Data Mining Final

응용통계학과 20152410 배형준

Data Mining I: Final Examination (Spring 2020)

Friday, June 19, 12:00 AM - Tuesday, June 23, 23:59 PM (100%)

Wednesday, June 24, 12:00 AM - 23:59 PM (70%)

Thursday, June 25, 12:00 AM - 23:59 PM (40%)

|  |
| --- |
| * **Upload a single pdf file including all your work for all problems once.** If you upload multiple files, then only the last uploaded file will be counted. For example, you upload a file on June 23 and upload another file on June 24, then the file submitted on June 23 will be ignored. * **You are fully responsible for uploading your work.** Try to upload your file as early as possible. There may be heavy internet traffic around the deadline. The above deadline is the time you finish uploading. It is not the time you start uploading. * **Do not email your exam.** * **Your pdf file must include this page with your name and CAU ID number.** * **Show all work to obtain full credits. Grading is based on what you write, not on what you think.** |

|  |  |
| --- | --- |
| (10 points) This is to certify that I have used no other person as a resource. In addition, I have read the above instructions. I fully understand the instructions and agree on the terms. | |
| Name: **Bae Hyungjun** | ID# : **20152410** |

|  |
| --- |
| Please choose Accuracy or AUC to calculate your exam score.  I want to use **AUC.** |

Use the attached “Train.csv” data to classify the binary response variable y. Follow the writing instructions posted in eClass. There are two more files that are posted in eClass. “Xtest.csv” file has the predictor values that you need to calculate their predicted classes. The other file “Ans.csv” has two columns: one is “yhat” and the second is “prob”. You fill the predicted classes and the posterior probabilities from your final model. Upload (1) your pdf file including all your work following the writing instructions and (2) your “Ans.csv” file that you fill two columns in the same ID order as “Xtest.csv” ID column. In addition, choose Accuracy or AUC to determine a part of your exam scores. Note that you are responsible for using the consistent coding and order as the original data when you fill the “Ans.csv” file.

|  |  |  |
| --- | --- | --- |
| grading items | points | policy |
| signed 1st page | 10 | Included as 1st page with name, ID, and your choice of Accuracy or AUC? |
| writing | 10 | Based on the writing instructions |
| methods and program | 10 | How many methods?  Are they properly applied to the problem?  Does the attached program run without error?  Does the program produce the same results in your report? etc. |
| Accuracy or AUC | 30 | [(Accuracy or AUC) - 85] X 2 based on your “Ans.csv” file.  If Accuracy or AUC is less than 85, then the score will be zero. |

Contents

[Main text 4](#_Toc43607193)

[Sequence of data analysis 4](#_Toc43607194)

[1. load and split dataset 4](#_Toc43607195)

[2. data modeling 5](#_Toc43607196)

[2-1. glmnet 5](#_Toc43607197)

[2-2. support vector machine with rbf kernel 6](#_Toc43607198)

[2-3. random forest 6](#_Toc43607199)

[2-4. xgboost 7](#_Toc43607200)

[3. evaluate the models 8](#_Toc43607201)

[3-1. glmnet 8](#_Toc43607202)

[3-2. support vector machine with rbf kernel 9](#_Toc43607203)

[3-3. random forest 10](#_Toc43607204)

[3-4. xgboost 11](#_Toc43607205)

[4. choose models to use or to make ensemble 12](#_Toc43607206)

[5. confime final model and predict class and posterior probability of Xtest 14](#_Toc43607207)

[Appendix : R code 15](#_Toc43607208)

[library package to use 15](#_Toc43607209)

[1. load and split dataset 16](#_Toc43607210)

[2. data modeling 17](#_Toc43607211)

[2-1. glmnet 17](#_Toc43607212)

[2-2. support vector machine with rbf kernel 18](#_Toc43607213)

[2-3. random forest 19](#_Toc43607214)

[2-4. xgboost 20](#_Toc43607215)

[3. evaluate the models 21](#_Toc43607216)

[3-1. glmnet 21](#_Toc43607217)

[3-2. support vector machine with rbf kernel 22](#_Toc43607218)

[3-3. random forest 23](#_Toc43607219)

[3-4. xgboost 24](#_Toc43607220)

[4. choose models to use or to make ensemble 25](#_Toc43607221)

[5. confime final model and predict class and posterior probability of Xtest 27](#_Toc43607222)

[5-1. glmnet 27](#_Toc43607223)

[5-2. support vector machine with rbf kernel 28](#_Toc43607224)

[5-3. random forest 29](#_Toc43607225)

[5-4. xgboost 30](#_Toc43607226)

[5-5. ensemble 31](#_Toc43607227)

# Main text

## Sequence of data analysis

1. load dataset, split dataset as train and validation using stratifed partition function
2. data modeling using caret to tune hyper parameter of candidate models
3. evaluate the models
4. choose models to use or to make ensemble
5. confime final model, predict class and posterior probability of Xtest

## 1. load and split dataset

read.csv 함수를 이용하여 train, test를 불러온 뒤 train을 다시 train : validation = 0.75 : 0.25비율로 train과 validation으로 나누었습니다. 나눌 때 sample을 이용하여 4000개의 데이터를 무작위로 나누지 않고, createDataPartition 함수를 이용하여 response variable의 0과 1의 비율을 유지하여 나눠주었습니다. 이는 train과 validation을 비슷한 상태로 만들어 validation을 이용해 train을 학습한 모델을 더 잘 평가하기 위함입니다. 아래의 코드 결과는 4000개의 trainset, 3001개의 train, 999개의 validation의 response variable의 0과 1의 비율입니다. 3개의 데이터셋 모두 0이 1보다 약 1.5배 많은 것을 확인할 수 있습니다.

