2                      : Factor w/ 4 levels "Full_paid","No_consumption",..: 3 3 4 4 4 4 1 4 2
##  $ PAY_3                      : Factor w/ 4 levels "Full_paid","No_consumption",..: 1 4 4 4 1 4 1 3 2
##  $ PAY_4                      : Factor w/ 4 levels "Full_paid","No_consumption",..: 1 4 4 4 4 4 4 4 2
##  $ PAY_5                      : Factor w/ 4 levels "Full_paid","No_consumption",..: 2 4 4 4 4 4 4 4 1
##  $ PAY_6                      : Factor w/ 4 levels "Full_paid","No_consumption",..: 2 3 4 4 4 4 1 4 1
##  $ BILL_AMT1                  : num [1:23999] 3913 2682 29239 46990 8617 ...
##  $ BILL_AMT2                  : num [1:23999] 3102 1725 14027 48233 5670 ...
##  $ BILL_AMT3                  : num [1:23999] 689 2682 13559 49291 35835 ...
##  $ BILL_AMT4                  : num [1:23999] 0 3272 14331 28314 20940 ...
##  $ BILL_AMT5                  : num [1:23999] 0 3455 14948 28959 19146 ...
##  $ BILL_AMT6                  : num [1:23999] 0 3261 15549 29547 19131 ...
##  $ PAY_AMT1                   : num [1:23999] 0 0 1518 2000 2000 ...
##  $ PAY_AMT2                   : num [1:23999] 689 1000 1500 2019 36681 ...
##  $ PAY_AMT3                   : num [1:23999] 0 1000 1000 1200 10000 38000 0 432 0 50 ...
##  $ PAY_AMT4                   : num [1:23999] 0 1000 1000 1100 9000 ...
##  $ PAY_AMT5                   : num [1:23999] 0 0 1000 1069 689 ...
##  $ PAY_AMT6                   : num [1:23999] 0 2000 5000 1000 679 ...
```

```
##  $ default.payment.next.month: Factor w/ 2 levels "Default Next Month",..: 1 1 2 2 2 2 2 2 2 2 ...
##  $ PAY0_DELAY                : num [1:23999] 2 0 0 0 0 0 0 0 0 0 ...
##  $ PAY2_DELAY                : num [1:23999] 2 2 0 0 0 0 0 0 0 0 ...
##  $ PAY3_DELAY                : num [1:23999] 0 0 0 0 0 0 0 2 0 2 ...
##  $ PAY4_DELAY                : num [1:23999] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PAY5_DELAY                : num [1:23999] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PAY6_DELAY                : num [1:23999] 0 2 0 0 0 0 0 0 0 0 ...
```

