7313 Prediction

Haihui Li

15 December 2019

Brief

To sum up, I use random forest with mtry = 3, and ntrees = 10 to predict the unknowns. I've tried out Logit, LDA, QDA, Rpart, Xgbtree, and Random Forest, with different feature selection and with or without PCA dimension reduction. The best model is Random Forest without dimension reduction.

I merged the IZettle Order and IZettle Annual Report data sets by "merchant_id". I only took the 2014 annual report data grouped by "merchant_id".

The feature set is:

{"purchase_amount", "gender", "merchant_id", "country", "purchase_month", "birthyear","MissingYear","ARPCA1","ARPCA2"}.

Among them, ","ARPCA1" ,"ARPCA2" are the two Principal Components of the 2014's data in IZettle Annual Report data set.

The other feature set is:

{"target","gender","merchant_id","country","purchase_month","MissingYear","year_cate","amount_cate" }

A guide to this code file is that:

- Data PreProcessing: Order – Processing the IZettle Order data
- Preprocessing AR data – Processing the IZettle AR data
- Merging Data: Order and AR – Merging IZettle Order and IZettle AR data
- Random Forest and Bagging – Training and tuning the used model
- Predicting Unknowns – Predicting the unknowns with Random Forest model
- Other Attempts – Details of other models' fitting process, for reference

In terms of data processing, I did the following to process the IZettle Order data:

1. Categorizing dates by month
2. Exchanging currency

3. Fixing strange values of purchase amount:
    a. Setting negative values to zero
    b. Adding cents
    c. Eliminating extreme values after trying cutting the variable with different bins. Omitting the observations with purchase amount larger than 50000
4. Fixing birth year:
    a. Filling the NAs with the mean
    b. Ruling out birth year > 2004
5. Adding three features:
    a. MissingYear: whether birth year is NA
    b. year_cate - four categories of birth year: fake, old, middle,young
    c. amount_cate – seven categories of purchase amounts

I did the following to process the IZettle Annual Report data:

1. Filling NAs with different strategies
2. Extracting the 2014's data
3. PCA on the extracted data

I trained the random forest model by changing mtry, from 1 to the number of features – 1, while holding ntrees = 10 fixed. The model is first trained on a 50/50 split training set, with Test Accuracy of 70%; then, on the full data set with Training Accuracy still being 70%.

As for other models and feature sets, they are not as good as the random forest model. Regarding PCA, I generated over 35 PCs and used only the first 26 of them, but there isn't significant improvement.

Data

Data PreProcessing: Order

```
library(caret)

library(doSNOW)

library(RMySQL)

library(dplyr)
```

```r
library(ggplot2)
library(psych)

library(lubridate)

library(tidyr)

library(pROC)

library(AUC)

#=============================== Getting Data =========================
============
rm(list = ls())
#setwd("C:/Users/41506/7313/izttle/prediction")
#con = dbConnect(MySQL(), dbname = "ToyStorey",
 # host = "db1.cfda.se", port = 3306, user = "toystorey",
  #        password = "toys@sse")
#df = fetch(dbSendQuery(con,"SELECT * FROM IZettle.IZettleOrder"), n=-1
)
#write.csv(df,"izt_order.csv")
setwd("D:/7313 prediction model")


iz = read.csv('izt_order.csv')
iz = iz[,-1]

#=============================== Preparing Data =======================
==============
#nrow(iz[is.na(iz$target),])/nrow(iz)

# Hold out the test data(rows whose target is NA)

#iz = iz[!is.na(iz$target),]

# Drop 'device': it is the target itself
dt_iz = subset(iz,select = -device)

# Turn datastamp into month (a factor with 12 levels)
dt_iz$datestamp = as.Date(dt_iz$datestamp)
dt_iz$purchase_month = months(dt_iz$datestamp)
dt_iz$purchase_month = as.factor(dt_iz$purchase_month)



# Factorizing(target,currency,gender,country, merchant_id)
dt_iz$currency = as.factor(dt_iz$currency)
dt_iz$gender = as.factor(dt_iz$gender)
dt_iz$country = as.factor(dt_iz$country)
# Target needs other level names!!!!!!!
```

```r
dt_iz$target = ifelse(dt_iz$target == 1, "Desktop","Others")
dt_iz$target = as.factor(dt_iz$target)
dt_iz$merchant_id = as.factor(dt_iz$merchant_id)
str(dt_iz)
```

```
## 'data.frame':    1150541 obs. of  11 variables:
##  $ id             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ datestamp      : Date, format: "2015-10-03" "2015-11-17" ...
##  $ cid            : num  1.32e+09 3.65e+08 6.06e+08 4.74e+08 1.35e+0
9 ...
##  $ purchase_amount: int  11200 24800 33200 35000 43300 13400 19500 1
3800 6900 64400 ...
##  $ currency       : Factor w/ 3 levels "EUR","NOK","SEK": 3 3 3 3 3
3 3 3 3 2 ...
##  $ birthyear      : int  NA 1981 1947 1967 NA 1956 1971 1949 1956 NA
 ...
##  $ gender         : Factor w/ 3 levels "female","male",..: 3 1 1 1 3
 2 1 1 1 3 ...
##  $ merchant_id    : Factor w/ 12 levels "258643","2026296",..: 5 5 5
 5 5 5 5 5 5 3 ...
##  $ country        : Factor w/ 3 levels "fi","no","se": 3 3 3 3 3 3 3
 3 3 2 ...
##  $ target         : Factor w/ 2 levels "Desktop","Others": 2 NA 2 NA
 2 1 1 1 NA NA ...
##  $ purchase_month : Factor w/ 12 levels "八月","二月",..: 9 8 8 7 9 7
 8 7 8 9 ...
```

```r
# === Purchase Amount =================
# Drop Currency, because it provides the same information as country
table(dt_iz$currency,dt_iz$country)
```

```
##
##             fi     no     se
##   EUR 157874      0      0
##   NOK      0  86902      0
##   SEK      0      0 905765
```

