R05546022 謝立成

Statistical Learning Final report

June 11, 2017

Abstract

Required packages :

`devtools`, `rmarkdown`, `rpart`, `glmnet`, `knitr`, `readr`, `stringr`, `ggthemes`, `ggplot2`, `syuzhet`, `SnowballC`, `tm`, `xgboost`

In this report, we will show the full process of text mining and data mining with procedure of model ensembling and predicting. This dataset is from theQuora Question Pairs Competition of Kaggle. (https://www.kaggle.com/c/quora-question-pairs/data) ,with the goal of identifying question pairs that have the same intent. The model includes the assembly of XGboost, Logistic regression (glm), CART,and C5.0.

DATA PREPROCESSING

```
## number of rows:   404290
## number of columns :  6
```

This dataset is from the Featured Prediction Competition of Kaggle, the **Quora Question Pairs** prediction testing dataset.(https://www.kaggle.com/c/quora-question-pairs/data) In this dataset,there are totally 404290 rows of sample question pairs.

id	qid1	qid2	question1
0	1	2	What is the step by step guide to invest in share market in india?
1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?
2	5	6	How can I increase the speed of my internet connection while using a VPN?
3	7	8	Why am I mentally very lonely? How can I solve it?
4	9	10	Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?
5	11	12	Astrology: I am a Capricorn Sun Cap moon and cap rising…what does that say about me?

question2	is_duplicate
What is the step by step guide to invest in share market?	0
What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
How can Internet speed be increased by hacking through DNS?	0
Find the remainder when [math]23^{24}[/math] is divided by 24,23?	0
Which fish would survive in salt water?	0
I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?	1

- id - the id of a training set question pair
- qid1, qid2 - unique ids of each question
- question1, question2 - the full text of each question
- is_duplicate - the target variable, set to 1 if question pairs have the same meaning, and 0 otherwise.

Note that the qid1 and qid2 column represents the label of questions, indicating which question it is, and may occure again in different pairs of questions.

The main goal of this project is to predict weather or not the pairs of the question are duplicated. In this project, we will seperate the analysis into 2 parts, text processing and model analysis. ## Text engineering Since it is a pair of sentence, it is obvious that we identify the string by "meaningful" words. Therefore for each rows, we will deal with 2 strings of sentence into 2 lists.

In this dataset analysis, the package `tm` was used to store the questions into **Corpus**, or sets of text. Below will demonstrate the work of one rows of question pairs.

Stopwords

Stopwords are words that are use to connect sentence in order to make the sentence complete, but are consider as noise in text mining. In the package words *who*, *what*, *when*, *where*, *why*, *how*, *which* are included as Stopwords. Since it is a question pairs, I will include the 6 W'sas meaningful words.

```
StopWord = stopwords("english") # import stopwords
for (i in 1:length(stopwords("english"))){
  if ( StopWord[i] %in% c("who","what","when","where","why","how","which") ){
    StopWord[i] = NA
  }
}
StopWord = as.character(na.exclude(StopWord))
```

Then we start decipher the question for each rows.

```
#Initialize data frame to store outputs in each iteration
Train = data.frame()
Train.row = data.frame()
## Iteration starts ##
# for (i in 1:nrow(train))
i = 1
q1 = Corpus(VectorSource(train$question1[i]))
q2 = Corpus(VectorSource(train$question2[i]))
```

Removing punctuations

```
q1 = tm_map(q1, removePunctuation)
q2 = tm_map(q2, removePunctuation)
```

Removing Numbers removeNumbers

```
q1 = tm_map(q1, removeNumbers)
q2 = tm_map(q2, removeNumbers)
```

To avoid duplicacy converting all to lower case tolower

```
q1 = tm_map(q1, tolower)
q2 = tm_map(q2, tolower)
```

Removing common word endings stemDocument

```
q1 = tm_map(q1, stemDocument)
q2 = tm_map(q2, stemDocument)
```