```
summary(df)
```

```
##        ID          LIMIT_BAL          SEX                  EDUCATION
##  Min.   :    1   Min.   :  10000   Male  : 9554   Others         :  374
##  1st Qu.: 7476   1st Qu.:  50000   Female:14445   Graduate_school: 8443
##  Median :15001   Median : 140000                  University     :11245
##  Mean   :14982   Mean   : 167139                  High_school    : 3937
##  3rd Qu.:22422   3rd Qu.: 240000
##  Max.   :30000   Max.   :1000000
##      MARRIAGE          AGE                             PAY_0
##  Others  :   43   Min.   :21.00   Full_paid              : 4547
##  Married :10923   1st Qu.:28.00   No_consumption         : 2205
##  Single  :12783   Median :34.00   Payment_delay          : 5478
##  Divorced:  250   Mean   :35.48   Use_of_revolving_credit:11769
##                   3rd Qu.:41.00
##                   Max.   :79.00
##                      PAY_2                            PAY_3
##  Full_paid              : 4854   Full_paid              : 4746
##  No_consumption         : 3034   No_consumption         : 3284
##  Payment_delay          : 3532   Payment_delay          : 3357
##  Use_of_revolving_credit:12579   Use_of_revolving_credit:12612
##
##
##                      PAY_4                            PAY_5
##  Full_paid              : 4545   Full_paid              : 4412
##  No_consumption         : 3484   No_consumption         : 3643
##  Payment_delay          : 2798   Payment_delay          : 2345
##  Use_of_revolving_credit:13172   Use_of_revolving_credit:13599
##
##
##                      PAY_6         BILL_AMT1         BILL_AMT2
##  Full_paid              : 4565   Min.   :-165580   Min.   :-67526
##  No_consumption         : 3948   1st Qu.:   3518   1st Qu.:  2968
##  Payment_delay          : 2458   Median :  22413   Median : 21292
##  Use_of_revolving_credit:13028   Mean   :  50924   Mean   : 48931
##                                  3rd Qu.:  66387   3rd Qu.: 63190
##                                  Max.   : 964511   Max.   :983931
##    BILL_AMT3         BILL_AMT4         BILL_AMT5        BILL_AMT6
##  Min.   :-157264   Min.   :-170000   Min.   :-81334   Min.   :-209051
##  1st Qu.:   2669   1st Qu.:   2355   1st Qu.:  1776   1st Qu.:   1244
##  Median :  20154   Median :  19067   Median : 18100   Median :  17113
##  Mean   :  46741   Mean   :  43107   Mean   : 40114   Mean   :  38692
##  3rd Qu.:  59801   3rd Qu.:  53897   3rd Qu.: 49945   3rd Qu.:  48969
##  Max.   : 855086   Max.   : 891586   Max.   :927171   Max.   : 961664
##     PAY_AMT1          PAY_AMT2          PAY_AMT3          PAY_AMT4
##  Min.   :     0   Min.   :      0   Min.   :     0   Min.   :     0
```

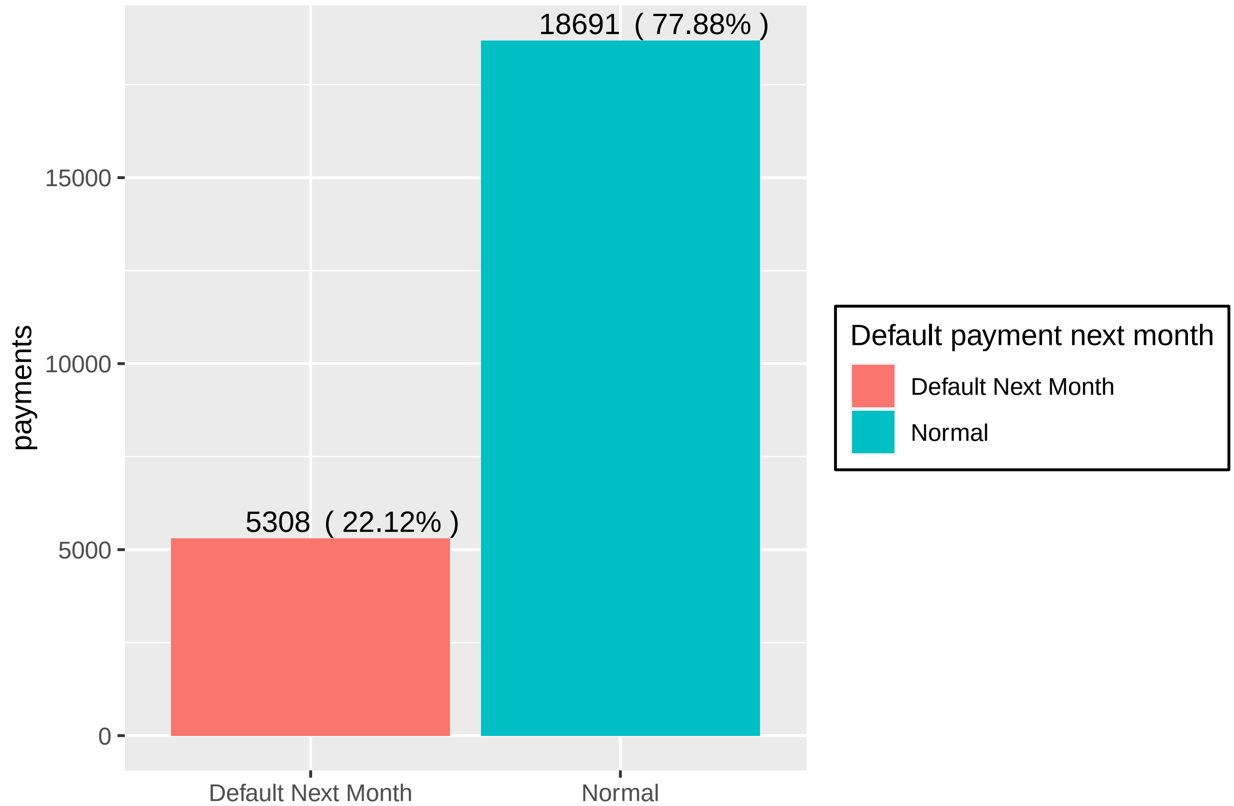
```
## 1st Qu.:  1000   1st Qu.:    839   1st Qu.:   390   1st Qu.:   291
## Median :  2100   Median :   2006   Median :  1800   Median :  1500
## Mean   :  5626   Mean   :   5848   Mean   :  5213   Mean   :  4839
## 3rd Qu.:  5006   3rd Qu.:   5000   3rd Qu.:  4500   3rd Qu.:  4006
## Max.   :873552   Max.   :1215471   Max.   :896040   Max.   :621000
##    PAY_AMT5          PAY_AMT6          default.payment.next.month
## Min.   :     0.0   Min.   :     0   Default Next Month: 5308
## 1st Qu.:   247.5   1st Qu.:   100   Normal            :18691
## Median :  1503.0   Median :  1500
## Mean   :  4768.3   Mean   :  5210
## 3rd Qu.:  4035.0   3rd Qu.:  4000
## Max.   :417990.0   Max.   :528666
##   PAY0_DELAY       PAY2_DELAY       PAY3_DELAY       PAY4_DELAY
## Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
## 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
## Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
## Mean   :0.3579   Mean   :0.3178   Mean   :0.3025   Mean   :0.2572
## 3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
## Max.   :8.0000   Max.   :8.0000   Max.   :8.0000   Max.   :8.0000
##   PAY5_DELAY       PAY6_DELAY
## Min.   :0.0000   Min.   :0.0000
## 1st Qu.:0.0000   1st Qu.:0.0000
## Median :0.0000   Median :0.0000
## Mean   :0.2182   Mean   :0.2258
## 3rd Qu.:0.0000   3rd Qu.:0.0000
## Max.   :8.0000   Max.   :8.0000
```

Visual Analysis

Default Distribution

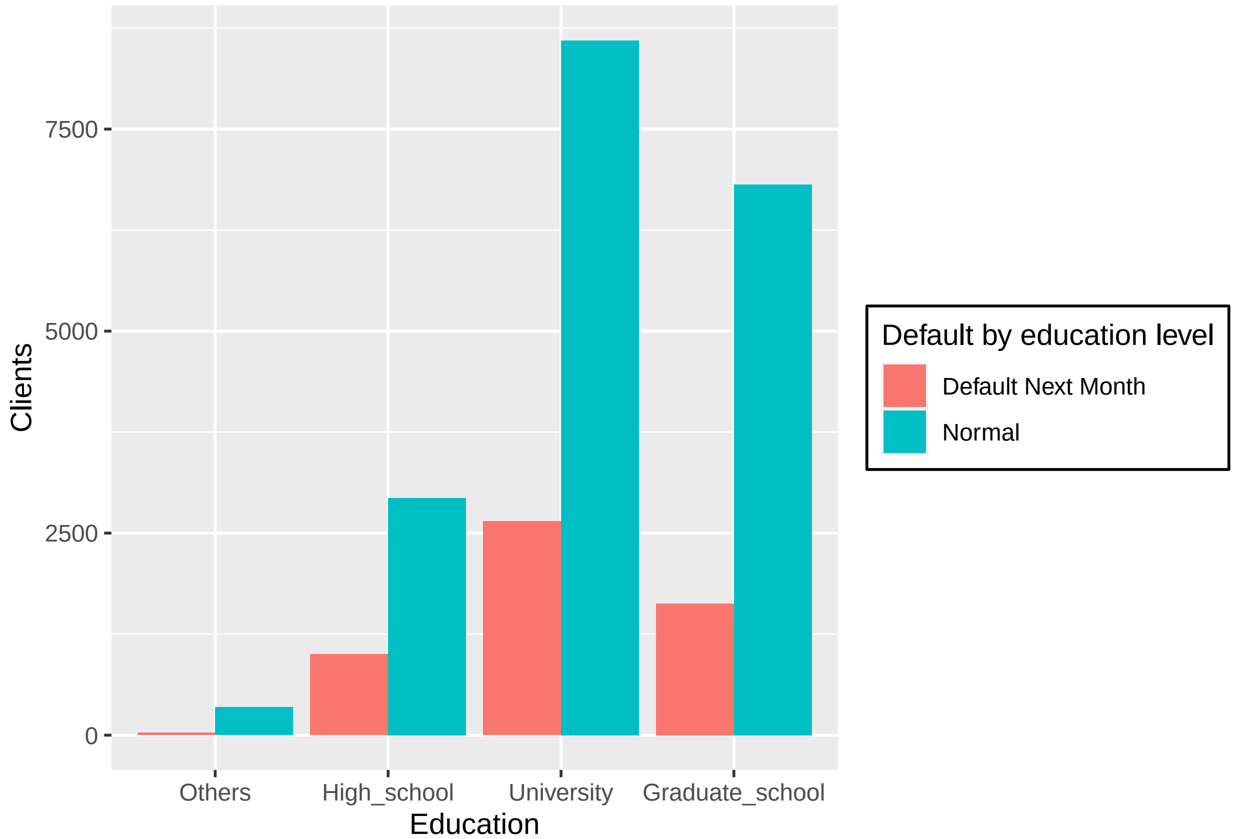
This graph shows the high prevalence in the data, only 22.12% of the clients have a default payment next month (at the end of the series) and the other 77.88% do not.