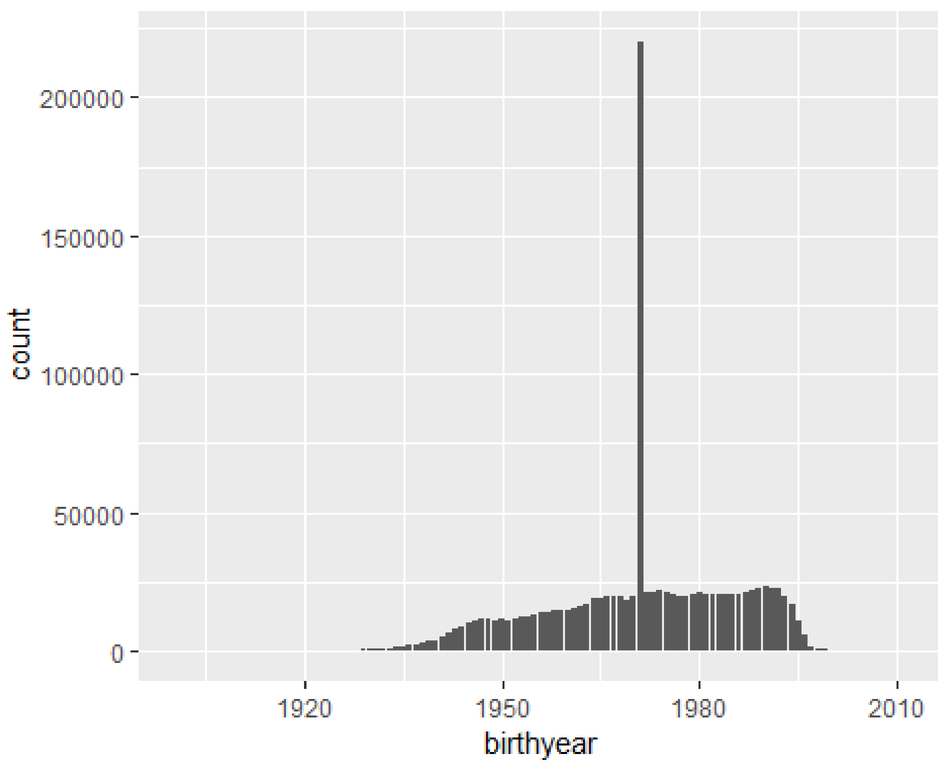
```r
#1. Keep Purchase Amount > 0(It can't be negative)
dt_iz[dt_iz$purchase_amount < 0,"purchase_amount"] = 0

#2. Divide Purchase Amount by 100.(It hasn't properly indicated "cent")
dt_iz$purchase_amount = dt_iz$purchase_amount/100

#3. Currency exchange: changing other currencies into SEK
eur_index = which(dt_iz$currency == "EUR")
dt_iz[eur_index,"purchase_amount"] = dt_iz[eur_index,"purchase_amount"]
*10
dt_iz[eur_index,"currency"] = "SEK"

nok_index = which(dt_iz$currency == "NOK")
```

```r
dt_iz[nok_index,"purchase_amount"] = dt_iz[nok_index,"purchase_amount"]
*1.04
dt_iz[nok_index,"currency"] = "SEK"



#==== Birth Year Modifying
# 1. Add a "MissingYear" Value
dt_iz$MissingYear = ifelse(is.na(dt_iz$birthyear),"Y","N")
dt_iz$MissingYear = as.factor(dt_iz$MissingYear)


#2. Add a "fake_year" feature (Yes: reporting fake birth year; No: repo
rting real birth year)
# Identify fake birthyear (birthyear < 1965 and birthyear = 2011)

dt_iz$FakeYear = rep("N",length(dt_iz$id))
dt_iz$FakeYear = ifelse(dt_iz$birthyear <= 1965|dt_iz$birthyear == 2011
, 'Y','N')
dt_iz$FakeYear[is.na(dt_iz$FakeYear)] = 'Y'
dt_iz$FakeYear = as.factor(dt_iz$FakeYear)


#3. Filling birthyear's none values with average
dt_iz$birthyear[is.na(dt_iz$birthyear)] = round(mean(dt_iz$birthyear,na
.rm = T))

summary(dt_iz)

##        id             datestamp                 cid
##  Min.   :      1   Min.   :2014-01-01   Min.   :9.007e+06
##  1st Qu.: 287665   1st Qu.:2014-09-19   1st Qu.:4.467e+08
##  Median : 575329   Median :2015-03-03   Median :6.610e+08
##  Mean   : 575329   Mean   :2015-02-24   Mean   :2.345e+30
##  3rd Qu.: 862994   3rd Qu.:2015-09-01   3rd Qu.:1.324e+09
##  Max.   :1150659   Max.   :2016-01-02   Max.   :2.678e+31
##
##  purchase_amount    currency        birthyear       gender
##  Min.   :     0.0   EUR:      0   Min.   :1901   female:643881
##  1st Qu.:   159.0   NOK:      0   1st Qu.:1963   male  :291142
##  Median :   251.0   SEK:1150541   Median :1971   none  :215518
##  Mean   :   489.3                 Mean   :1971
##  3rd Qu.:   452.3                 3rd Qu.:1982
##  Max.   :580948.0                 Max.   :2011
##
##   merchant_id      country          target        purchase_month
##  3642550:891604   fi:157874   Desktop:381823   十二月 :156070
##  258643 :137647   no: 86902   Others :538069   十一月 :119544
##  2723490: 60020   se:905765   NA's   :230649   九月   :114898
```

```
##  9402067: 24036                                    八月    :109348
##  5258736: 18076                                    十月    :108825
##  6394740:  5393                                    一月    : 92301
##  (Other): 13765                                    (Other):449555
##  MissingYear FakeYear
##  N:951173    N:616448
##  Y:199368    Y:534093
##
##
##
##
##
```

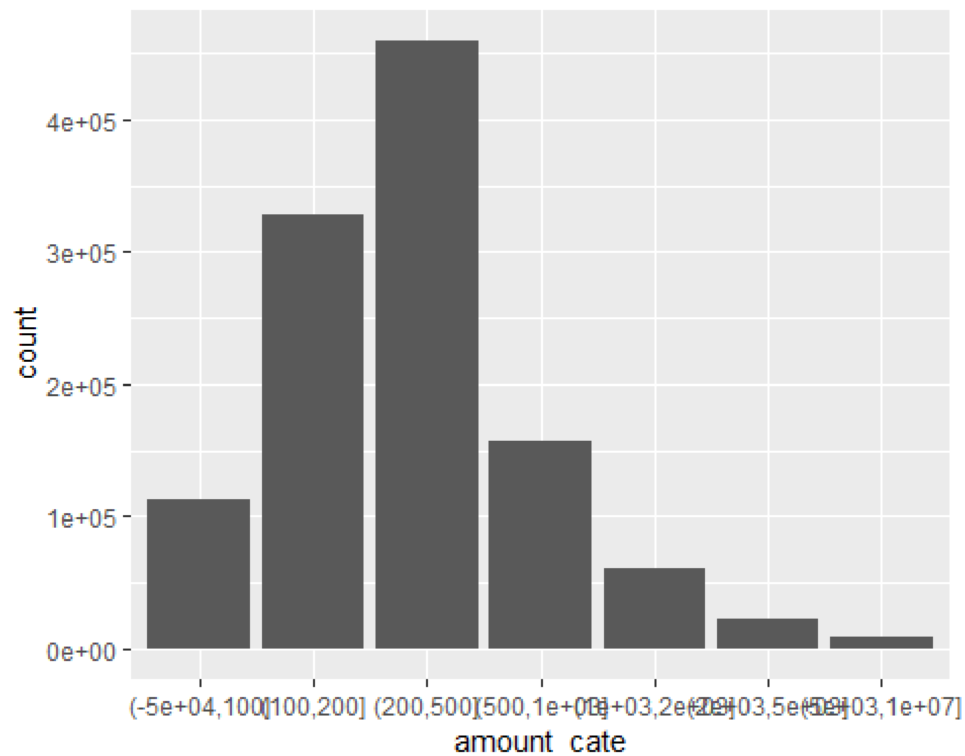
```r
ggplot(dt_iz, aes(x=birthyear)) + geom_histogram(stat="count")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```r
#ggplot(dt_iz, aes(x=purchase_amount)) + geom_histogram()

# Adding Feature year_cate: birth year into four groups: fake, old, mid
dle, young
# Turning num to int
dt_iz$birthyear = as.integer(dt_iz$birthyear)

cut_year = c(1899,1964,1980,2000,2019)
year_cate = cut(dt_iz$birthyear,cut_year)
dt_iz$year_cate = year_cate
```

```
# Adding Feature amount_cate: purchase amout: little, more, much more, very much
dt_iz$purchase_amount = as.integer(dt_iz$purchase_amount)
cut_amount = c(-50000,100,200,500,1000,2000,5000,10000000)
amount_cate = cut(dt_iz$purchase_amount,cut_amount)
dt_iz$amount_cate = amount_cate

ggplot(dt_iz, aes(x=amount_cate)) + geom_histogram(stat="count")

## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

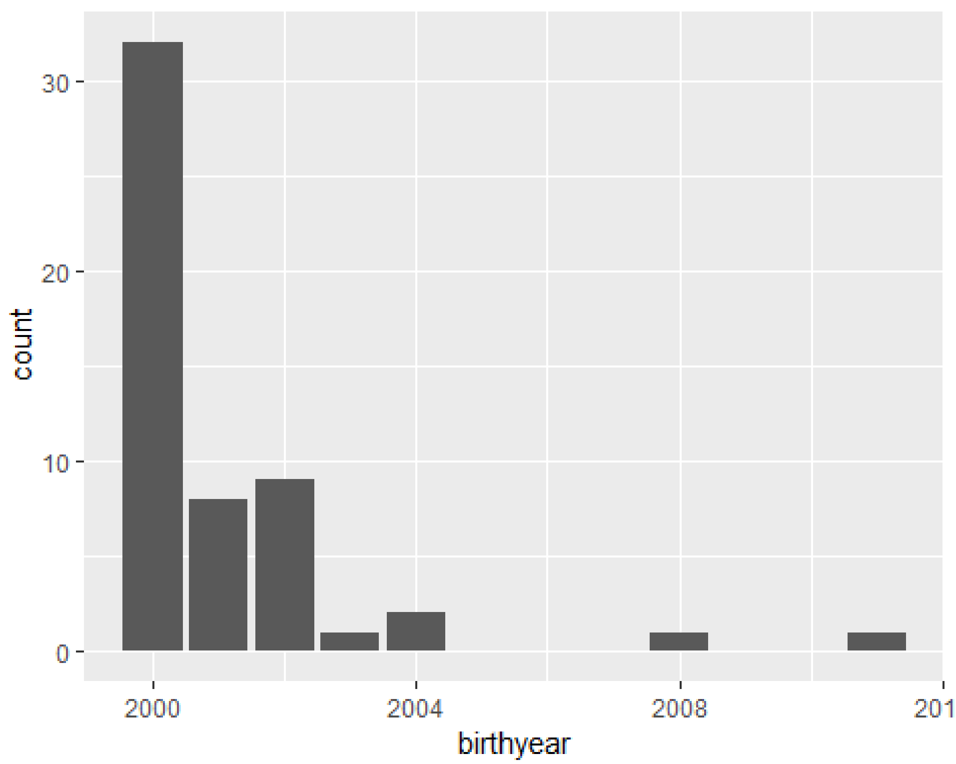