## ratio of trainset target : 1.533249   
## ratio of train target : 1.532489   
## ratio of validation target : 1.535533

## 2. data modeling

3001개의 train을 학습하기 위한 모델 후보로 glmnet (elastic net), support vector machine with radial kernel, random forest, xgboost 4개의 모델을 선정하였습니다. 모델의 결과에 따라서 4개 중 가장 좋은 모델을 선택할지 또는 복수의 모델을 앙상블할지 결정하려고 합니다. 모델의 학습으로 caret 패키지의 train 함수를 이용하였고, cross validation의 fold는 4개, 튜닝 횟수는 250번, 하이퍼 파라미터 탐색은 ‘random’으로 동일하게 설정한 뒤 학습하였습니다. 평가 기준을 AUC로 선택했기 때문에 모델 평가 인수인 metric을 ‘ROC’로 설정해주었습니다.

### 2-1. glmnet

학습을 위해 elastic net를 위한 preProcess=c(‘center’, ‘scale’)로 설정해주어 표준화된 데이터를 학습할 수 있도록 설정하였습니다. 학습 결과 alpha=0.9571266, lambda=0.001491034를 최적 하이퍼 파라미터로 얻었습니다. alpha가 1에 가까운 것으로 보아 ridge보다는 lasso의 L1 penalized term이 모델 학습에 큰 영향을 끼쳤다고 볼 수 있습니다. 최적 하이퍼 파라미터일 때의 CV AUC는 0.9615987입니다. 선형모델의 CV AUC가 꽤 준수한 성능을 보여주기 때문에 이후에 사용할 복잡한 모델들이 더 좋은 성능을 보여줄 것이라 기대할 수 있었습니다.

## alpha lambda ROC

## 1 0.9571266 0.001491034 0.9615987

### 2-2. support vector machine with rbf kernel

rbf 커널을 이용한 support vector machine도 glmnet과 마찬가지로 preProcess=c(‘center’, ‘scale’)로 설정해주어 변수의 단위에 따른 영향력을 같게 만들어주었습니다. 학습 결과 sigma=0.009966466, C=21.71236을 최적 하이퍼 파라미터로 얻었습니다. C 값이 큰 것으로 보아 모델이 오분류된 데이터에 대한 가중치를 높게 책정한 것을 확인할 수 있습니다. 최적 하이퍼 파라미터일때의 CV AUC는 0.9678749입니다. 이는 glmnet의 CV AUC보다 0.0062762보다 크므로 더 좋은 분류기라고 평가할 수 있겠으나 그 차이가 미비하여 성능이 크게 차이 난다고 볼 수 없다고 생각합니다.

## sigma C ROC

## 1 0.009966466 21.71236 0.9678749

### 2-3. random forest

method=’rf’보다 빠르게 학습할 수 있는 method=’ranger’를 이용하여 random forest classifier를 학습하였습니다. 최적 하이퍼 파라미터로 min.node.size=4, mtry=4, splitrule=’gini’를 얻었고 그때의 CV AUC는 0.977648입니다. 앞선 두 모델보다 성능이 살짝 더 향상된 것을 확인할 수 있습니다. splitrule이 extratrees가 아닌 것으로 보아 tree의 절단점을 무작위로 자르는 것보단 어느 정도 절단점을 탐색한 후 최적 지점을 선정하는 것이 이 데이터에 대해선 좋은 방법이란 것을 확인할 수 있습니다.

## min.node.size mtry splitrule ROC

## 1 4 4 gini 0.977648

### 2-4. xgboost

method=’xgbTree’를 이용하여 base learner가 tree인 xgboost classifier를 학습하였습니다. 아래의 코드 결과와 같이 7개의 최적 하이퍼 파라미터를 얻었습니다. 그때의 CV AUC는 0.9800166으로, 앞서 학습했던 glmnet, svmradial, random forest보다 더 좋은 성능을 보여줍니다.

이제 4개의 모델이 학습에 사용되지 않았던 validation에 대해 어느 정도의 성능을 가지는지 확인해보도록 하겠습니다.

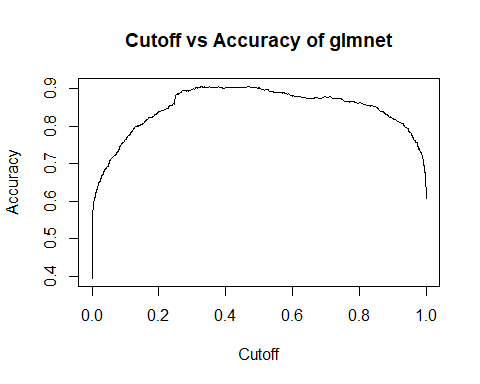
## eta max\_depth gamma colsample\_bytree min\_child\_weight subsample  
## 1 0.09257654 9 1.453736 0.3281601 0 0.9524782  
## nrounds ROC

## 1 338 0.9800166

## 3. evaluate the models

Validation에 대한 AUC, maximum Accuracy와 그때의 Cutoff 값을 이용하여 4개의 모델을 평가하려 합니다. 또한 Cutoff vs Accuracy 그래프를 확인하여, 만약 앙상블을 할 경우 어떻게 cutoff를 결정해야 하는지 생각해보도록 하겠습니다.