```
df%>%
    group_by(default.payment.next.month)%>%
    summarise(n = n())%>%
    mutate(percentage = n/sum(n)*100)%>%
    ggplot(aes(default.payment.next.month, n, fill=default.payment.next.month))+
    geom_bar(stat = "identity")+
    geom_text(aes(label=n), vjust=-0.3, hjust = 1, size=4)+
    geom_text(aes(label=paste0("( ",round(percentage,2),"% )")),
              vjust=-0.3, hjust = -0.1, size=4)+
    ylab("payments")+
    xlab("")+
    scale_fill_manual(name = "Default payment next month",
                      values = c("Default Next Month" = "#F8766D", "Normal" = "#00BFC4"))+
    theme(legend.background = element_rect(fill="white",
                                           size=0.5,
                                           linetype="solid",
                                           colour ="black"))
```



Default by Education level

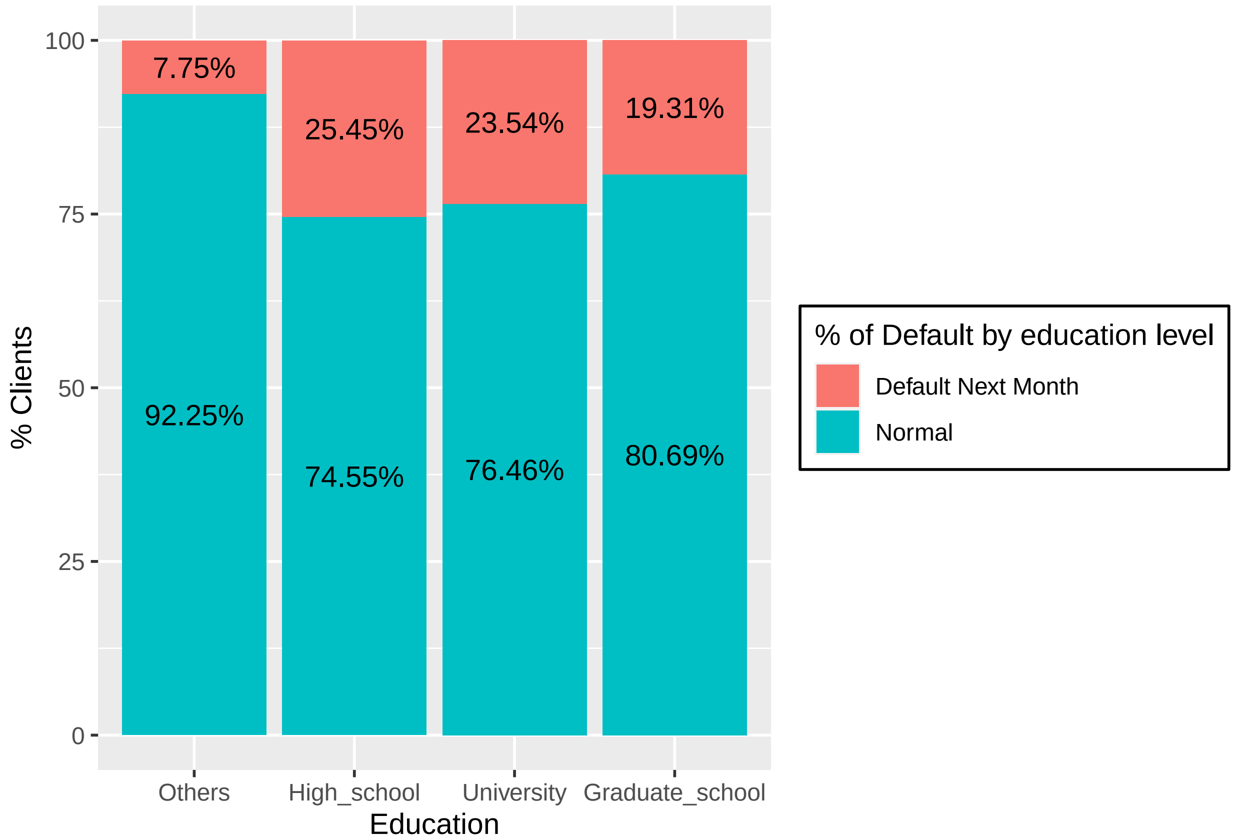
```
df%>%
    group_by(EDUCATION,default.payment.next.month)%>%
    summarise(n = n(), .groups = 'drop')%>%
ggplot(aes(fill=default.payment.next.month, y=n,
x=factor(EDUCATION, levels = c("Others", "High_school", "University", "Graduate_school")))) +
    geom_bar(position="dodge", stat="identity")+
    scale_fill_manual(name = "Default by education level",
                      values = c("Default Next Month" = "#F8766D", "Normal" = "#00BFC4"))+
    xlab("Education")+
    ylab("Clients")+
    theme(legend.background = element_rect(fill="white",
                                           size=0.5,
                                           linetype="solid",
                                           colour ="black"))
```



there appears to be difference in the defaults by education level, but since there are different quantities in the groups we can't conclude this, so let's see the following graph to be able to compare between categories.

Percentage of Default by Education level