Remove additional white space

```
q1 = tm_map(q1, stripWhitespace)
q2 = tm_map(q2, stripWhitespace)
```

Removing Stop Words as they don't add any value

```
q1 = tm_map(q1, removeWords, StopWord)
q2 = tm_map(q2, removeWords, StopWord)
```

Convert documents into text documents

```
q1 = tm_map(q1, PlainTextDocument); q2 = tm_map(q2, PlainTextDocument)
Q1 = TermDocumentMatrix(q1)$dimnames$Terms
```

```
Q2 = TermDocumentMatrix(q2)$dimnames$Terms
Q1;Q2
```

```
## [1] "guid"   "india"  "invest" "market" "share"  "step"   "what"
```

```
## [1] "guid"   "invest" "market" "share"  "step"   "what"
```

To this point, we can see that 2 Corpus are transformed into 2 list of dataset. In the usual text mining process, we will forn a TDM (term document matrix) and count the frequency and table the words for each rows and calculate the distance. However, this cause the problem that the sparsity of the data and therefore it is reasonable to calculate the "distance" of only the pairs of question; or as to say, the **Similarity** of each question pairs. We calculate intuitively by the proportion of 2 list Q1 and Q2 matches.

```
  same_items = sum(Q1 %in% Q2)
  distinct_items = length(Q1) + length(Q2)
  #distinct_items
  match_count = (2*same_items)/(distinct_items)
  cat("Proportion of matched words in pairs : ", match_count)
```

```
## Proportion of matched words in pairs :  0.9230769
```

Now we form additional features by the package `syuzhet`. The function `get_nrc_sentiment` produce the table of words in a Corpus that belongs to certain categories of sentiment of data. For each category, we will calculate the proportion of the corresponding sentiments as an independent feature, and store the result into the data matrix.

```
  sentiment1 = get_nrc_sentiment(train$question1[i])
  sentiment2 = get_nrc_sentiment(train$question2[i])

  if(sum(sentiment1)!= 0 ) sentiment1 = sentiment1/sum(sentiment1)
  if(sum(sentiment2)!= 0 ) sentiment2 = sentiment2/sum(sentiment2)
```

```
  Q1_anger =  sum(sentiment1$anger)
  Q1_anticipation =  sum(sentiment1$anticipation)
  Q1_disgust =  sum(sentiment1$disgust)
  Q1_fear =  sum(sentiment1$fear)
  Q1_joy =  sum(sentiment1$joy)
  Q1_sadness =  sum(sentiment1$sadness)
  Q1_surprise =  sum(sentiment1$surprise)
  Q1_trust =  sum(sentiment1$trust)
  Q1_negative =  sum(sentiment1$negative)
  Q1_positive =  sum(sentiment1$positive)

  Q2_anger =  sum(sentiment2$anger)
  Q2_anticipation =  sum(sentiment2$anticipation)
  Q2_disgust =  sum(sentiment2$disgust)
  Q2_fear =  sum(sentiment2$fear)
  Q2_joy =  sum(sentiment2$joy)
  Q2_sadness =  sum(sentiment2$sadness)
  Q2_surprise =  sum(sentiment2$surprise)
  Q2_trust =  sum(sentiment2$trust)
  Q2_negative =  sum(sentiment2$negative)
  Q2_positive =  sum(sentiment2$positive)
  Train.row = cbind(match_count,
                Q1_anger,Q1_anticipation,Q1_disgust,Q1_fear,Q1_joy,
                Q1_sadness,Q1_surprise,Q1_trust,Q1_negative,Q1_positive,
                Q2_anger,Q2_anticipation,Q2_disgust,Q2_fear,Q2_joy,
                Q2_sadness,Q2_surprise,Q2_trust,Q2_negative,Q2_positive)
    Train = rbind.data.frame(Train,Train.row)
```

```
  i = i+1
##End iteration
```

After the iterations, we will get the table, this the first row of our output data in the first part of data engineering

match_count	Q1_anger	Q1_anticipation	Q1_disgust	Q1_fear	Q1_joy	Q1_sadness	Q1_surprise
0.92	0	0.17	0	0	0.17	0	0