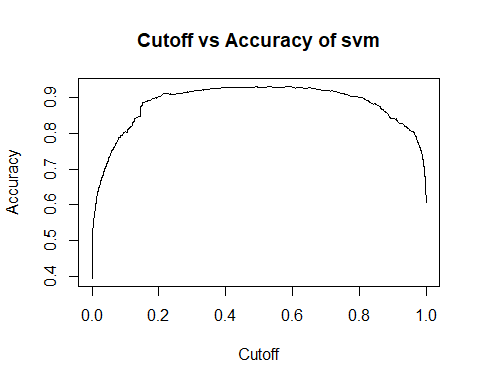
### 3-1. glmnet



Validation AUC는 0.9628162, cutoff가 0.326344일 때 Accuracy가 0.9059059로 최댓값을 가집니다. 위의 그래프를 보면 Accuracy가 cutoff에 살짝 민감하게 반응하는 것처럼 보입니다. AUC는 꽤 높은 편이라고 생각하지만 Accuracy가 약 90.6%로 만족스럽지 않은 결과입니다.

## AUC of glmnet : 0.9628162   
## Max Accuracy of glmnet : 0.9059059   
## Cutoff of maximum accuracy of glmnet : 0.326344

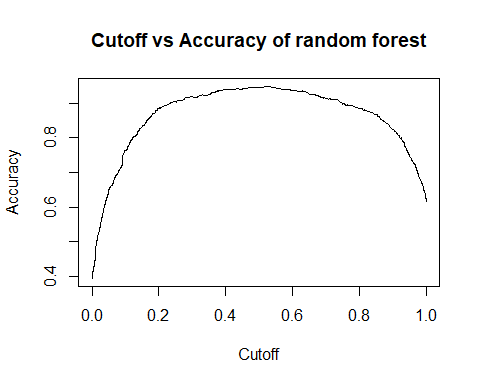
### 3-2. support vector machine with rbf kernel



Validation AUC는 0.9730314, cutoff가 0.5292952일 때 Accuracy가 0.9319319로 최댓값을 가집니다. 위의 그래프를 보면 cutoff에 대한 Accuracy의 민감함이 glmnet보다 덜 민감한 것처럼 보입니다. AUC와 Accuracy가 glmnet보다 높은 수치를 기록하여, 성능이 더 좋은 모델이라고 평가할 수 있습니다.

## AUC of svm : 0.9730314   
## Max Accuracy of svm : 0.9319319   
## Cutoff of maximum accuracy of svm : 0.5292952

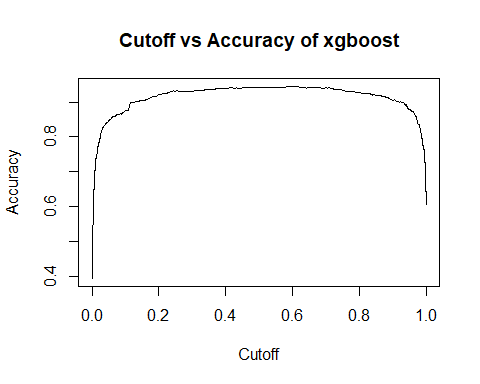
### 3-3. random forest



Validation AUC는 0.9830914, cutoff가 0.5272259일 때 Accuracy가 0.9479479로 최댓값을 가집니다. 4개의 모델 중 가장 높은 Accuracy를 가지지만 AUC는 xgboost보다 살짝 낮은 성능을 보여줍니다. 즉 Accuracy가 기준일 땐 random forest가 가장 좋은 모델로 평가할 수 있지만 AUC 기준일 땐 2번째로 좋은 모델이라고 평가할 수 있습니다다.

## AUC of rf : 0.9830914   
## Max Accuracy of rf : 0.9479479   
## Cutoff of maximum accuracy of rf : 0.5272259

### 3-4. xgboost

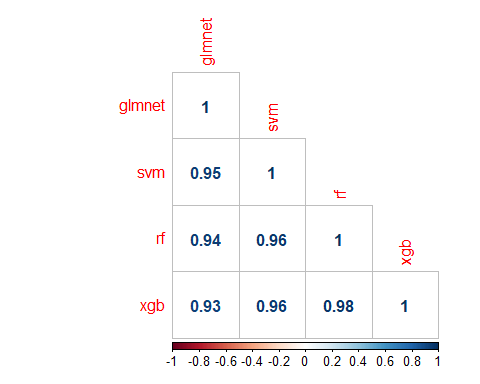


Validation AUC는 0.9832278, cutoff가 0.5992255일 때 Accuracy가 0.9449449로 최댓값을 가집니다. 4개의 모델 중 가장 높은 AUC를 가지지만 Accuracy는 random forest보다 살짝 낮은 성능을 보여줍니다. 즉 AUC가 기준일 땐 xgboost가 가장 좋은 모델로 평가할 수 있지만 Accuracy가 기준일 땐 2번째로 좋은 모델이라고 평가할 수 있습니다.