```
df %>%
    count(EDUCATION, default.payment.next.month) %>%
    group_by(EDUCATION) %>%
    mutate(n = n/sum(n) * 100) %>%
    ggplot() +
    aes(factor(EDUCATION,
               levels = c("Others", "High_school", "University", "Graduate_school")), n,
        fill = default.payment.next.month, label = paste0(round(n, 2), "%")) +
    geom_col() +
    geom_text(position=position_stack(0.5))+
    scale_fill_manual(name = "% of Default by education level",
                      values = c("Default Next Month" = "#F8766D", "Normal" = "#00BFC4"))+
    xlab("Education")+
    ylab("% Clients")+
    theme(legend.background = element_rect(fill="white",
                                           size=0.5,
                                           linetype="solid",
                                           colour ="black"))
```

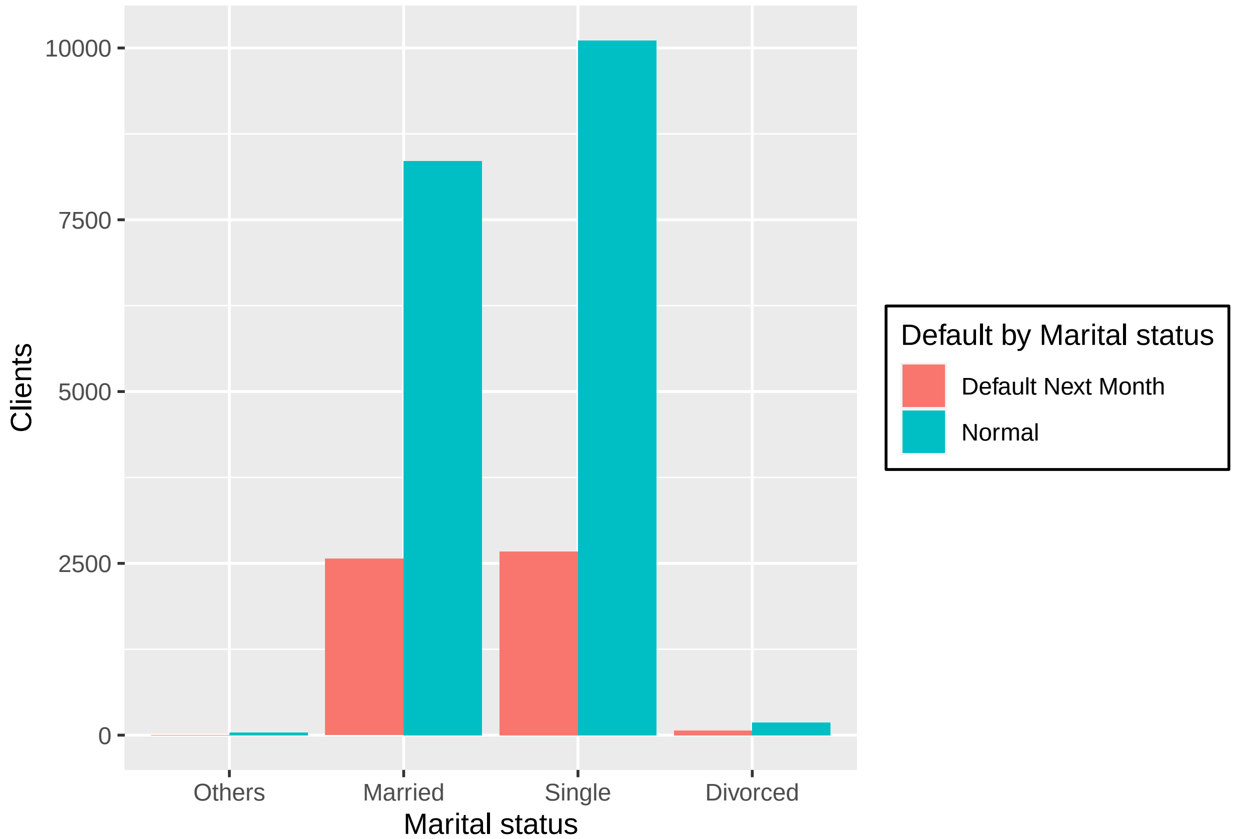


And now we certainly see that there is a little difference among the education levels.

Default by marital status

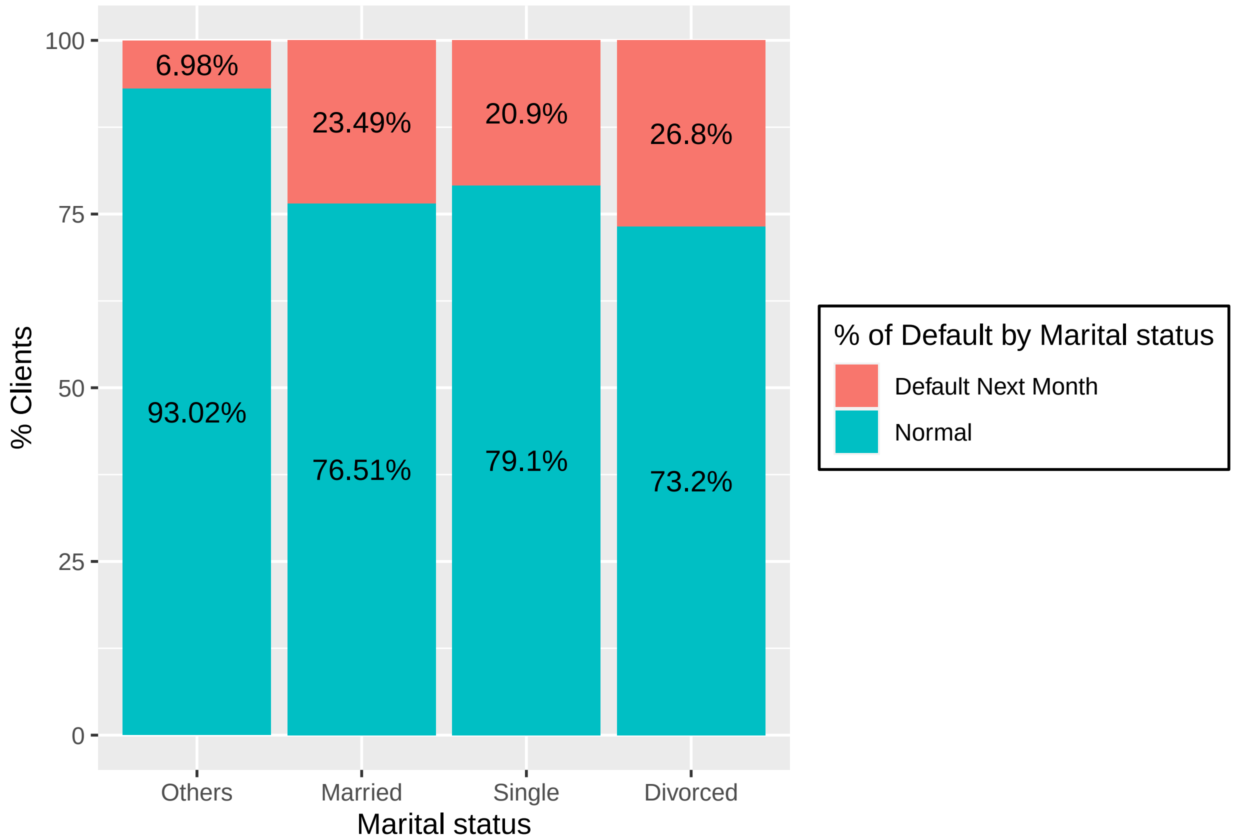
Let's see the same relations but by marital status

```
df%>%
    group_by(MARRIAGE,default.payment.next.month)%>%
    summarise(n = n(), .groups = 'drop')%>%
    ggplot(aes(fill=default.payment.next.month, y=n,
x=factor(MARRIAGE, levels = c("Others","Married", "Single", "Divorced")))) +
    geom_bar(position="dodge", stat="identity")+
    scale_fill_manual(name = "Default by Marital status",
                      values = c("Default Next Month" = "#F8766D", "Normal" = "#00BFC4"))+
    xlab("Marital status")+
    ylab("Clients")+
    theme(legend.background = element_rect(fill="white",
                                           size=0.5,
                                           linetype="solid",
                                           colour ="black"))
```



Percentage of Default by Marital status