```
str(dt_iz)

## 'data.frame':    1150541 obs. of  15 variables:
##  $ id             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ datestamp      : Date, format: "2015-10-03" "2015-11-17" ...
##  $ cid            : num  1.32e+09 3.65e+08 6.06e+08 4.74e+08 1.35e+0
9 ...
##  $ purchase_amount: int  112 248 332 350 433 134 195 138 69 669 ...
##  $ currency       : Factor w/ 3 levels "EUR","NOK","SEK": 3 3 3 3 3
3 3 3 3 3 ...
##  $ birthyear      : int  1971 1981 1947 1967 1971 1956 1971 1949 195
6 1971 ...
##  $ gender         : Factor w/ 3 levels "female","male",..: 3 1 1 1 3
 2 1 1 1 3 ...
##  $ merchant_id    : Factor w/ 12 levels "258643","2026296",..: 5 5 5
```

```
 5 5 5 5 5 5 3 ...
##  $ country       : Factor w/ 3 levels "fi","no","se": 3 3 3 3 3 3 3
 3 3 2 ...
##  $ target        : Factor w/ 2 levels "Desktop","Others": 2 NA 2 NA
 2 1 1 1 NA NA ...
##  $ purchase_month : Factor w/ 12 levels "八月","二月",..: 9 8 8 7 9 7
 8 7 8 9 ...
##  $ MissingYear   : Factor w/ 2 levels "N","Y": 2 1 1 1 2 1 1 1 1 2
...
##  $ FakeYear      : Factor w/ 2 levels "N","Y": 2 1 2 1 2 2 1 2 2 2
...
##  $ year_cate     : Factor w/ 4 levels "(1.9e+03,1.96e+03]",..: 2 3
1 2 2 1 2 1 1 2 ...
##  $ amount_cate   : Factor w/ 7 levels "(-5e+04,100]",..: 2 3 3 3 3
2 2 2 1 4 ...
```

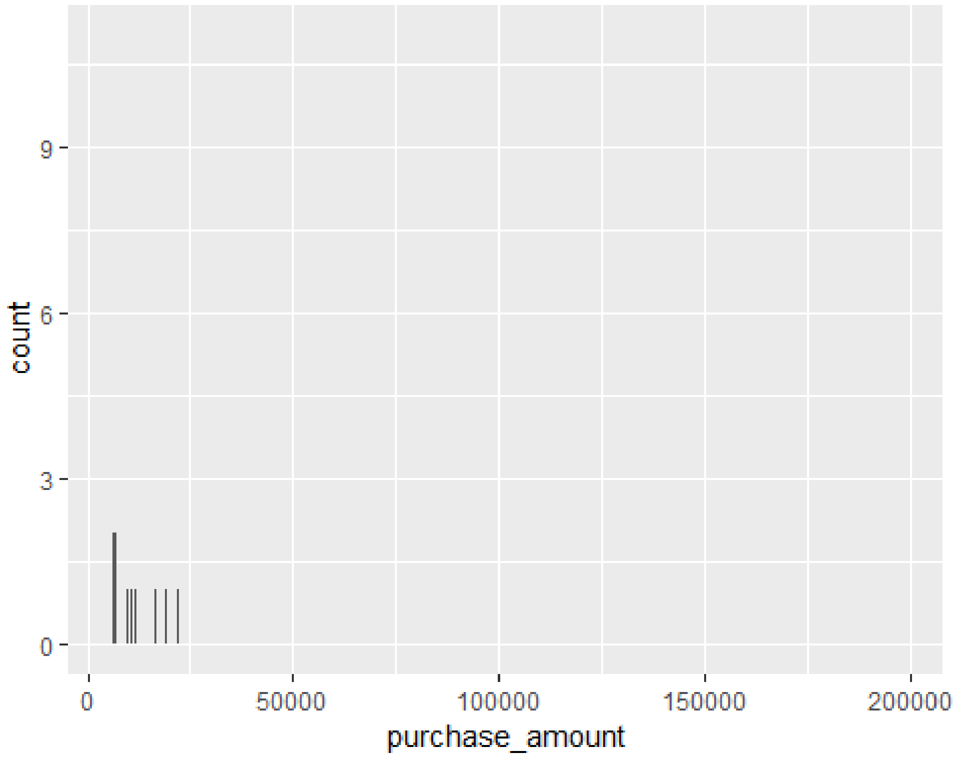
```
ggplot(dt_iz %>% filter(birthyear >=2000), aes(x=birthyear)) + geom_his
togram(stat="count")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```
ggplot(dt_iz %>% filter(purchase_amount >=5000,purchase_amount <200000)
, aes(x=purchase_amount)) + geom_histogram(stat="count")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```
summary(dt_iz)

##        id             datestamp                  cid
##  Min.   :      1   Min.   :2014-01-01   Min.   :9.007e+06
##  1st Qu.: 287665   1st Qu.:2014-09-19   1st Qu.:4.467e+08
##  Median : 575329   Median :2015-03-03   Median :6.610e+08
##  Mean   : 575329   Mean   :2015-02-24   Mean   :2.345e+30
##  3rd Qu.: 862994   3rd Qu.:2015-09-01   3rd Qu.:1.324e+09
##  Max.   :1150659   Max.   :2016-01-02   Max.   :2.678e+31
##
##  purchase_amount     currency          birthyear         gender
##  Min.   :     0.0   EUR:      0    Min.   :1901   female:643881
##  1st Qu.:   159.0   NOK:      0    1st Qu.:1963   male  :291142
##  Median :   251.0   SEK:1150541    Median :1971   none  :215518
##  Mean   :   489.3                  Mean   :1971
##  3rd Qu.:   452.0                  3rd Qu.:1982
##  Max.   :580948.0                  Max.   :2011
##
##   merchant_id      country          target         purchase_month
##  3642550:891604   fi:157874   Desktop:381823   十二月 :156070
##  258643 :137647   no: 86902   Others :538069   十一月 :119544
##  2723490: 60020   se:905765   NA's   :230649   九月   :114898
##  9402067: 24036                                八月   :109348
##  5258736: 18076                                十月   :108825
##  6394740:  5393                                一月   : 92301
##  (Other): 13765                                (Other):449555
```

```
##  MissingYear FakeYear                 year_cate   
##  N:951173    N:616448   (1.9e+03,1.96e+03] :315921  
##  Y:199368    Y:534093   (1.96e+03,1.98e+03]:522179  
##                         (1.98e+03,2e+03]   :312419  
##                         (2e+03,2.02e+03]   :    22  
##                                                     
##                                                     
##                                                     
##          amount_cate    
##  (-5e+04,100] :112911  
##  (100,200]    :328187  
##  (200,500]    :459476  
##  (500,1e+03]  :157597  
##  (1e+03,2e+03]: 60630  
##  (2e+03,5e+03]: 23409  
##  (5e+03,1e+07]:  8331
```

Preprocessing AR data

```
#============================== Getting Izettle Merchant Data =========
===========


con = dbConnect(MySQL(), dbname = "ToyStorey",
                host = "db1.cfda.se", port = 3306,
                user = "toystorey", password = "toys@sse")
df = fetch(dbSendQuery(con, "Select *
                       FROM IZettle.IZettleAR"), n=-1)

#============================== Preparing Merchant Data ===============
=====
#  Fixing 2014 AND 2015 Turnover
for (i in 1:nrow(df)){
  if (is.na(df[i,"turnover"])){
    next
  }
  else {
    if (df[i,"year"] == 2014) {
      df[i,"turnover"] = df[i,"turnover"]*100
    }
    else if (df[i,"year"] == 2015){
      df[i,"turnover"] = df[i,"turnover"]*100
    }
    else
    {next}