Q1_trust	Q1_negative	Q1_positive	Q2_anger	Q2_anticipation	Q2_disgust	Q2_fear	Q2_joy
0.33	0	0.33	0	0.17	0	0	0.17

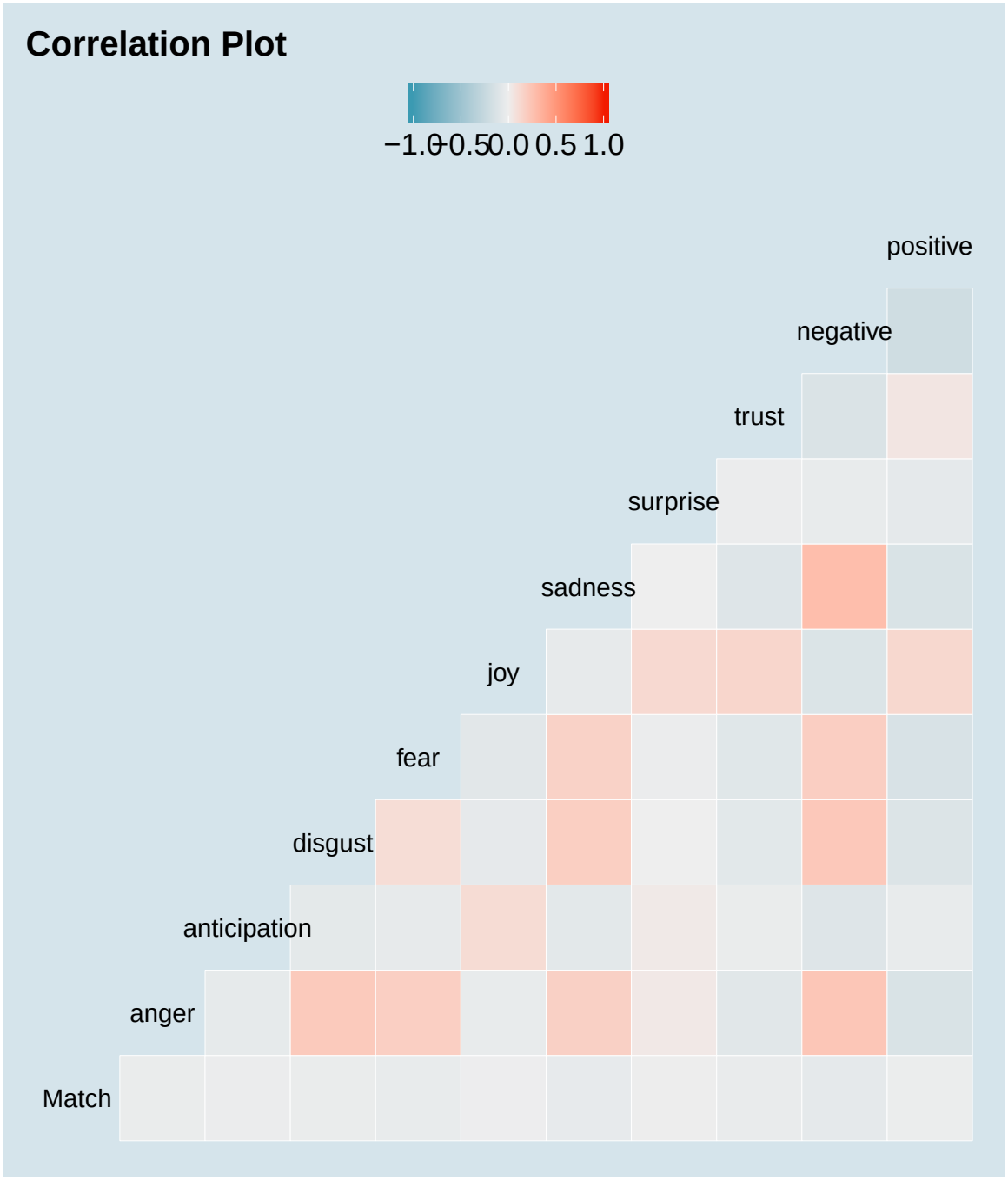
Q2_sadness	Q2_surprise	Q2_trust	Q2_negative	Q2_positive
0	0	0.33	0	0.33