```
df %>%
    count(MARRIAGE, default.payment.next.month) %>%
    group_by(MARRIAGE) %>%
    mutate(n = n/sum(n) * 100) %>%
    ggplot() + aes(factor(MARRIAGE,
                          levels = c("Others","Married", "Single", "Divorced")), n,
                   fill = default.payment.next.month, label = paste0(round(n, 2), "%")) +
    geom_col() +
    geom_text(position=position_stack(0.5))+
    scale_fill_manual(name = "% of Default by Marital status",
                      values = c("Default Next Month" = "#F8766D", "Normal" = "#00BFC4"))+
    xlab("Marital status")+
    ylab("% Clients")+
    theme(legend.background = element_rect(fill="white",
                                           size=0.5,
                                           linetype="solid",
                                           colour ="black"))
```

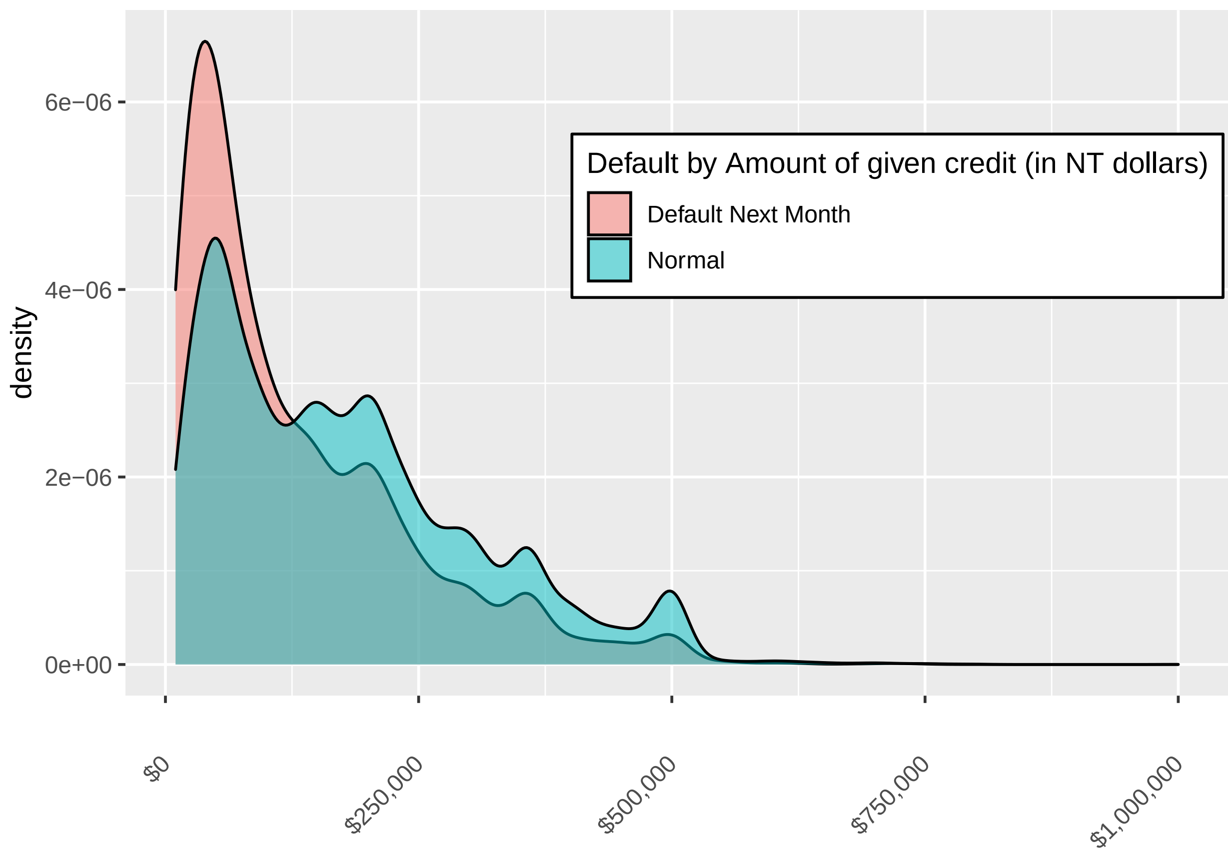


Default by Amount of given credit (in NT dollars)

Here we can see that the most of the client that default at the end of the period do not have a big Amount of credit compared to others, in fact they even tend to have less.

Note: NT dollar is the New Taiwan dollar, the official currency of Taiwan.

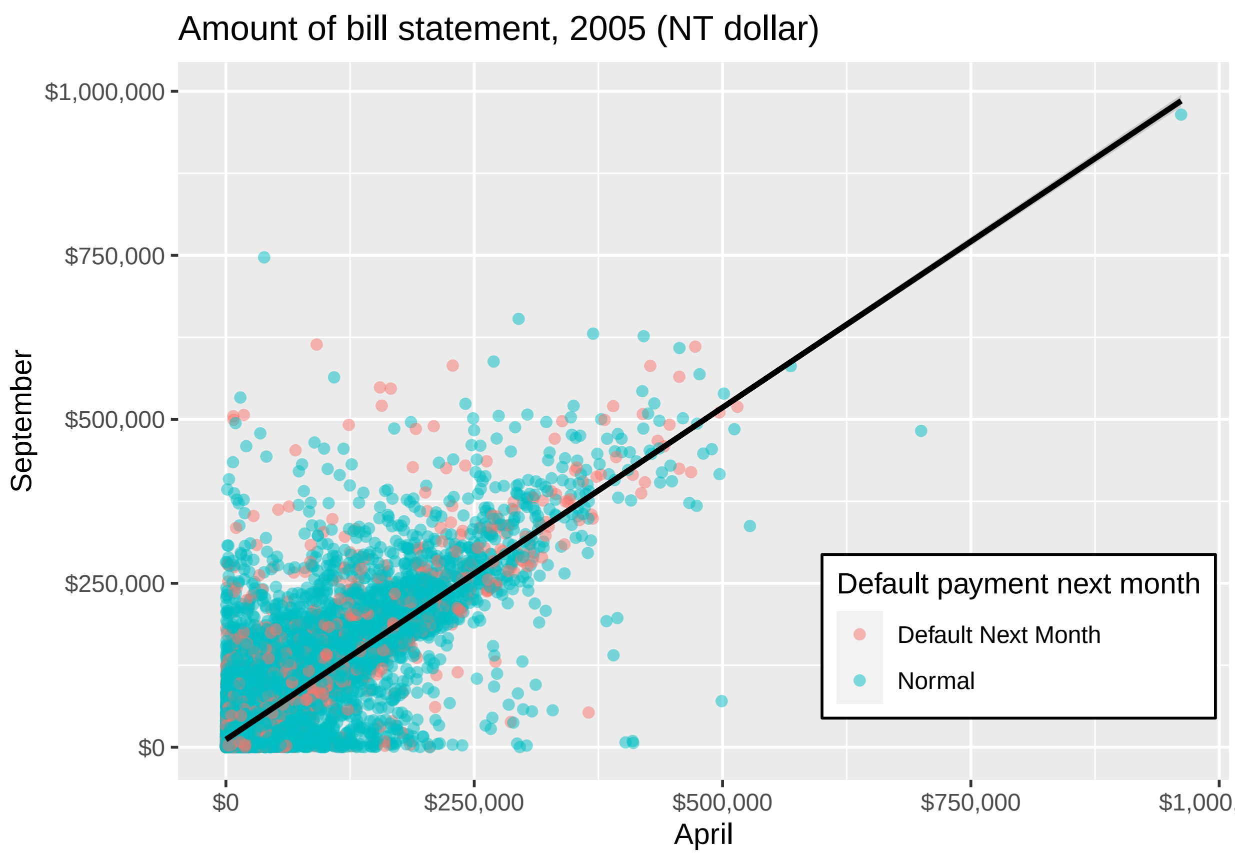
```
df%>%
    group_by(default.payment.next.month)%>%
    ggplot()+
    geom_density(aes(LIMIT_BAL, fill = default.payment.next.month),
                 alpha = .5)+
    theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=1))+
    xlab("")+
    scale_x_continuous(labels = label_dollar())+
    scale_fill_manual(name = "Default by Amount of given credit (in NT dollars)",
                      values = c("Default Next Month" = "#F8766D", "Normal" = "#00BFC4"))+
    theme(legend.position = c(0.7, .7),
          legend.background = element_rect(fill="white",
                                           size=0.5, linetype="solid",
                                           colour ="black"))
```



Amount of bill statement across time

Here we can see that customers tend to maintain the amount of their bill in different periods, this is, customers with high bills tend to maintain high bills over time and vice versa.