  }
}

# ========================== Data Description ========================
```

```
===

# assests, equity, employees, turnoverm ebit, netresults have NAs

#========================== More Data Preparasion ====================
====
df$merchant_id = as.factor(df$merchant_id)
str(df)
# Drop merchant_id = 8244704, becasue it has basically no values

df= df %>%
      filter(merchant_id != "8244704")

# merchant_id = 2026296, missing 2014 turnover, so fill in 2015's turno
ver/ ebit ratio * 2014 EBIT
df[df$merchant_id == "2026296" & df$year == 2014,"turnover"] =
  df[df$merchant_id == "2026296" & df$year == 2015,"turnover"] *
  df[df$merchant_id == "2026296" & df$year == 2014,"ebit"]/df[df$mercha
nt_id == "2026296" & df$year == 2015,"ebit"]


# merchant_id = 2756123, misisng employess values in 2014, 2015, 2016
# Use 2017's employee number 1
df[df$merchant_id == "2756123"&(df$year %in% c(2014,2015,2016)),"employ
ees"] = df[df$merchant_id == "2756123"& df$year == 2017,"employees"]

# merchant_id = 3642550 missing all values in 2018, drop it
df = df %>%
        filter(!(merchant_id == "3642550" & year == 2018))

# merchant_id = 4218266 missing employees value in 2016
# Use 2017's employees value
df[df$merchant_id == "4218266"&df$year == 2016,"employees"] = df[df$mer
chant_id == "4218266"&df$year == 2017,"employees"]

# merchant_id = 4218266 missing turnover value in 2014
# use the transaction data to fill in
df[df$merchant_id == "4218266"&df$year == 2014,"turnover"] =
                                                        dt_iz %>%
                                                          filter(mercha
nt_id == 4218266, year == 2014) %>%
                                                          summarise(sal
es = sum(purchase_amount)/1000)
# merchant_id = 4218266 missing turnover value in 2015
# use the transaction data to fill in
df[df$merchant_id == "4218266"&df$year == 2015,"turnover"] =
  dt_iz %>%
  filter(merchant_id == 4218266, year == 2015) %>%
  summarise(sales = sum(purchase_amount)/1000)
```

```r
# merchant_id = 4218266 missing employees value in 2014, 2015
# Use 2017's employees value
df[df$merchant_id == "4218266"&df$year == 2014,"employees"] = df[df$mer
chant_id == "4218266"&df$year == 2017,"employees"]
df[df$merchant_id == "4218266"&df$year == 2015,"employees"] = df[df$mer
chant_id == "4218266"&df$year == 2017,"employees"]


# merchant_id = 5913810 missing assets, equity, employees value in 2014
# Use 2015's value
df[df$merchant_id == "5913810"&df$year == 2014,"assets"] = df[df$mercha
nt_id == "5913810"&df$year == 2015,"assets"]
df[df$merchant_id == "5913810"&df$year == 2014,"equity"] = df[df$mercha
nt_id == "5913810"&df$year == 2015,"equity"]
df[df$merchant_id == "5913810"&df$year == 2014,"employees"] = df[df$mer
chant_id == "5913810"&df$year == 2015,"employees"]


# merchant_id = 6394740 miss netresult in 2017 and 2018
# Use 2017's EBIT and 2018's EBIT to approximate
df[df$merchant_id == "6394740"&df$year == 2017,"netresult"] = df[df$mer
chant_id == "6394740"&df$year == 2017,"ebit"]
df[df$merchant_id == "6394740"&df$year == 2018,"netresult"] = df[df$mer
chant_id == "6394740"&df$year == 2018,"ebit"]


# Drop the rows with no values at all. The reason is that those compani
es may
# not even exist back then




df_14 = df %>%
  filter(year == 2014)

# Use 2026296's data
df_14[df_14$merchant_id == 4218266,]$assets = 729
df_14[df_14$merchant_id == 4218266,]$equity = 107
df_14[df_14$merchant_id == 4218266,]$turnover = 95
df_14[df_14$merchant_id == 4218266,]$ebit = 38
df_14[df_14$merchant_id == 4218266,]$netresult = 29
```

Merging Data: Order and AR

```r
library(readr)
ar2014 <- read_csv("D:/7313 prediction model/ar2014.csv", col_types = c
ols(X1 = col_skip()))
```

```r
ar2014$merchant_id = as.factor(ar2014$merchant_id)

dt_iz =
left_join(dt_iz, ar2014, by = "merchant_id")

dt_iz = subset(dt_iz, select = -c(year))

summary(dt_iz %>% filter(merchant_id == "8244704"))
```

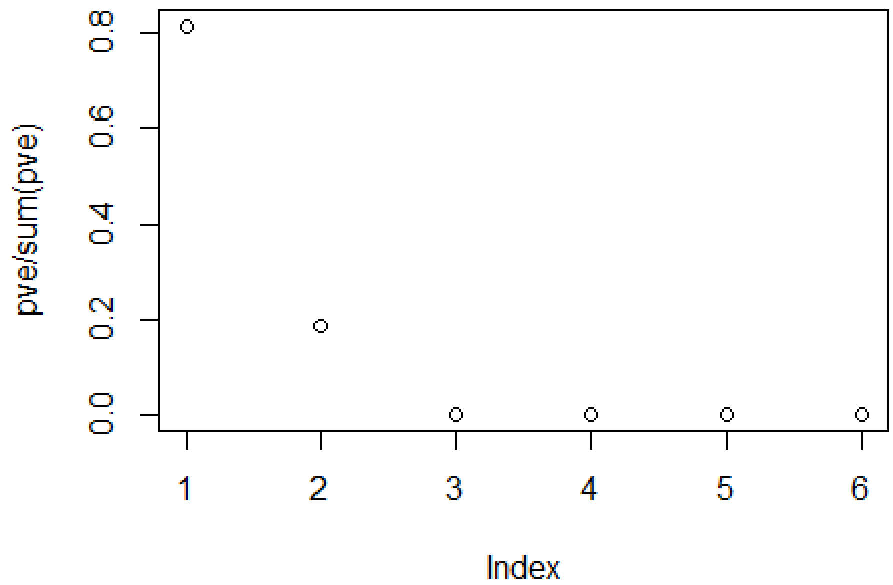
```
##        id             datestamp                  cid
##  Min.   :    531   Min.   :2014-01-02   Min.   :2.180e+08
##  1st Qu.: 228723   1st Qu.:2014-08-06   1st Qu.:4.540e+08
##  Median : 501345   Median :2015-01-08   Median :5.996e+08
##  Mean   : 519923   Mean   :2015-01-22   Mean   :1.187e+30
##  3rd Qu.: 796177   3rd Qu.:2015-08-14   3rd Qu.:7.214e+08
##  Max.   :1149607   Max.   :2015-12-30   Max.   :2.675e+31
##
##  purchase_amount  currency     birthyear       gender
##  Min.   :  33.0   EUR:   0   Min.   :1931   female:920
##  1st Qu.: 176.0   NOK:   0   1st Qu.:1965   male  :151
##  Median : 254.0   SEK:1097   Median :1976   none  : 26
##  Mean   : 297.1              Mean   :1973
##  3rd Qu.: 389.0              3rd Qu.:1983
##  Max.   :1531.0              Max.   :1995
##
##  merchant_id         country       target     purchase_month MissingYe
ar
##  Length:1097        fi:   0   Desktop:420   十二月  :170    N:1071

##  Class :character   no:   0   Others :437   十一月  :126    Y:  26

##  Mode  :character   se:1097   NA's   :240   一月    :125

##                                             十月    : 94

##                                             三月    : 85

##                                             八月    : 83

##                                             (Other):414

##  FakeYear                  year_cate            amount_cate       assets

##  N:792     (1.9e+03,1.96e+03] :269   (-5e+04,100] : 41   Min.   : NA

##  Y:305     (1.96e+03,1.98e+03]:451   (100,200]    :322   1st Qu.: NA

##            (1.98e+03,2e+03]   :377   (200,500]    :634   Median : NA
```

```
##           (2e+03,2.02e+03]   :  0   (500,1e+03]  : 86   Mean   :NaN
##                                     (1e+03,2e+03]: 14   3rd Qu.: NA
##                                     (2e+03,5e+03]:  0   Max.   : NA
##                                     (5e+03,1e+07]:  0   NA's   :1097
##      equity       employees        turnover          ebit
##  Min.   : NA    Min.   : NA    Min.   : NA    Min.   : NA
##  1st Qu.: NA    1st Qu.: NA    1st Qu.: NA    1st Qu.: NA
##  Median : NA    Median : NA    Median : NA    Median : NA
##  Mean   :NaN    Mean   :NaN    Mean   :NaN    Mean   :NaN
##  3rd Qu.: NA    3rd Qu.: NA    3rd Qu.: NA    3rd Qu.: NA
##  Max.   : NA    Max.   : NA    Max.   : NA    Max.   : NA
##  NA's   :1097   NA's   :1097   NA's   :1097   NA's   :1097
##    netresult
##  Min.   : NA
##  1st Qu.: NA
##  Median : NA
##  Mean   :NaN
##  3rd Qu.: NA
##  Max.   : NA
##  NA's   :1097
```