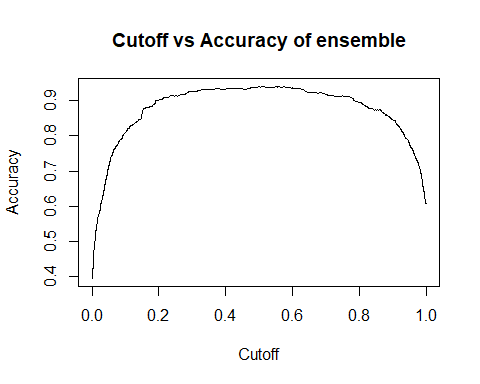
## AUC of xgb : 0.9832278   
## Max Accuracy of xgb : 0.9449449   
## Cutoff of maximum accuracy of xgb : 0.5992255

기준 통계량이 AUC인지 Accuracy인지에 따라 가장 좋은 모델이 달라지므로, 일반화가 더 잘되는 모델을 구하기 위해선 위에서 구한 4개의 모델을 앙상블하는 방법을 고려해볼 수 있다고 생각합니다.

## 4. choose models to use or to make ensemble



4개의 validation posterior probability들이 강한 상관성을 가지고 있어서 앙상블 했을 때 성능이 훨씬 더 좋아지지는 않을 것으로 추정됩니다. 하지만 앞서 언급했듯이 기준 통계량에 따라 가장 좋은 모델이 바뀌므로 4개의 모델을 앙상블하여 최종 예측값을 구하려고 합니다.



앙상블을 했을 때 validation AUC는 0.9814973, cutoff가 0.516017일 때 maximum Accuracy는 0.9409490입니다. 개별 모델들과 비교했을 때 random forest, xgboost보다 AUC가 조금 낮아졌지만 대신 결과를 합했기 때문에 더 견고한 모델을 얻었다고 평가할 수 있습니다. 이 모델을 최종 모델로 사용하기로 했고 test class를 예측할 때의 cutoff는 ensemble의 cutoff인 0.516017로 사용하기로 결정했습니다.

## AUC of ensemble : 0.9814973   
## Max Accuracy of ensemble : 0.9409409   
## Cutoff of maximum accuracy of ensemble : 0.516017

## AUC Accuracy Cutoff  
## glmnet 0.9628162 0.9059059 0.3263440  
## svm 0.9730314 0.9319319 0.5292952  
## rf 0.9830914 0.9479479 0.5272259  
## xgb 0.9832278 0.9449449 0.5992255  
## ensemble 0.9814973 0.9409409 0.5160170

## 5. confime final model and predict class and posterior probability of Xtest

4000개의 train 데이터 전부를 이용하여 glmnet, svmradial, random forest, xgboost를 학습했습니다. 그때의 하이퍼 파라미터는 3001개의 데이터만 사용해서 구한 최적 하이퍼 파라미터를 사용하였습니다. Test 데이터의 class ‘1’에 대한 4개의 posterior probability를 평균 내어 최종 ensemble posterior probability를 구할 수 있었습니다. cutoff는 validation ensemble accuracy를 최대로 하는 0.516017를 사용하였고, 1일 확률이 cutoff보다 크면 1로 예측하고 cutoff보다 작으면 0으로 예측하여 최종 predicted test class를 얻을 수 있었습니다.

# Appendix : R code

## library package to use

library(caret)

library(dplyr)

library(ROCR) # for auc, acc, cutoff

library(glmnet)

library(kernlab) # for support vector machine

library(ranger) # for random forest as ranger

library(xgboost) # for xgboost

library(writexl) # for exporting answer

library(ggplot2)  
library(corrplot)

## 1. load and split dataset

load dataset, split dataset as train and validation using stratifed partition based on target variable

trainset = read.csv('./Train.csv', header=TRUE)  
testset = read.csv('./Xtest.csv', header=TRUE)  
  
train\_X = trainset[, 2:51]  
train\_Y = as.factor(trainset[, 52])  
test\_X = testset[, 2:51]  
  
  
  
  
student = 20152410  
train\_size = 0.75  
set.seed(student)  
train\_index = createDataPartition(train\_Y, p=train\_size)  
  
X\_train = train\_X[train\_index$Resample1, ]  
Y\_train = train\_Y[train\_index$Resample1]  
X\_val = train\_X[-train\_index$Resample1, ]  
Y\_val = train\_Y[-train\_index$Resample1]  
  
factor\_Y\_train = ifelse(Y\_train == '1', 'yes', 'no')  
  
# check wheter data is stratifed based on target  
  
cat(' ratio of trainset target :', summary(train\_Y)[1] / summary(train\_Y)[2],  
 '\n ratio of train target :', summary(Y\_train)[1] / summary(Y\_train)[2],  
 '\n ratio of validation target :', summary(Y\_val)[1] / summary(Y\_val)[2])

## ratio of trainset target : 1.533249   
## ratio of train target : 1.532489   
## ratio of validation target : 1.535533

## 2. data modeling

data modeling using caret to tune hyper parameter of candidate models

candidate : glmnet, svmradial, random forest, xgboost

### 2-1. glmnet

glmnet\_tune\_length = 250  
glmnet\_fold\_number = 4  
glmnet\_train\_control = trainControl(method='cv',  
 number=glmnet\_fold\_number,  
 search='random',  
 classProbs = TRUE,  
 summaryFunction=twoClassSummary)  
  
glmnet\_start = Sys.time()  
  
set.seed(student)  
glmnet\_model = train(X\_train,  
 factor\_Y\_train,  
 method='glmnet',  
 trControl=glmnet\_train\_control,  
 metric='ROC',  
 tuneLength=glmnet\_tune\_length,  
 preProcess = c('center', 'scale'))  
  
glmnet\_train\_time = Sys.time() - glmnet\_start  
cat('Train time of glmnet model : '); glmnet\_train\_time