```
df%>%
    filter(BILL_AMT6>0,BILL_AMT1>0)%>%
    ggplot(aes(BILL_AMT6,BILL_AMT1, col = default.payment.next.month))+
    geom_point(alpha = .5)+
    scale_color_manual(name = "Default payment next month",
                       values = c("Default Next Month" = "#F8766D", "Normal" = "#00BFC4"))+
    geom_smooth(method = "lm", col = "black")+
    scale_x_continuous(labels = label_dollar())+
    scale_y_continuous(labels = label_dollar())+
    ggtitle("Amount of bill statement, 2005 (NT dollar)")+
    xlab("April")+
    ylab("September")+
    theme(legend.position = c(0.8, .2),
          legend.background = element_rect(fill="white",
                                           size=0.5,
                                           linetype="solid",
                                           colour ="black"))
```



Model Building

Prevalence Problem

As shown above, there is a high prevalence of one class in the Default variable (default.payment.next.month). This brings big problems when creating a model, as seen below a classification tree is trained and its performance on the test set is displayed.

```
fit_bad_model <- train(default.payment.next.month~.,
                       data = train_set,
                       method = "rpart",
                       trControl = trainControl(method = "cv", number = 10))

hat_bad_model <- predict(fit_bad_model, newdata = test_set)

confusionMatrix(data = hat_bad_model, reference = test_set$default.payment.next.month)
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction              Default Next Month Normal
##   Default Next Month                   365    143
##   Normal                               696   3595
```

```
##
##                Accuracy : 0.8252
##                  95% CI : (0.8141, 0.8358)
##     No Information Rate : 0.7789
##     P-Value [Acc > NIR] : 1.227e-15
##
##                   Kappa : 0.3759
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.34402
##             Specificity : 0.96174
##          Pos Pred Value : 0.71850
##          Neg Pred Value : 0.83780
##              Prevalence : 0.22109
##          Detection Rate : 0.07606
##    Detection Prevalence : 0.10586
##       Balanced Accuracy : 0.65288
##
##        'Positive' Class : Default Next Month
##
```

Up-Sampling

To deal with the problem of prevalence we can do a technique called up-sampling. this method increases the size of the minority class by sampling with replacement so that the classes in the data will have the same size. Note that by doing this, the observations will balance out, causing there to be no more prevalence.

```
trainup <- upSample(y=train_set$default.payment.next.month,
                    x=train_set[,-which(names(train_set)=="default.payment.next.month")],
                    yname = "default.payment.next.month")

table(trainup$default.payment.next.month)
```

```
##
## Default Next Month             Normal
##              14953              14953
```

Machine Learning Models

Now with the balanced data we are going to train a set of different models for classification and observe their performances on the test set. Predictors will be centered and scaled by the preProcess option. In addition, we are going to use a 10-fold cross-validation, this is randomly split the observations of our train set into 10 non-overlapping sets, and do the calculation of the Mean Squared Error (MSE) for each of these sets having into account the different parameters of the model that lead us to that MSE, then the model selects the optimal parameters, the ones that minimize the MSE.

Classification Tree

```
fit_rpart <- train(default.payment.next.month ~ .,
                   data = trainup,
                   method = "rpart",
                   trControl = trainControl(method = "cv", number = 10),
                   preProcess = c("nzv"))

hat_rpart <- predict(fit_rpart, newdata = test_set)

confusionMatrix(data = hat_rpart, reference = test_set$default.payment.next.month)
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction             Default Next Month Normal
##   Default Next Month                  601    644
##   Normal                              460   3094
##
##                Accuracy : 0.77
##                  95% CI : (0.7578, 0.7818)
##     No Information Rate : 0.7789
##     P-Value [Acc > NIR] : 0.9344
##
##                   Kappa : 0.3711
##
##  Mcnemar's Test P-Value : 3.636e-08
##
##             Sensitivity : 0.5664
##             Specificity : 0.8277
##          Pos Pred Value : 0.4827
##          Neg Pred Value : 0.8706
##              Prevalence : 0.2211
##          Detection Rate : 0.1252
##    Detection Prevalence : 0.2594
##       Balanced Accuracy : 0.6971
##
##        'Positive' Class : Default Next Month
##
```

Generalized Linear Model

```
fit_glm <- train(default.payment.next.month ~ .,
                 data = trainup,
                 method = "glm",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale", "nzv"))

hat_glm <- predict(fit_glm, newdata = test_set)

confusionMatrix(data = hat_glm, reference = test_set$default.payment.next.month)
```

```
## Confusion Matrix and Statistics
```

```
##
##                        Reference
## Prediction              Default Next Month Normal
##   Default Next Month                   649    670
##   Normal                               412   3068
##
##                Accuracy : 0.7745
##                  95% CI : (0.7624, 0.7863)
##     No Information Rate : 0.7789
##     P-Value [Acc > NIR] : 0.7732
##
##                   Kappa : 0.3978
##
##  Mcnemar's Test P-Value : 5.583e-15
##
##             Sensitivity : 0.6117
##             Specificity : 0.8208
##          Pos Pred Value : 0.4920
##          Neg Pred Value : 0.8816
##              Prevalence : 0.2211
##          Detection Rate : 0.1352
##    Detection Prevalence : 0.2748
##       Balanced Accuracy : 0.7162
##
##        'Positive' Class : Default Next Month
##
```

Stochastic Gradient Boosting

```r
fit_gbm <- train(default.payment.next.month ~ .,
                 data = trainup,
                 method = "gbm",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale", "nzv"),
                 verbose = FALSE)

hat_gbm <- predict(fit_gbm, newdata = test_set)

confusionMatrix(data = hat_gbm, reference = test_set$default.payment.next.month)
```

```
## Confusion Matrix and Statistics
##
##                        Reference
## Prediction              Default Next Month Normal
##   Default Next Month                   682    765
##   Normal                               379   2973
##
##                Accuracy : 0.7616
##                  95% CI : (0.7493, 0.7736)
##     No Information Rate : 0.7789
##     P-Value [Acc > NIR] : 0.998
##
```