Now, from the data, we observe the data's correlation

As we can see, the correlation are not high, and there are no connection between the 2 pairs of data in the sentiments feature, therefore, we would like to transform the feature pairs into one, selecting the intersection of the 2 pairs of questions. If there are same information in 2 questions, it is likely to have similar attributes in the corresponding sentiments. Therefore, we choose the minimum of each of the pairs of features. As an example, we pick the minimum of **Q1__joy** and **Q2__joy** and transform to a new feature **joy**.

```
sentiment = get_nrc_sentiment(iconv(train$question1[1], "latin1", "ASCII", sub=""))
#Compare and select the label that is intersected
T_compared = pmin(as.matrix(Training[,3:12]),as.matrix(Training[,3:12]))
#remake the data frame
colnames(T_compared) = colnames(sentiment)
DATA = cbind.data.frame(Match = Training$match_count,T_compared,Y = train$is_duplicate)
#Resample
set.seed(12)
DATA = DATA[sample(nrow(DATA), nrow(DATA)), ]
DATA[,1:11] = as.data.frame(DATA[,1:11])
```

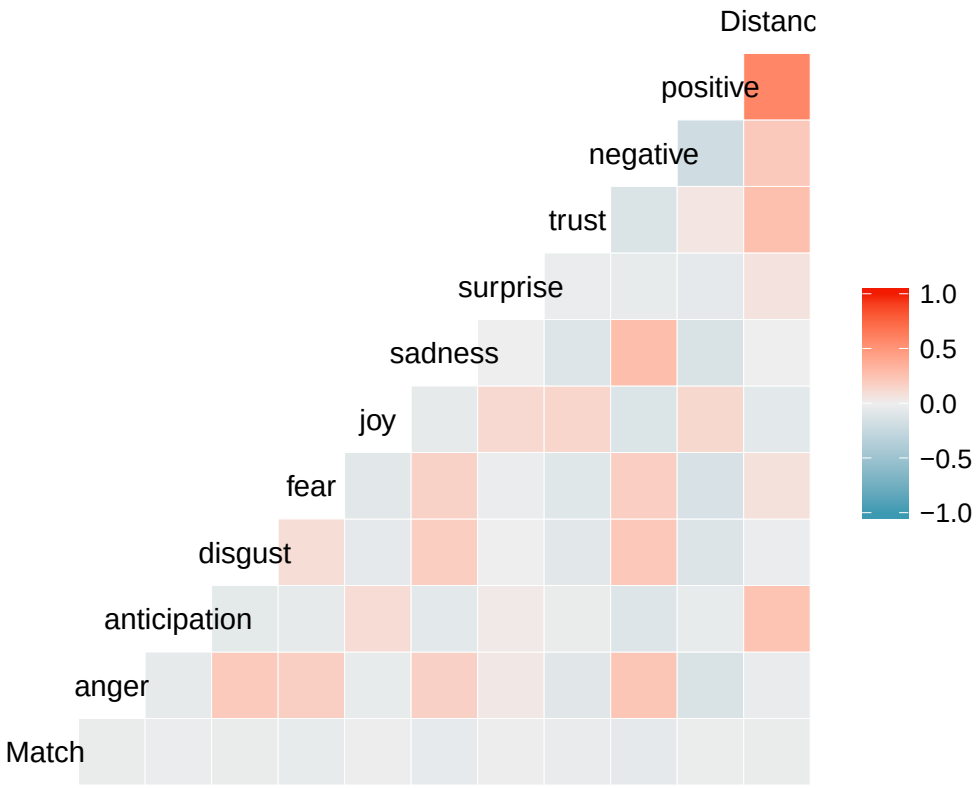
```
ggcorr(DATA[,-12]) + ggtitle("Correlation Plot") + theme_economist()
```