## Train time of glmnet model :

## Time difference of 13.29373 mins

best\_alpha = as.numeric(glmnet\_model$bestTune[1])  
best\_lambda = as.numeric(glmnet\_model$bestTune[2])  
  
glmnet\_pred = predict(glmnet\_model, newdata=X\_val, type='prob')  
  
glmnet\_best\_tune = glmnet\_model$results %>%   
 arrange(desc(ROC)) %>%   
 head(1)  
glmnet\_best\_tune

## alpha lambda ROC Sens Spec ROCSD SensSD  
## 1 0.9571266 0.001491034 0.9615987 0.9449339 0.861586 0.009071119 0.02112701  
## SpecSD  
## 1 0.02058914

### 2-2. support vector machine with rbf kernel

svm\_tune\_length = 250  
svm\_fold\_number = 4  
svm\_train\_control = trainControl(method='cv',  
 number=svm\_fold\_number,  
 search='random',  
 classProbs = TRUE,  
 summaryFunction=twoClassSummary)  
  
svm\_start = Sys.time()  
  
set.seed(student)  
svm\_model = train(X\_train,  
 factor\_Y\_train,  
 method='svmRadial',  
 trControl=svm\_train\_control,  
 metric='ROC',  
 tuneLength=svm\_tune\_length,  
 preProcess = c('center', 'scale'))  
  
svm\_train\_time = Sys.time() - svm\_start  
cat('Train time of svm model : '); svm\_train\_time

## Train time of svm model :

## Time difference of 38.41799 mins

best\_sigma = as.numeric(svm\_model$bestTune[1])  
best\_C = as.numeric(svm\_model$bestTune[2])  
  
svm\_pred = predict(svm\_model, newdata=X\_val, type='prob')  
  
svm\_best\_tune = svm\_model$results %>%   
 arrange(desc(ROC)) %>%   
 head(1)  
svm\_best\_tune

## sigma C ROC Sens Spec ROCSD SensSD  
## 1 0.009966466 21.71236 0.9678749 0.9526432 0.8801557 0.003811696 0.0137555  
## SpecSD  
## 1 0.01510196

### 2-3. random forest

rf\_tune\_length = 250  
rf\_fold\_number = 4  
rf\_train\_control = trainControl(method='cv',  
 number=rf\_fold\_number,  
 search='random',  
 classProbs=TRUE,  
 summaryFunction=twoClassSummary)  
  
rf\_start = Sys.time()  
  
set.seed(student)  
rf\_model = train(X\_train,  
 factor\_Y\_train,  
 method='ranger',  
 trControl=rf\_train\_control,  
 metric='ROC',  
 tuneLength=rf\_tune\_length)  
  
rf\_train\_time = Sys.time() - rf\_start  
cat('Train time of random forest model : '); rf\_train\_time

## Train time of random forest model :

## Time difference of 41.18659 mins

best\_mtry = as.numeric(rf\_model$bestTune[1])  
best\_splitrule = as.character(rf\_model$bestTune[2][1, 1])  
best\_min.node.size = as.numeric(rf\_model$bestTune[3])  
  
rf\_pred = predict(rf\_model, newdata=X\_val, type='prob')  
  
rf\_best\_tune = rf\_model$results %>%   
 arrange(desc(ROC)) %>%   
 head(1)  
rf\_best\_tune

## min.node.size mtry splitrule ROC Sens Spec ROCSD  
## 1 4 4 gini 0.977648 0.9609031 0.8919687 0.00258328  
## SensSD SpecSD  
## 1 0.0136226 0.01978735

### 2-4. xgboost

xgb\_tune\_length = 250  
xgb\_fold\_number = 4  
xgb\_train\_control = trainControl(method='cv',  
 number=xgb\_fold\_number,  
 search='random',  
 classProbs=TRUE,  
 summaryFunction=twoClassSummary)  
  
xgb\_start = Sys.time()  
  
set.seed(student)  
xgb\_model = train(X\_train,  
 factor\_Y\_train,  
 method='xgbTree',  
 trControl=xgb\_train\_control,  
 metric='ROC',  
 tuneLength=xgb\_tune\_length)  
  
xgb\_train\_time = Sys.time() - xgb\_start  
cat('Train time of xgboost model : '); xgb\_train\_time

## Train time of xgboost model :

## Time difference of 37.58226 mins

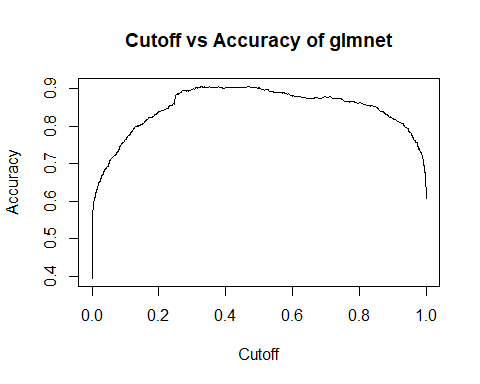
best\_nrounds = as.numeric(xgb\_model$bestTune[1])  
best\_max\_depth = as.numeric(xgb\_model$bestTune[2])  
best\_eta = as.numeric(xgb\_model$bestTune[3])  
best\_gamma = as.numeric(xgb\_model$bestTune[4])  
best\_colsample\_bytree = as.numeric(xgb\_model$bestTune[5])  
best\_min\_child\_weight = as.numeric(xgb\_model$bestTune[6])  
best\_subsample = as.numeric(xgb\_model$bestTune[7])  
  
xgb\_pred = predict(xgb\_model, newdata=X\_val, type='prob')  
  
xgb\_best\_tune = xgb\_model$results %>%   
 arrange(desc(ROC)) %>%   
 head(1)  
xgb\_best\_tune