```r
dt_iz[dt_iz$merchant_id == "8244704", "assets"] = 0
dt_iz[dt_iz$merchant_id == "8244704", "equity"] = 0
dt_iz[dt_iz$merchant_id == "8244704", "employees"] = 0


dt_iz %>%
  filter(merchant_id == "8244704") %>%
  tally(purchase_amount)

dt_iz[dt_iz$merchant_id == "8244704", "turnover"] = 0
dt_iz[dt_iz$merchant_id == "8244704", "ebit"] = 0
dt_iz[dt_iz$merchant_id == "8244704", "netresult"] = 0


summary(dt_iz %>% filter(merchant_id == "8244704"))
```

```
##        id            datestamp                cid
##  Min.   :    531   Min.   :2014-01-02   Min.   :2.180e+08
##  1st Qu.: 228723   1st Qu.:2014-08-06   1st Qu.:4.540e+08
##  Median : 501345   Median :2015-01-08   Median :5.996e+08
##  Mean   : 519923   Mean   :2015-01-22   Mean   :1.187e+30
##  3rd Qu.: 796177   3rd Qu.:2015-08-14   3rd Qu.:7.214e+08
##  Max.   :1149607   Max.   :2015-12-30   Max.   :2.675e+31
##
```

```
##  purchase_amount  currency     birthyear       gender
##  Min.   :  33.0   EUR:   0   Min.   :1931   female:920
##  1st Qu.: 176.0   NOK:   0   1st Qu.:1965   male  :151
##  Median : 254.0   SEK:1097   Median :1976   none  : 26
##  Mean   : 297.1              Mean   :1973
##  3rd Qu.: 389.0              3rd Qu.:1983
##  Max.   :1531.0              Max.   :1995
##
##  merchant_id         country       target    purchase_month MissingYe
ar
##  Length:1097        fi:   0   Desktop:420   十二月 :170    N:1071

##  Class :character   no:   0   Others :437   十一月 :126    Y:  26

##  Mode  :character   se:1097   NA's   :240   一月   :125

##                                             十月   : 94

##                                             三月   : 85

##                                             八月   : 83

##                                             (Other):414

##  FakeYear                year_cate           amount_cate      assets
##  N:792    (1.9e+03,1.96e+03] :269   (-5e+04,100] : 41   Min.   :0
##  Y:305    (1.96e+03,1.98e+03]:451   (100,200]    :322   1st Qu.:0
##           (1.98e+03,2e+03]   :377   (200,500]    :634   Median :0
##           (2e+03,2.02e+03]   :  0   (500,1e+03]  : 86   Mean   :0
##                                     (1e+03,2e+03]: 14   3rd Qu.:0
##                                     (2e+03,5e+03]:  0   Max.   :0
##                                     (5e+03,1e+07]:  0
##      equity     employees     turnover       ebit       netresult
##  Min.   :0   Min.   :0   Min.   :0   Min.   :0   Min.   :0
##  1st Qu.:0   1st Qu.:0   1st Qu.:0   1st Qu.:0   1st Qu.:0
##  Median :0   Median :0   Median :0   Median :0   Median :0
##  Mean   :0   Mean   :0   Mean   :0   Mean   :0   Mean   :0
##  3rd Qu.:0   3rd Qu.:0   3rd Qu.:0   3rd Qu.:0   3rd Qu.:0
##  Max.   :0   Max.   :0   Max.   :0   Max.   :0   Max.   :0
##

# PCA
colnames(dt_iz[,c(16:21)])

pca.ar = prcomp(dt_iz[,c(16:21)],scale. = T)
pca.ar.data = pca.ar$x
pve = pca.ar$sdev^2
plot(pve/sum(pve))
```




```
pca.ar.data = pca.ar.data[,c(1,2)]
colnames(pca.ar.data) = c("ARPCA1","ARPCA2")


dt_iz = cbind(dt_iz,pca.ar.data)



# Backup the full processed set
dt_full = dt_iz


# Outliers! Dropping Data : birthyear > 2007, purchase_amount > 50000

dt_iz =
  dt_iz %>%
    filter(birthyear < 2008, purchase_amount <= 50000)

summary(dt_iz)

##        id            datestamp                 cid
##  Min.   :      1   Min.   :2014-01-01   Min.   :9.007e+06
##  1st Qu.: 287664   1st Qu.:2014-09-19   1st Qu.:4.467e+08
##  Median : 575332   Median :2015-03-03   Median :6.610e+08
##  Mean   : 575334   Mean   :2015-02-24   Mean   :2.345e+30
##  3rd Qu.: 862999   3rd Qu.:2015-09-01   3rd Qu.:1.324e+09
```

```
##  Max.   :1150659   Max.   :2016-01-02   Max.   :2.678e+31
##
##  purchase_amount   currency        birthyear       gender
##  Min.   :    0.0   EUR:      0   Min.   :1901   female:643879
##  1st Qu.:  159.0   NOK:      0   1st Qu.:1963   male  :291142
##  Median :  251.0   SEK:1150422   Median :1971   none  :215401
##  Mean   :  478.4                 Mean   :1971
##  3rd Qu.:  452.0                 3rd Qu.:1982
##  Max.   :50000.0                 Max.   :2004
##
##  merchant_id         country         target        purchase_month
##  Length:1150422     fi:157873   Desktop:381823   十二月 :156050
##  Class :character   no: 86894   Others :537980   十一月 :119531
##  Mode  :character   se:905655   NA's   :230619   九月   :114892
##                                                  八月   :109336
##                                                  十月   :108815
##                                                  一月   : 92284
##                                                  (Other):449514
##  MissingYear FakeYear                year_cate
##  N:951171    N:616447   (1.9e+03,1.96e+03] :315921
##  Y:199251    Y:533975   (1.96e+03,1.98e+03]:522062
##                         (1.98e+03,2e+03]   :312419
##                         (2e+03,2.02e+03]   :    20
##
##
##
##         amount_cate          assets           equity         employees

##  (-5e+04,100] :112911   Min.   :     0   Min.   :    0   Min.   :  0
.0
##  (100,200]    :328186   1st Qu.:273809   1st Qu.:12480   1st Qu.:129
.0
##  (200,500]    :459475   Median :273809   Median :12480   Median :129
.0
##  (500,1e+03]  :157597   Mean   :230694   Mean   :12624   Mean   :106
.4
##  (1e+03,2e+03]: 60630   3rd Qu.:273809   3rd Qu.:12480   3rd Qu.:129
.0
##  (2e+03,5e+03]: 23409   Max.   :273809   Max.   :23544   Max.   :129
.0
##  (5e+03,1e+07]:  8214

##     turnover           ebit            netresult          ARPCA1

##  Min.   :      0   Min.   :-69213   Min.   :-70051   Min.   :-1.18288
1
##  1st Qu.:191700   1st Qu.:-69213   1st Qu.:-70051   1st Qu.:-1.18288
1
##  Median :191700   Median :-69213   Median :-70051   Median :-1.18288
```

```
1
##  Mean   :149417   Mean   :-53473   Mean   :-54320   Mean   : 0.00007
8
##  3rd Qu.:191700   3rd Qu.:-69213   3rd Qu.:-70051   3rd Qu.:-1.18288
1
##  Max.   :191700   Max.   :  1319   Max.   :   502   Max.   : 4.87461
1
##

##      ARPCA2
##  Min.   :-2.2927453
##  1st Qu.: 0.0652390
##  Median : 0.0652390
##  Mean   :-0.0000219
##  3rd Qu.: 0.0652390
##  Max.   : 2.3836933
##
```