Now, we can assemble another feature by calculating the distance between the sentiment. In the previous combination of sentiment, it is approximately the feature that indicate the gap, or absolute distance in between the same sentiment category between the pairs of question.

$$\text{Distance} = \sqrt{\sum (Q_1 sentiment - Q_2 sentiment)^2} = \sqrt{\sum (sentiment)^2}$$

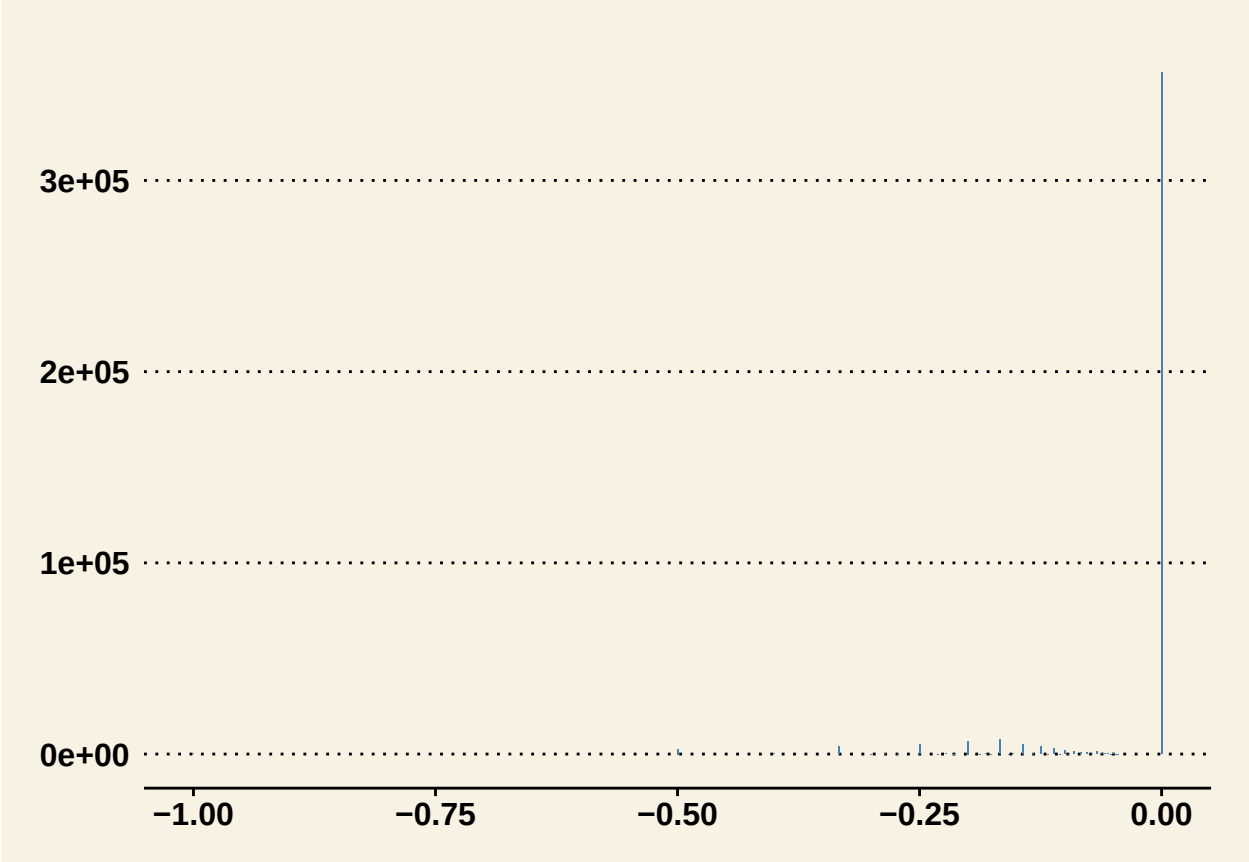
```
#Distance = matrix(matrix(0,nrow = nrow(DATA) ,ncol = 1))
#for(x in 1:nrow(DATA)){
#  Distance[x,1] =sum((DATA[x,2:11])^2)
#  print(i)
#}
#Distance = sqrt(Distance)
#write.csv( Distance,"C:\\Distance.csv")
Distance = read.csv("C:\\Distance.csv")
DATA = cbind.data.frame(DATA, Distance = Distance$V1)
ggcorr(DATA[,-12]) + ggtitle("Correlation Plot") + theme_pander()
```