## eta max\_depth gamma colsample\_bytree min\_child\_weight subsample  
## 1 0.09257654 9 1.453736 0.3281601 0 0.9524782  
## nrounds ROC Sens Spec ROCSD SensSD SpecSD  
## 1 338 0.9800166 0.9548458 0.902937 0.005578064 0.014218 0.01374313

## 3. evaluate the models

### 3-1. glmnet

glmnet\_prediction = prediction(glmnet\_pred['yes'], Y\_val)  
glmnet\_performance\_auc = performance(glmnet\_prediction, 'auc', 'cutoff')  
glmnet\_performance\_acc = performance(glmnet\_prediction, 'acc', 'cutoff')  
plot(glmnet\_performance\_acc, main='Cutoff vs Accuracy of glmnet')

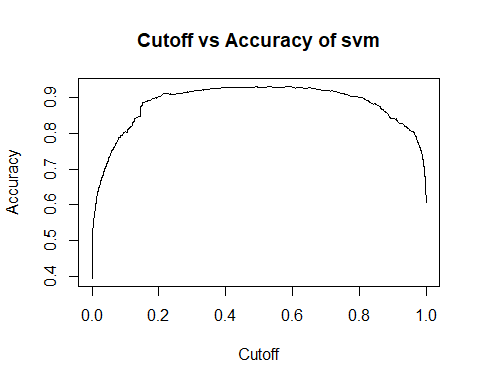


glmnet\_auc = glmnet\_performance\_auc@y.values[[1]]  
glmnet\_acc = max(glmnet\_performance\_acc@y.values[[1]])  
glmnet\_cutoff = glmnet\_performance\_acc@x.values[[1]][which.max(glmnet\_performance\_acc@y.values[[1]])]  
  
cat(' AUC of glmnet :', glmnet\_auc,  
 '\n Max Accuracy of glmnet :', glmnet\_acc,  
 '\n Cutoff of maximum accuracy of glmnet :', glmnet\_cutoff)

## AUC of glmnet : 0.9628162   
## Max Accuracy of glmnet : 0.9059059   
## Cutoff of maximum accuracy of glmnet : 0.326344

### 3-2. support vector machine with rbf kernel

svm\_prediction = prediction(svm\_pred['yes'], Y\_val)  
svm\_performance\_auc = performance(svm\_prediction, 'auc', 'cutoff')  
svm\_performance\_acc = performance(svm\_prediction, 'acc', 'cutoff')  
plot(svm\_performance\_acc, main='Cutoff vs Accuracy of svm')

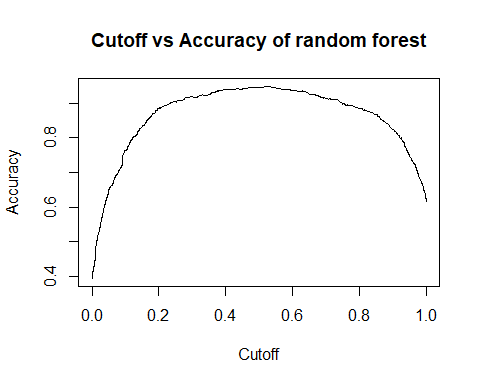


svm\_auc = svm\_performance\_auc@y.values[[1]]  
svm\_acc = max(svm\_performance\_acc@y.values[[1]])  
svm\_cutoff = svm\_performance\_acc@x.values[[1]][which.max(svm\_performance\_acc@y.values[[1]])]  
  
cat(' AUC of svm :', svm\_auc,  
 '\n Max Accuracy of svm :', svm\_acc,  
 '\n Cutoff of maximum accuracy of svm :', svm\_cutoff)

## AUC of svm : 0.9730314   
## Max Accuracy of svm : 0.9319319   
## Cutoff of maximum accuracy of svm : 0.5292952

### 3-3. random forest

rf\_prediction = prediction(rf\_pred['yes'], Y\_val)  
rf\_performance\_auc = performance(rf\_prediction, 'auc', 'cutoff')  
rf\_performance\_acc = performance(rf\_prediction, 'acc', 'cutoff')  
plot(rf\_performance\_acc, main='Cutoff vs Accuracy of random forest')

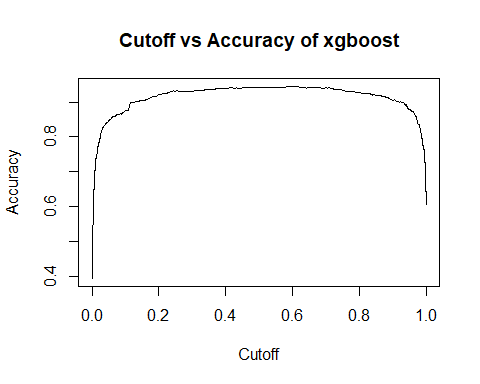


rf\_auc = rf\_performance\_auc@y.values[[1]]  
rf\_acc = max(rf\_performance\_acc@y.values[[1]])  
rf\_cutoff = rf\_performance\_acc@x.values[[1]][which.max(rf\_performance\_acc@y.values[[1]])]  
  
cat(' AUC of rf :', rf\_auc,  
 '\n Max Accuracy of rf :', rf\_acc,  
 '\n Cutoff of maximum accuracy of rf :', rf\_cutoff)