```
##                  Kappa : 0.3876
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6428
##             Specificity : 0.7953
##          Pos Pred Value : 0.4713
##          Neg Pred Value : 0.8869
##              Prevalence : 0.2211
##          Detection Rate : 0.1421
##    Detection Prevalence : 0.3015
##       Balanced Accuracy : 0.7191
##
##        'Positive' Class : Default Next Month
##
```

Naive Bayes

```
fit_nb <- train(default.payment.next.month ~ .,
                data = trainup,
                method = "naive_bayes",
                trControl = trainControl(method = "cv", number = 10),
                preProcess = c("center", "scale", "nzv"))

hat_nb <- predict(fit_nb, newdata = test_set)

confusionMatrix(data = hat_nb, reference = test_set$default.payment.next.month)
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction             Default Next Month Normal
##   Default Next Month                  607    679
##   Normal                              454   3059
##
##                Accuracy : 0.7639
##                  95% CI : (0.7516, 0.7759)
##     No Information Rate : 0.7789
##     P-Value [Acc > NIR] : 0.9939
##
##                   Kappa : 0.3629
##
##  Mcnemar's Test P-Value : 2.837e-11
##
##             Sensitivity : 0.5721
##             Specificity : 0.8184
##          Pos Pred Value : 0.4720
##          Neg Pred Value : 0.8708
##              Prevalence : 0.2211
##          Detection Rate : 0.1265
##    Detection Prevalence : 0.2680
##       Balanced Accuracy : 0.6952
```

```
##
##        'Positive' Class : Default Next Month
##
```

Generalized linear model by likelihood based boosting

```
fit_glmboost <- train(default.payment.next.month ~ .,
                      data = trainup,
                      method = "glmboost",
                      trControl = trainControl(method = "cv", number = 10),
                      preProcess = c("center", "scale", "nzv"))

hat_glmboost <- predict(fit_glmboost, newdata = test_set)

confusionMatrix(data = hat_glmboost, reference = test_set$default.payment.next.month)
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction            Default Next Month Normal
##   Default Next Month                612    627
##   Normal                            449   3111
##
##                Accuracy : 0.7758
##                  95% CI : (0.7637, 0.7875)
##     No Information Rate : 0.7789
##     P-Value [Acc > NIR] : 0.7059
##
##                   Kappa : 0.3859
##
##  Mcnemar's Test P-Value : 6.817e-08
##
##             Sensitivity : 0.5768
##             Specificity : 0.8323
##          Pos Pred Value : 0.4939
##          Neg Pred Value : 0.8739
##              Prevalence : 0.2211
##          Detection Rate : 0.1275
##    Detection Prevalence : 0.2582
##       Balanced Accuracy : 0.7045
##
##        'Positive' Class : Default Next Month
##
```

Linear Discriminant Analysis

```
fit_lda <- train(default.payment.next.month ~ .,
                 data = trainup,
                 method = "lda",
                 trControl = trainControl(method = "cv", number = 10),
```

```r
                 preProcess = c("center", "scale", "nzv"))

hat_lda <- predict(fit_lda, newdata = test_set)

confusionMatrix(data = hat_lda, reference = test_set$default.payment.next.month)
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           Default Next Month Normal
##   Default Next Month                637    663
##   Normal                            424   3075
##
##                Accuracy : 0.7735
##                  95% CI : (0.7614, 0.7853)
##     No Information Rate : 0.7789
##     P-Value [Acc > NIR] : 0.8218
##
##                   Kappa : 0.3914
##
##  Mcnemar's Test P-Value : 5.247e-13
##
##             Sensitivity : 0.6004
##             Specificity : 0.8226
##          Pos Pred Value : 0.4900
##          Neg Pred Value : 0.8788
##              Prevalence : 0.2211
##          Detection Rate : 0.1327
##    Detection Prevalence : 0.2709
##       Balanced Accuracy : 0.7115
##
##        'Positive' Class : Default Next Month
##
```

Lasso and Elastic-Net Regularized Generalized Linear Models

```r
fit_glmnet <- train(default.payment.next.month ~ .,
                    data = trainup,
                    method = "glmnet",
                    trControl = trainControl(method = "cv", number = 10),
                    preProcess = c("center", "scale", "nzv"))

hat_glmnet <- predict(fit_glmnet, newdata = test_set)

confusionMatrix(data = hat_glmnet, reference = test_set$default.payment.next.month)
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           Default Next Month Normal
##   Default Next Month                650    669
```

```
##   Normal                            411   3069
##
##                Accuracy : 0.775
##                  95% CI : (0.7629, 0.7867)
##     No Information Rate : 0.7789
##     P-Value [Acc > NIR] : 0.7518
##
##                   Kappa : 0.3989
##
##  Mcnemar's Test P-Value : 5.271e-15
##
##             Sensitivity : 0.6126
##             Specificity : 0.8210
##          Pos Pred Value : 0.4928
##          Neg Pred Value : 0.8819
##              Prevalence : 0.2211
##          Detection Rate : 0.1354
##    Detection Prevalence : 0.2748
##       Balanced Accuracy : 0.7168
##
##        'Positive' Class : Default Next Month
##
```

Ensemble

An ensemble is the concept of combining the results of different algorithms, i.e. group different machine learning algorithms into one, this often greatly improve the final results. To achieve this goal we are going to take the vector of conditional probabilities given to each class by every particular algorithm, and then sum all the conditional probabilities given by the algorithms and take an average, so the result would be an average conditional probability for each observation.

Conditional Probabilities

First save the conditional probabilities with the option `type = "prob"` in the `predict()` function and then create the ensemble.

```
p_rpart <- predict(fit_rpart, newdata = test_set, type = "prob")
p_glm <- predict(fit_glm, newdata = test_set, type = "prob")
p_gbm <- predict(fit_gbm, newdata = test_set, type = "prob")
p_nb <- predict(fit_nb, newdata = test_set, type = "prob")
p_glmboost <- predict(fit_glmboost, newdata = test_set, type = "prob")
p_lda <- predict(fit_lda, newdata = test_set, type = "prob")
p_glmnet <- predict(fit_glmnet, newdata = test_set, type = "prob")

p <- (p_rpart+
      p_glm+
      p_gbm+
      p_nb+
      p_glmboost+
      p_lda+
      p_glmnet)/7
```

```
head(p)
```

```
##   Default Next Month    Normal
## 1          0.5762548 0.4237452
## 2          0.3812848 0.6187152
## 3          0.2705422 0.7294578
## 4          0.2002498 0.7997502
## 5          0.2620616 0.7379384
## 6          0.3077199 0.6922801
```

Ensemble Results

Now test the performance of the ensemble.