In additional, we apply the negative sentiment with an additional negative sign.

	Match	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive	Y	Distance
28042	0.91	0	0	0.00	0	0	0	0	0.0	0.00	0.00	1	0.00
330618	0.33	0	0	0.00	0	0	0	0	0.0	0.00	0.00	0	0.00
381091	0.21	0	0	0.00	0	0	0	0	0.5	0.00	0.50	1	0.50
108908	0.40	0	0	-0.33	0	0	0	0	0.0	-0.33	0.33	0	0.33
68466	0.75	0	0	0.00	0	0	0	0	0.0	0.00	0.00	1	0.00
13704	0.73	0	0	0.00	0	0	0	0	0.0	0.00	1.00	1	1.00

Note that for better analysis, we take into consideration that spasity is significant in the sentiment category.



DATA MODELING AND PPREDICTION

Introduction to XGBoost

XGBoost stands for **E**xtreme **G**radient **Boost**ing. It is a supervised learning algorithm. It is a library for developing fast and high performance gradient boosting tree models. Parallel computation behind the scenes is what makes it this fast. It has been very popular in recent years due to its versatiltiy, scalability and efficiency. Like any other ML algorithm, this too has its own pros and cons. But fortunately pros easily outweigh cons given we have an astute understanding of the algorithm and an intuition for proper parameter tuning. This has been proven by its huge popularity on Kaggle.

In `DATA`, we cut the first 250000 rows as training set and the rest 154289 rows as testing dataset.

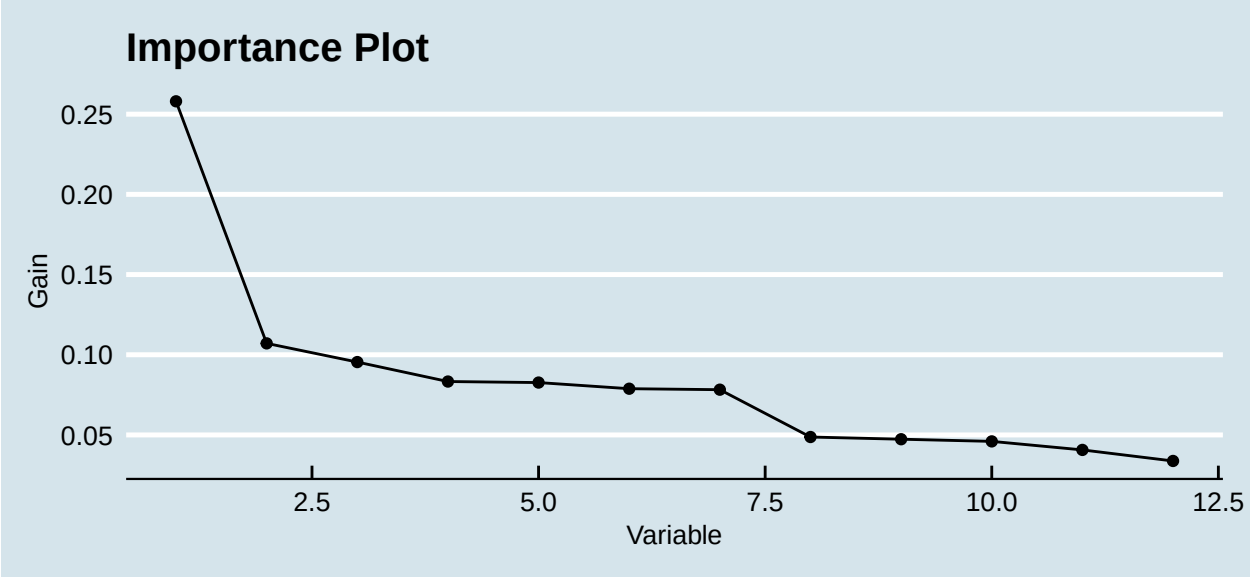
```
#Seperate into training and testing data
Label_train = c(data.matrix(DATA$Y[1:250000]))
Label_test = as.factor(DATA$Y[250001:404290])
X_train = DATA[1:250000,c(1:11,13)]
X_test = DATA[250001:404290,c(1:11,13)]
```