```
dt_iz = dt_iz[!is.na(dt_iz$target),]
colnames(dt_iz)
```

```
##  [1] "id"              "datestamp"       "cid"
##  [4] "purchase_amount" "currency"        "birthyear"
##  [7] "gender"          "merchant_id"     "country"
## [10] "target"          "purchase_month"  "MissingYear"
## [13] "FakeYear"        "year_cate"       "amount_cate"
## [16] "assets"          "equity"          "employees"
## [19] "turnover"        "ebit"            "netresult"
## [22] "ARPCA1"          "ARPCA2"
```

Loading Data

```
#setwd("C:/Users/41506/7313/izttle/prediction")
#source("data_gen.R")
#setwd("C:/Users/41506/7313/izttle/prediction")
features = c("target","purchase_amount",
             "gender","merchant_id",
             "country","purchase_month",
             "birthyear","MissingYear","ARPCA1" ,
             "ARPCA2")
train_df = dt_iz[,features]
```

Splitting Sets

```
set.seed(7313)
indexes <- createDataPartition(train_df$target,
                               times = 1,
                               p = 0.5,
                               list = FALSE)
iz.train <- train_df[indexes,]
iz.test <- train_df[-indexes,]
```

Parelle running Setting

```
library(doSNOW)
cl <- makeCluster(2, type = "SOCK")
registerDoSNOW(cl)
```

Modelling

Training Logit, LDA, and QDA

```
control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 2,
                              classProbs = T,
                              summaryFunction = twoClassSummary)
fit.logit = train(target~., data = iz.train, method = "glm",
                  family = binomial(), trControl =  control, metric = "
Accuracy" )

fit.lda = train(target~., data = iz.train,
                method = "lda",  trControl =  control, metric = "Accura
cy" )

#fit.qda = train(target~., data = iz.train,
                #method = "qda",  trControl =  control,metric = "Accura
cy" )

pred = predict(fit.logit, iz.test)
confusionMatrix(as.factor(pred), iz.test$target)

pred = predict(fit.lda, iz.test)
confusionMatrix(as.factor(pred), iz.test$target)

#saveRDS(fit.lda,"ldamod.Rdata")
#saveRDS(fit.logit,"logitmod.Rdata")
```

Logit is unreliable according to the warnings:“prediction from a rank-deficient fit may be misleading.”

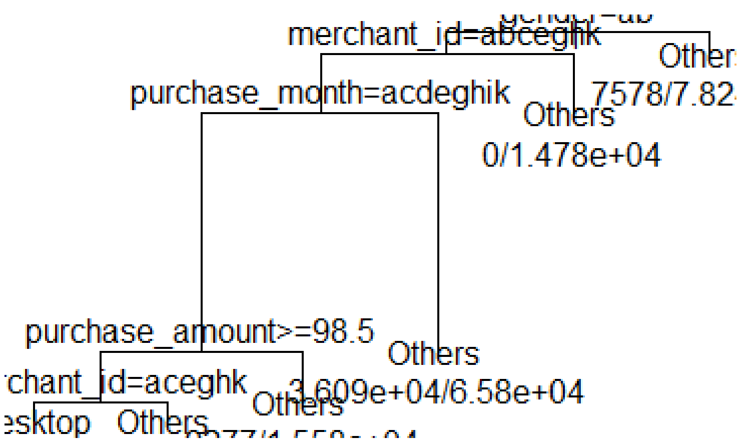
LDA is subject to collinearity.

QDA is off dut to rank deficiency?

Tree

```
library(rpart)

## Warning: package 'rpart' was built under R version 3.5.3
```

```r
set.seed(7313)

fit.rpart = rpart(target~., data = iz.train)
plot(fit.rpart)
text(fit.rpart, use.n = TRUE)
```




```r
pred = predict(fit.rpart, iz.test)
pred = pred[,1]
rpart.class = rep("Others",length(iz.test$target))
# Changing cutoff
rpart.class[pred > 0.55 ] = "Desktop"

confusionMatrix(as.factor(rpart.class), iz.test$target)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction Desktop Others
##    Desktop  122870  76662
##    Others    68041 192328
##
##                Accuracy : 0.6854
##                  95% CI : (0.684, 0.6867)
##     No Information Rate : 0.5849
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3563
```

```
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6436
##             Specificity : 0.7150
##          Pos Pred Value : 0.6158
##          Neg Pred Value : 0.7387
##              Prevalence : 0.4151
##          Detection Rate : 0.2672
##    Detection Prevalence : 0.4339
##       Balanced Accuracy : 0.6793
##
##        'Positive' Class : Desktop
##
```

```
# We try cp = 0.05, but the results deteriorate
#fit.rpart = rpart(target~., data = iz.train, control = rpart.control(c
p = 0.05))
#plot(fit.rpart)
#text(fit.rpart, use.n = TRUE)

#pred = predict(fit.rpart, iz.test)
#pred = pred[,1]
#rpart.class = rep("Others",length(iz.test$target))
#rpart.class[pred > 0.55 ] = "Desktop"
#confusionMatrix(as.factor(rpart.class), iz.test$target)
# Caret training is off......,for some reason
#ctrl = trainControl(method = "repeatedcv",
 #                     number = 5,
  #                    repeats = 3)
#
#train.tree = train(target~., data = iz.train,
 #               labels,
  #              method = "rpart",
   #             trControl = ctrl)
#train.tree

# So we do tree tuning manually, by changeing complexity parameter
cps = seq(0.01,0.05,length.out = 10)
tree.accracy = rep(0,length(cps))

for (i in 1:length(cps)){
  complexitypar = cps[i]
  fit.rpart = rpart(target~., data = iz.train,
                    control = rpart.control(cp = complexitypar))
  pred = predict(fit.rpart, iz.test)
  pred = pred[,1]
  rpart.class = rep("Others",length(iz.test$target))
  rpart.class[pred > 0.55 ] = "Desktop"
  tree.accracy[i] = mean(rpart.class == iz.test$target)
```

```
}
print(tree.accracy)

##  [1] 0.6853605 0.6809552 0.6809552 0.6809552 0.6809552 0.6809552 0.6
671240
##  [8] 0.6671240 0.6671240 0.6671240

print(which.max(tree.accracy))

## [1] 1

# So we do tree tuning manually, by changeing complexity parameter
cps = seq(0.001,0.01,length.out = 10)
tree.accracy = rep(0,length(cps))

for (i in 1:length(cps)){
  complexitypar = cps[i]
  fit.rpart = rpart(target~., data = iz.train,
                    control = rpart.control(cp = complexitypar))
  pred = predict(fit.rpart, iz.test)
  pred = pred[,1]
  rpart.class = rep("Others",length(iz.test$target))
  rpart.class[pred > 0.55 ] = "Desktop"
  tree.accracy[i] = mean(rpart.class == iz.test$target)
}
print(tree.accracy)

##  [1] 0.6941081 0.6941081 0.6888135 0.6888135 0.6888135 0.6888135 0.6
888135
##  [8] 0.6888135 0.6853605 0.6853605

print(which.max(tree.accracy))

## [1] 1

print(cps[which.max(tree.accracy)])

## [1] 0.001

fit.rpart = rpart(target~., data = iz.train,
                    control =
                    rpart.control(cp =   cps[which.max(tree.accracy)]))
  pred = predict(fit.rpart, iz.test)
  pred = pred[,1]
  rpart.class = rep("Others",length(iz.test$target))
  rpart.class[pred > 0.55 ] = "Desktop"
  confusionMatrix(as.factor(rpart.class), iz.test$target)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction Desktop Others
```

```
##    Desktop  109997  59766
##    Others    80914 209224
##
##                Accuracy : 0.6941
##                  95% CI : (0.6928, 0.6954)
##     No Information Rate : 0.5849
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3598
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5762
##             Specificity : 0.7778
##          Pos Pred Value : 0.6479
##          Neg Pred Value : 0.7211
##              Prevalence : 0.4151
##          Detection Rate : 0.2392
##    Detection Prevalence : 0.3691
##       Balanced Accuracy : 0.6770
##
##        'Positive' Class : Desktop
##
```

If we use caret to train, there will be something like"There were missing values in resampled performance measures". In our manual tryouts, it seems 0.01 is the best cp. We try to loop through lower cps. The results of our second loop show that we can boost our tree model a bit, with `cp = 0.001`. So, we wolud have `cp = 0.001`, which makes the model less complex.

Random Forest and Bagging

```
library(randomForest)

numberofvar = ncol(iz.train) -1
rf.acurracy = c(1:numberofvar)

for (i in 1:numberofvar){
  fit.rf = randomForest(target~., data = iz.train, mtry = i ,ntree = 10
)
  pred = predict(fit.rf, iz.test)
  rf.acurracy[i] = mean(pred == iz.test$target)
}

print(paste("Best mtry is:",which.max(rf.acurracy)))