```
y_pred <- factor(apply(p, 1, which.max))

y_pred <- factor(y_pred, labels = c("Default Next Month", "Normal"))

confusionMatrix(y_pred, test_set$default.payment.next.month)
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           Default Next Month Normal
##   Default Next Month                645    691
##   Normal                            416   3047
##
##                Accuracy : 0.7693
##                  95% CI : (0.7571, 0.7812)
##     No Information Rate : 0.7789
##     P-Value [Acc > NIR] : 0.9466
##
##                   Kappa : 0.3871
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6079
##             Specificity : 0.8151
##          Pos Pred Value : 0.4828
##          Neg Pred Value : 0.8799
##              Prevalence : 0.2211
##          Detection Rate : 0.1344
##    Detection Prevalence : 0.2784
##       Balanced Accuracy : 0.7115
##
##        'Positive' Class : Default Next Month
##
```

Final Model

Now that we have an idea of how te models work on the test set and see that there is not a big difference in the models that justify drop one or another we are going to train the models in all the df set and create the ensemble, our **Final Model**, then we use the validation set **JUST TO SEE** the final performance.

Up-Sampling

We use the up-sampling technique again but in the `df`.

```
df_trainup <- upSample(y=df$default.payment.next.month,
                       x=df[,-which(names(df)=="default.payment.next.month")],
                       yname = "default.payment.next.month")

table(df_trainup$default.payment.next.month)
```

```
##
## Default Next Month             Normal
##              18691              18691
```

Machine Learning Models

Now train the models with all the `df_trainup` set.

Classification Tree

```
fit_rpart_final <- train(default.payment.next.month ~ .,
                   data = df_trainup,
                   method = "rpart",
                   trControl = trainControl(method = "cv", number = 10),
                   preProcess = c("nzv"))

p_rpart_final <- predict(fit_rpart_final, newdata = validation, type = "prob")
```

Generalized Linear Model

```
fit_glm_final <- train(default.payment.next.month ~ .,
                 data = df_trainup,
                 method = "glm",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale", "nzv"))

p_glm_final <- predict(fit_glm_final, newdata = validation, type = "prob")
```

Stochastic Gradient Boosting

```
fit_gbm_final <- train(default.payment.next.month ~ .,
                 data = df_trainup,
                 method = "gbm",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale", "nzv"),
                 verbose = FALSE)

p_gbm_final <- predict(fit_gbm_final, newdata = validation, type = "prob")
```

Naive Bayes

```
fit_nb_final <- train(default.payment.next.month ~ .,
                data = df_trainup,
                method = "naive_bayes",
                trControl = trainControl(method = "cv", number = 10),
                preProcess = c("center", "scale", "nzv"))

p_nb_final <- predict(fit_nb_final, newdata = validation, type = "prob")
```

Generalized linear model by likelihood based boosting

```
fit_glmboost_final <- train(default.payment.next.month ~ .,
                      data = df_trainup,
                      method = "glmboost",
                      trControl = trainControl(method = "cv", number = 10),
                      preProcess = c("center", "scale", "nzv"))

p_glmboost_final <- predict(fit_glmboost_final, newdata = validation, type = "prob")
```

Linear Discriminant Analysis

```
fit_lda_final <- train(default.payment.next.month ~ .,
                 data = df_trainup,
                 method = "lda",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale", "nzv"))

p_lda_final <- predict(fit_lda_final, newdata = validation, type = "prob")
```

Lasso and Elastic-Net Regularized Generalized Linear Models

```
fit_glmnet_final <- train(default.payment.next.month ~ .,
                    data = df_trainup,
                    method = "glmnet",
                    trControl = trainControl(method = "cv", number = 10),
```

```
                    preProcess = c("center", "scale", "nzv"))

p_glmnet_final <- predict(fit_glmnet_final, newdata = validation, type = "prob")
```

Final Ensemble

And now make the final Ensemble.

```
p_hat_final <- (p_rpart_final+
            p_glm_final+
            p_gbm_final+
            p_nb_final+
            p_glmboost_final+
            p_lda_final+
            p_glmnet_final)/7

y_pred_final <- factor(apply(p_hat_final, 1, which.max))

y_pred_final <- factor(y_pred_final, labels = c("Default Next Month", "Normal"))
```

Results

Final Model

and the results of our Ensemble, the **Final Model**, are as follows:

```
confusionMatrix(y_pred_final, validation$default.payment.next.month)
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction             Default Next Month Normal
##   Default Next Month                  790    823
##   Normal                              538   3850
##
##                Accuracy : 0.7732
##                  95% CI : (0.7624, 0.7837)
##     No Information Rate : 0.7787
##     P-Value [Acc > NIR] : 0.8512
##
##                   Kappa : 0.3889
##
##  Mcnemar's Test P-Value : 1.38e-14
##
##             Sensitivity : 0.5949
##             Specificity : 0.8239
##          Pos Pred Value : 0.4898
##          Neg Pred Value : 0.8774
##              Prevalence : 0.2213
##          Detection Rate : 0.1316
```

```
##    Detection Prevalence : 0.2688
##       Balanced Accuracy : 0.7094
##
##        'Positive' Class : Default Next Month
##
```

Let's interpret these results.

The **Accuracy** is the ratio of correct predictions to total predictions made, the accuracy of our final model was 0.7732045, this means that 77.32% of the time we predict classes well, but due to the prevalence of one class in the data we may want to look at other metrics like the **Balanced Accuracy**, this metric is the average between the sensitivity and specificity.

The **Sensitivity** (also called **Recall**) is the number of correct positive predictions, i.e. we predict $\widehat{Y} = 1$ and indeed $Y = 1$, divided by the total number of positives, all the times that $Y = 1$. In our case, the sensitivity represents the number of times we correctly predict a default payment the next month, divided by the total number of cases. So we can correctly predict a default payment next month 59.49% of the times.

The **Specificity** is the number of correct negative predictions, divided by the total number of negatives, i.e. $\widehat{Y} = 0$ and indeed $Y = 0$ divided by all the times that $Y = 0$. In our case, the specificity represents the number of times we correctly predict a non-default payment next month (normal payment), divided by the total number of normal payment cases. Therefore, we are able to correctly predict a normal payment 82.39% of the times.

If we remember, the **Balanced Accuracy** was the average of this two last metrics ($\frac{Sensitivity+Specificity}{2}$), and we may prefer to use it instead of Accuracy due to the imbalance, because note that if we just predict normal payment we can achieve a high accuracy due to prevalence. However, if we use Balanced Accuracy we weight our predictive ability for both classes equivalently and is more representative of our predictive power. In our case, we have correct predictions 70.94% of the times by weighting both classes equally (taking into account the prevalence).

Conclusion

By way of conclusion, it should be noted that although we can predict better than mere chance, our predictive power is not very high, we are able to anticipate only 6 out of 10 defaults approximately and our balanced accuracy is around 70%. It must be taken into account that we will not always be able to predict everything we want. However, with the information that the model gives us we can make several decisions, for example, we could anticipate the losses for our company from the default of certain groups of customers and make future decisions based on this, in turn, we could even start an information campaign for the correct management of funds, and the recipients could be only the group of customers of which we predict a default next month, among other options.

Going forward, we should increase the predictive capability of the models if possible.