## AUC of rf : 0.9830914   
## Max Accuracy of rf : 0.9479479   
## Cutoff of maximum accuracy of rf : 0.5272259

### 3-4. xgboost

xgb\_prediction = prediction(xgb\_pred['yes'], Y\_val)  
xgb\_performance\_auc = performance(xgb\_prediction, 'auc', 'cutoff')  
xgb\_performance\_acc = performance(xgb\_prediction, 'acc', 'cutoff')  
plot(xgb\_performance\_acc, main='Cutoff vs Accuracy of xgboost')

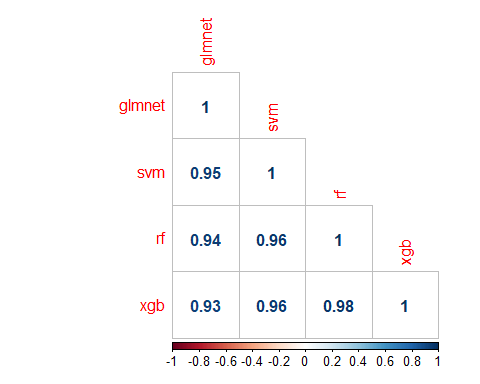


xgb\_auc = xgb\_performance\_auc@y.values[[1]]  
xgb\_acc = max(xgb\_performance\_acc@y.values[[1]])  
xgb\_cutoff = xgb\_performance\_acc@x.values[[1]][which.max(xgb\_performance\_acc@y.values[[1]])]  
  
cat(' AUC of xgb :', xgb\_auc,  
 '\n Max Accuracy of xgb :', xgb\_acc,  
 '\n Cutoff of maximum accuracy of xgb :', xgb\_cutoff)

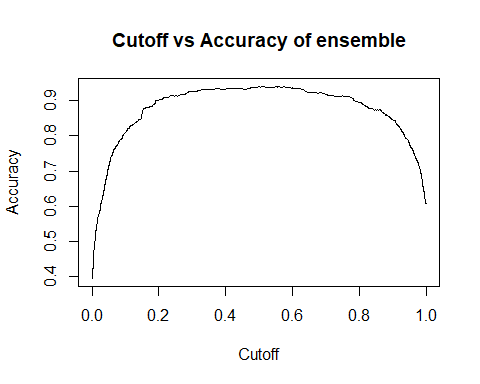
## AUC of xgb : 0.9832278   
## Max Accuracy of xgb : 0.9449449   
## Cutoff of maximum accuracy of xgb : 0.5992255

## 4. choose models to use or to make ensemble

validation\_pred = data.frame(glmnet\_pred['yes'],  
 svm\_pred['yes'],  
 rf\_pred['yes'],  
 xgb\_pred['yes'])  
colnames(validation\_pred) = c('glmnet', 'svm', 'rf', 'xgb')  
  
corr\_matrix = cor(validation\_pred)  
corrplot(corr\_matrix, method='number', type='lower')



validation\_pred = validation\_pred %>%   
 mutate(ensemble = (glmnet + svm + rf + xgb) / 4)  
  
ensemble\_prediction = prediction(validation\_pred$ensemble, Y\_val)  
ensemble\_performance\_auc = performance(ensemble\_prediction, 'auc', 'cutoff')  
ensemble\_performance\_acc = performance(ensemble\_prediction, 'acc', 'cutoff')  
plot(ensemble\_performance\_acc, main='Cutoff vs Accuracy of ensemble')



ensemble\_auc = ensemble\_performance\_auc@y.values[[1]]  
ensemble\_acc = max(ensemble\_performance\_acc@y.values[[1]])  
ensemble\_cutoff = ensemble\_performance\_acc@x.values[[1]][which.max(ensemble\_performance\_acc@y.values[[1]])]  
  
cat(' AUC of ensemble :', ensemble\_auc,  
 '\n Max Accuracy of ensemble :', ensemble\_acc,  
 '\n Cutoff of maximum accuracy of ensemble :', ensemble\_cutoff)

## AUC of ensemble : 0.9814973   
## Max Accuracy of ensemble : 0.9409409   
## Cutoff of maximum accuracy of ensemble : 0.516017

modeling\_result = data.frame(AUC = c(glmnet\_auc, svm\_auc, rf\_auc, xgb\_auc, ensemble\_auc),  
 Accuracy = c(glmnet\_acc, svm\_acc, rf\_acc, xgb\_acc, ensemble\_acc),  
 Cutoff = c(glmnet\_cutoff, svm\_cutoff, rf\_cutoff, xgb\_cutoff, ensemble\_cutoff))  
rownames(modeling\_result) = c('glmnet', 'svm', 'rf', 'xgb', 'ensemble')  
modeling\_result

## AUC Accuracy Cutoff  
## glmnet 0.9628162 0.9059059 0.3263440  
## svm 0.9730314 0.9319319 0.5292952  
## rf 0.9830914 0.9479479 0.5272259  
## xgb 0.9832278 0.9449449 0.5992255  
## ensemble 0.9814973 0.9409409 0.5160170