## [1] "Best mtry is: 3"

fit.rf = randomForest(target~., data = iz.train, mtry = which.max(rf.ac
urracy) ,ntree = 10)
```

```r
pred = predict(fit.rf, iz.test)
confusionMatrix(as.factor(pred), iz.test$target)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Desktop Others
##    Desktop  125199  71467
##    Others    65712 197523
##
##                Accuracy : 0.7017
##                  95% CI : (0.7004, 0.703)
##     No Information Rate : 0.5849
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3884
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6558
##             Specificity : 0.7343
##          Pos Pred Value : 0.6366
##          Neg Pred Value : 0.7504
##              Prevalence : 0.4151
##          Detection Rate : 0.2722
##    Detection Prevalence : 0.4276
##       Balanced Accuracy : 0.6951
##
##        'Positive' Class : Desktop
##
```

```r
# Comuptational demanding, so we need a smaller data set
# Still can't run it
#set.seed(7313)
#small.indexes <- createDataPartition(iz.train$target,
#                                     times = 1,
#                                     p = 0.5,
#                                     list = FALSE)
#iz.train.small <- iz.train[indexes,]
#iz.test.small <- iz.train[-indexes,]


#set.seed(7313)
#control = trainControl(method="repeatedcv", number=5,
#                        repeats=2,
 #                       classProbs = T,
  #                      summaryFunction = twoClassSummary)

#tunegrid = expand.grid(.mtry=
 #                         floor(sqrt(ncol(iz.train))):ncol(iz.train) -1
```

```
 )

# From RandomForest to Bagging

#fit.rf <- train(target ~ .,
 #               data = iz.train.small,
  #              method = "rf",
   #             metric = "Accuracy",
    #            tuneGrid = tunegrid,
     #           ntrees = 10,
      #          trControl = control,
       #         na.action = na.exclude)


# Bagging Manually
#fit.bg = randomForest(target~., data = iz.train, mtry = 7 ,ntree = 10)
#pred = predict(fit.bg, iz.test)
#confusionMatrix(as.factor(pred), iz.test$target)
```

Random Forest is better than bagging, with accuracy 70%

Predicting Unknowns

We decide to use RandomForest with the original data the rest is different attempts

```
fit.rf = randomForest(target~., data = train_df, mtry = 3 ,ntree = 15)
pred = predict(fit.rf, train_df)
confusionMatrix(as.factor(pred), train_df$target)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction Desktop Others
##    Desktop  251389 138843
##    Others   130434 399137
##
##                Accuracy : 0.7072
##                  95% CI : (0.7063, 0.7082)
##     No Information Rate : 0.5849
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.399
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6584
##             Specificity : 0.7419
##          Pos Pred Value : 0.6442
```

```
##          Neg Pred Value : 0.7537
##              Prevalence : 0.4151
##          Detection Rate : 0.2733
##    Detection Prevalence : 0.4243
##       Balanced Accuracy : 0.7002
##
##        'Positive' Class : Desktop
##
```

```
dt_null =
dt_full %>%
  filter(is.na(target))
dim(dt_null)
```

```
## [1] 230649     23
```

```
pred.data = dt_null[,features]


pred.for.null = predict(fit.rf,pred.data)
dt_null$device =pred.for.null
dt_null$target = ifelse(dt_null$device == "Desktop",1,0)

#write.csv(dt_null,"prediction_resubmit.csv")
```

Other Attempts

SVM Linear

```
library(e1071)
fit.svml = svm(target~., iz.train, kernel = "linear",scale = T, cost =
1)
pred = predict(fit.svml, iz.test)
confusionMatrix(as.factor(pred), iz.test$target)
```

SVM Poly

```
library(e1071)
fit.svmply = svm(target~., iz.train, kernel = "Polynominal", cost = 1)
pred = predict(fit.svmply, iz.test)
confusionMatrix(as.factor(pred), iz.test$target)
```

Xgboost

```
train.control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 2,
                              classProbs = T,
                              summaryFunction = twoClassSummary,
                              search = "grid")
```

```
tune.grid <- expand.grid(eta = c(0.05, 0.075, 0.1),
                         nrounds = c(50, 75, 100),
                         max_depth = 6:8,
                         min_child_weight = c(2.0, 2.25, 2.5),
                         colsample_bytree = c(0.3, 0.4, 0.5),
                         gamma = 0,
                         subsample = 1)

# Train the xgboost model using 5-fold CV repeated 2 times
fit.xjb <- train(target ~ .,
                 data = iz.train,
                 method = "xgbTree",
                 tuneGrid = tune.grid,
                 trControl = train.control,
                 metric = "Accuracy" )
pred = predict(fit.xjb, iz.test)
confusionMatrix(as.factor(pred), iz.test$target)
#stopCluster(cl)
```

I run this in another computer, the accuracy is 70.4%. No time to run it again.

PCA1

```
#source("data_gen.R")
train_df = dt_iz[,features]
# Turn all the factors into dummy variables
dummy.vars = dummyVars(~., subset(train_df,select = - target))
train.dummy = predict(dummy.vars,subset(train_df,select = - target))
pca.results = prcomp(train.dummy, scale. = T)
pca.data = pca.results$x
pve = pca.results$sdev^2
train_df_pca = data.frame(target = train_df$target, pca.data)
plot(pve/sum(pve))

pca.results = prcomp(train.dummy, scale. = T)
pca.data = pca.results$x
pve = pca.results$sdev^2
train_df_pca = data.frame(target = train_df$target, pca.data)
plot(pve/sum(pve))

# Only take the first 26 PC
train_df_pca = train_df_pca[,c(1:27)]

set.seed(7313)
indexes <- createDataPartition(train_df_pca$target,
                               times = 1,
                               p = 0.5,
                               list = FALSE)
iz.train.pca <- train_df_pca[indexes,]
iz.test.pca <- train_df_pca[-indexes,]
```

Training Logit, LDA, on PCA

```
control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 2,
                              classProbs = T,
                              summaryFunction = twoClassSummary)
fit.logit.pca = train(target~., data = iz.train.pca, method = "glm",
                  family = binomial(), trControl =  control, metric = "
Accuracy" )

fit.lda.pca = train(target~., data = iz.train.pca,
                method = "lda",  trControl =  control, metric = "Accura
cy" )


pred = predict(fit.logit.pca, iz.test.pca)
confusionMatrix(as.factor(pred), iz.test.pca$target)

pred = predict(fit.lda.pca, iz.test.pca)
confusionMatrix(as.factor(pred), iz.test.pca$target)


#saveRDS(fit.lda,"ldamod.Rdata")
#saveRDS(fit.logit,"logitmod.Rdata")
```

Still underperform Random Forest

QDA with PCA

```
fit.qda.pca = train(target~., data = iz.train.pca,
                method = "qda",
                trControl =  control, metric = "Accuracy" )


pred = predict(fit.qda.pca, iz.test.pca)
confusionMatrix(as.factor(pred), iz.test.pca$target)
```

Random Forest and Bagging with PCA

```
library(randomForest)
numberofvar = ncol(iz.test.pca) -1
rf.acurracy = c(1:numberofvar)

for (i in 1:numberofvar){
  fit.rf = randomForest(target~., data = iz.train.pca, mtry = i ,ntree
= 10)
  pred = predict(fit.rf, iz.test.pca)
  rf.acurracy[i] = mean(pred == iz.test.pca$target)
}
```

```
print(paste("Best mtry is:",which.max(rf.acurracy)))

fit.rf.pca = randomForest(target~., data = iz.train.pca, mtry = which.m
ax(rf.acurracy) ,ntree = 10)
pred = predict(fit.rf, iz.test.pca)
confusionMatrix(as.factor(pred), iz.test.pca$target)

# Comuptational demanding, so we need a smaller data set
# Still can't run it
#set.seed(7313)
#small.indexes <- createDataPartition(iz.train$target,
#                              times = 1,
#                              p = 0.5,
#                              list = FALSE)
#iz.train.small <- iz.train[indexes,]
#iz.test.small <- iz.train[-indexes,]


#set.seed(7313)
#control = trainControl(method="repeatedcv", number=5,
#                        repeats=2,
 #                       classProbs = T,
  #                      summaryFunction = twoClassSummary)