For the modeling, since it is a binary classification problem, I choose to blend 4 models : **XGboost**, **Logistic regression (glm)**, **CART**,and **C5.0**.

Training and validation of XGBoost

XGBoost

```
IMP = xgb.importance(colnames(DATA), model = XGBModel)
IMP = IMP[order(IMP$Cover,decreasing = TRUE),]
IMP = as.data.frame(IMP)
qplot(x = c(1:length(IMP$Gain)),y = IMP$Cover, xlab = "Variable", ylab = "Gain") +
ggtitle("Importance Plot") + geom_line() + theme_economist()
```




```
predict_XGB = predict(XGBModel, data.matrix(X_test))
prediction_XGB = as.numeric(predict_XGB > 0.5)
Table_XGB = table(prediction_XGB,Label_test)
Accruacy_XGB = sum(diag(Table_XGB))/sum(Table_XGB)
```

Logistic regression

```
GLMModel = glm(Label_train ~., family=binomial(link='logit'),data = X_train)
pred_GLM = predict(GLMModel, X_test)
prediction_GLM = as.numeric(pred_GLM > 0.5)
Accruacy_GLM = sum( prediction_GLM == Label_test ) / length( prediction_GLM )
```

CART classification tree

```
CRTModel = rpart(as.factor(Label_train)~.,data=X_train)
pred_CRT = predict(CRTModel, X_test, type = "class")
prediction_CRT = as.numeric(as.matrix(pred_CRT))
Accruacy_CRT = sum( prediction_CRT == Label_test ) / length( prediction_CRT )
```

C5.0 boosting tree

```
C50Model = C5.0(as.factor(Label_train)~ . ,data=X_train , trials=50)
prediction_C50 = predict(C50Model, X_test, type = "class")
Accruacy_C50 = sum( prediction_C50 == Label_test ) / length( prediction_C50 )
```

Ensembling models

Now after the training of the above models, we ensemble the model by voting the testing set and categorize it as 1 if more than 2 model picked as category 1.

```
Vote = as.numeric(prediction_XGB) +
  as.numeric(prediction_GLM) +
  as.numeric(prediction_CRT) +
  as.numeric(prediction_C50) -1
prediction_Vote = as.numeric(Vote > 2)
Ensembled_Accruacy = sum( prediction_Vote == Label_test ) / length(prediction_Vote)
```

This is the final result in compairson.

Ensembled_Accruacy	Accruacy_XGB	Accruacy_GLM	Accruacy_CRAT	Accruacy_C50
0.6906475	0.6998833	0.6575669	0.6700305	0.6837708

From the result, we can see that the perfoemance of XGBoost is the best of the 4 methods. The performance of the models are nearly as powerful, XGBoost conducted 2000 iterations (trees). taking much longer than the other 3 models but with only slightly better performance.

Reference

1. http://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
2. An Introduction to Statistical Learning with Applications in R.Hastie, Tibshirani and Friedman.
3. The Elements of Statistical Learning (2nd edition).Hastie, Tibshirani and Friedman .