## 5. confime final model and predict class and posterior probability of Xtest

### 5-1. glmnet

final\_glmnet\_model = glmnet(x=as.matrix(train\_X),  
 y=train\_Y,  
 family='binomial',  
 alpha=best\_alpha,  
 lambda=best\_lambda,  
 standardize = TRUE)  
  
final\_glmnet\_pred = predict(final\_glmnet\_model, newx=as.matrix(test\_X), type='response')[, 1]  
  
final\_glmnet\_model

##   
## Call: glmnet(x = as.matrix(train\_X), y = train\_Y, family = "binomial", alpha = best\_alpha, lambda = best\_lambda, standardize = TRUE)   
##   
## Df %Dev Lambda  
## [1,] 47 0.6367 0.001491

### 5-2. support vector machine with rbf kernel

final\_svm\_model = ksvm(train\_Y ~ .,  
 data=cbind(train\_X, train\_Y),  
 scaled=TRUE,  
 type='C-svc',  
 kernel='rbfdot',  
 kpar=list(sigma=best\_sigma),  
 C=best\_C,  
 prob.model=TRUE)  
  
final\_svm\_pred = predict(final\_svm\_model, newdata=test\_X, type="probabilities")[, 2]  
  
final\_svm\_model

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 21.7123636978789   
##   
## Gaussian Radial Basis kernel function.   
## Hyperparameter : sigma = 0.00996646616396962   
##   
## Number of Support Vectors : 896   
##   
## Objective Function Value : -12279.39   
## Training error : 0.04625   
## Probability model included.

### 5-3. random forest

final\_rf\_model = ranger(train\_Y ~ .,  
 data=cbind(train\_X, train\_Y),  
 mtry=best\_mtry,  
 splitrule=best\_splitrule,  
 min.node.size=best\_min.node.size,  
 probability=TRUE)  
  
final\_rf\_pred\_ = predict(final\_rf\_model, data=test\_X, type="response",  
 num.trees=final\_rf\_model$num.trees)  
final\_rf\_pred = final\_rf\_pred\_$predictions[, 2]  
  
final\_rf\_model

## Ranger result  
##   
## Call:  
## ranger(train\_Y ~ ., data = cbind(train\_X, train\_Y), mtry = best\_mtry, splitrule = best\_splitrule, min.node.size = best\_min.node.size, probability = TRUE)   
##   
## Type: Probability estimation   
## Number of trees: 500   
## Sample size: 4000   
## Number of independent variables: 50   
## Mtry: 4   
## Target node size: 4   
## Variable importance mode: none   
## Splitrule: gini   
## OOB prediction error (Brier s.): 0.05036759

### 5-4. xgboost

temp\_train\_Y = ifelse(train\_Y == '1', 1, 0)  
final\_xgb\_model = xgboost(data=as.matrix(train\_X),  
 label=temp\_train\_Y,  
 objective='binary:logistic',  
 nrounds=best\_nrounds,  
 max\_depth=best\_max\_depth,  
 eta=best\_eta,  
 gamma=best\_gamma,  
 colsample\_bytree=best\_colsample\_bytree,  
 min\_child\_weight=best\_min\_child\_weight,  
 subsample=best\_subsample,  
 verbose=FALSE)  
  
final\_xgb\_pred = predict(final\_xgb\_model, newdata=as.matrix(test\_X), type="prob")  
  
final\_xgb\_model

## ##### xgb.Booster  
## raw: 1.3 Mb   
## call:  
## xgb.train(params = params, data = dtrain, nrounds = nrounds,   
## watchlist = watchlist, verbose = verbose, print\_every\_n = print\_every\_n,   
## early\_stopping\_rounds = early\_stopping\_rounds, maximize = maximize,   
## save\_period = save\_period, save\_name = save\_name, xgb\_model = xgb\_model,   
## callbacks = callbacks, objective = "binary:logistic", max\_depth = ..2,   
## eta = ..3, gamma = ..4, colsample\_bytree = ..5, min\_child\_weight = ..6,   
## subsample = ..7)  
## params (as set within xgb.train):  
## objective = "binary:logistic", max\_depth = "9", eta = "0.0925765427979641", gamma = "1.45373629638925", colsample\_bytree = "0.328160088695586", min\_child\_weight = "0", subsample = "0.95247815316543", silent = "1"  
## xgb.attributes:  
## niter  
## callbacks:  
## cb.evaluation.log()  
## # of features: 50   
## niter: 338  
## nfeatures : 50   
## evaluation\_log:  
## iter train\_error  
## 1 0.11950  
## 2 0.08475  
## ---   
## 337 0.01750  
## 338 0.01750

### 5-5. ensemble

final\_ensemble\_pred = (final\_glmnet\_pred + final\_svm\_pred + final\_rf\_pred + final\_xgb\_pred) / 4  
final\_ensemble\_class = ifelse(final\_ensemble\_pred >= ensemble\_cutoff, '1', '0')  
  
final\_ensemble\_ = data.frame(final\_ensemble\_class, final\_ensemble\_pred)  
final\_ensemble = cbind(rownames(final\_ensemble\_), final\_ensemble\_)  
colnames(final\_ensemble) = c('ID', 'yhat', 'prob')  
  
head(final\_ensemble)

## ID yhat prob  
## 1 1 1 0.99590291  
## 2 2 0 0.28749520  
## 3 3 0 0.01066789  
## 4 4 1 0.61025069  
## 5 5 1 0.94874179  
## 6 6 1 0.75386461

write\_xlsx(final\_ensemble, path='./final\_answer.xlsx')