#tunegrid = expand.grid(.mtry=
 #                         floor(sqrt(ncol(iz.train))):ncol(iz.train) -1
 )

# From RandomForest to Bagging

#fit.rf <- train(target ~ .,
 #               data = iz.train.small,
  #              method = "rf",
   #             metric = "Accuracy",
    #            tuneGrid = tunegrid,
     #           ntrees = 10,
      #          trControl = control,
       #         na.action = na.exclude)


# Bagging Manually
#fit.bg = randomForest(target~., data = iz.train.pca, mtry = numberofva
r ,ntree = 10)
#pred = predict(fit.bg, iz.test.pca)
#confusionMatrix(as.factor(pred), iz.test.pca$target)
```

On PCA, random Forest gets worse.

LOGIT ON PCA WITH LASSO

```
library(glmnet)
xmat = model.matrix(target~., iz.train.pca)[,-1]
y = iz.train.pca$target


cv.out = cv.glmnet(xmat,y, alpha = 1, family = "binomial")
cv.out$lambda.min

logit.pca = glmnet(xmat,y,family = "binomial", alpha = 1,lambda = cv.ou
t$lambda.min)
pred = predict(logit.pca,newx = model.matrix(target~., iz.test.pca)[,-1
], type = "class")
head(pred)
confusionMatrix(as.factor(pred),iz.test.pca$target)
```

Still not as good as Random Forest, I run it in another computer, because computers at PC labs can not install "glmnet".

Xgboost with PCA

```
train.control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 2,
                              classProbs = T,
                              summaryFunction = twoClassSummary,
                              search = "grid")

tune.grid <- expand.grid(eta = c(0.05, 0.075, 0.1),
                         nrounds = c(50, 75, 100),
                         max_depth = 6:8,
                         min_child_weight = c(2.0, 2.25, 2.5),
                         colsample_bytree = c(0.3, 0.4, 0.5),
                         gamma = 0,
                         subsample = 1)
set.seed(7313)
index.mini = createDataPartition(train_df_pca$target,
                                 times = 1,
                                 p = 0.05,
                                 list = FALSE)
train.mini = train_df_pca[index.mini,]

set.seed(7313)
index.mini.pca = createDataPartition(train.mini$target,
                                 times = 1,
                                 p = 0.5,
                                 list = FALSE)
little.train = train.mini[index.mini.pca,]
little.test = train.mini[-index.mini.pca,]
```

```r
# Train the xgboost model using 5-fold CV repeated 2 times
fit.xjb.pca <- train(target ~ .,
                  data = little.train,
                  method = "xgbTree",
                  tuneGrid = tune.grid,
                  trControl = train.control,
                  metric = "Accuracy" )
pred = predict(fit.xjb.pca, little.test)
confusionMatrix(as.factor(pred), little.test$target)
#stopCluster(cl)
```

Xgboost gets 70% as well. I tried training on the full data set, no improvemnt was made.

Second Feature Set

```r
#setwd("C:/Users/41506/7313/izttle/prediction")
#source("data_gen.R")
#setwd("C:/Users/41506/7313/izttle/prediction")
#colnames(dt_iz)
features = c("target",
             "gender","merchant_id",
             "country","purchase_month",
             "MissingYear","year_cate",
             "amount_cate" )
train_df = dt_iz[,features]
colnames(train_df)
```

Splitting Sets

```r
set.seed(7313)
indexes <- createDataPartition(train_df$target,
                               times = 1,
                               p = 0.5,
                               list = FALSE)
iz.train <- train_df[indexes,]
iz.test <- train_df[-indexes,]
```

Tree with Second Feature set

```r
library(rpart)
set.seed(7313)

fit.rpart = rpart(target~., data = iz.train)
plot(fit.rpart)
text(fit.rpart, use.n = TRUE)

pred = predict(fit.rpart, iz.test)
pred = pred[,1]
rpart.class = rep("Others",length(iz.test$target))
# Changing cutoff
rpart.class[pred > 0.55 ] = "Desktop"
```

```r
confusionMatrix(as.factor(rpart.class), iz.test$target)

# We try cp = 0.05, but the results deteriorate
#fit.rpart = rpart(target~., data = iz.train, control = rpart.control(c
p = 0.05))
#plot(fit.rpart)
#text(fit.rpart, use.n = TRUE)

#pred = predict(fit.rpart, iz.test)
#pred = pred[,1]
#rpart.class = rep("Others",length(iz.test$target))
#rpart.class[pred > 0.55 ] = "Desktop"
#confusionMatrix(as.factor(rpart.class), iz.test$target)
# Caret training is off......,for some reason
#ctrl = trainControl(method = "repeatedcv",
 #                     number = 5,
  #                    repeats = 3)
#
#train.tree = train(target~., data = iz.train,
 #               labels,
  #              method = "rpart",
   #             trControl = ctrl)
#train.tree

# So we do tree tuning manually, by changeing complexity parameter
cps = seq(0.01,0.05,length.out = 10)
tree.accracy = rep(0,length(cps))

for (i in 1:length(cps)){
  complexitypar = cps[i]
  fit.rpart = rpart(target~., data = iz.train,
                    control = rpart.control(cp = complexitypar))
  pred = predict(fit.rpart, iz.test)
  pred = pred[,1]
  rpart.class = rep("Others",length(iz.test$target))
  rpart.class[pred > 0.55 ] = "Desktop"
  tree.accracy[i] = mean(rpart.class == iz.test$target)
}
print(tree.accracy)
print(which.max(tree.accracy))

# So we do tree tuning manually, by changeing complexity parameter
cps = seq(0.001,0.01,length.out = 10)
tree.accracy = rep(0,length(cps))

for (i in 1:length(cps)){
  complexitypar = cps[i]
  fit.rpart = rpart(target~., data = iz.train,
```

```
                          control = rpart.control(cp = complexitypar))
  pred = predict(fit.rpart, iz.test)
  pred = pred[,1]
  rpart.class = rep("Others",length(iz.test$target))
  rpart.class[pred > 0.55 ] = "Desktop"
  tree.accracy[i] = mean(rpart.class == iz.test$target)
}
print(tree.accracy)
print(which.max(tree.accracy))
print(cps[which.max(tree.accracy)])


fit.rpart = rpart(target~., data = iz.train,
                          control =
                          rpart.control(cp =   cps[which.max(tree.accracy)]))
  pred = predict(fit.rpart, iz.test)
  pred = pred[,1]
  rpart.class = rep("Others",length(iz.test$target))
  rpart.class[pred > 0.55 ] = "Desktop"
  confusionMatrix(as.factor(rpart.class), iz.test$target)
```

Random Forest and Bagging with second feature set

```
library(randomForest)
numberofvar = ncol(iz.train) -1
rf.acurracy = c(1:numberofvar)

for (i in 1:numberofvar){
  fit.rf = randomForest(target~., data = iz.train, mtry = i ,ntree = 10
)
  pred = predict(fit.rf, iz.test)
  rf.acurracy[i] = mean(pred == iz.test$target)
}

print(paste("Best mtry is:",which.max(rf.acurracy)))
fit.rf = randomForest(target~., data = iz.train, mtry = which.max(rf.ac
urracy) ,ntree = 10)
pred = predict(fit.rf, iz.test)
confusionMatrix(as.factor(pred), iz.test$target)

# Comuptational demanding, so we need a smaller data set
# Still can't run it
#set.seed(7313)
#small.indexes <- createDataPartition(iz.train$target,
#                                times = 1,
#                                p = 0.5,
#                                list = FALSE)
#iz.train.small <- iz.train[indexes,]
#iz.test.small <- iz.train[-indexes,]
```

```
#set.seed(7313)
#control = trainControl(method="repeatedcv", number=5,
#                        repeats=2,
 #                       classProbs = T,
  #                      summaryFunction = twoClassSummary)

#tunegrid = expand.grid(.mtry=
 #                         floor(sqrt(ncol(iz.train))):ncol(iz.train) -1
 )

# From RandomForest to Bagging

#fit.rf <- train(target ~ .,
 #               data = iz.train.small,
  #              method = "rf",
   #             metric = "Accuracy",
    #            tuneGrid = tunegrid,
     #           ntrees = 10,
      #          trControl = control,
       #         na.action = na.exclude)


# Bagging Manually
#fit.bg = randomForest(target~., data = iz.train, mtry = 7 ,ntree = 10)
#pred = predict(fit.bg, iz.test)
#confusionMatrix(as.factor(pred), iz.test$target)
```