Modeling_ind_0.R

yaxin

2021-04-12

```
# Uncomment if packages not installed
## install.packages("psych")
## install.packages("caret")
## install.packages("randomForest")
## install.packages("MLmetrics")
## install.packages("doParallel")
## install.packages("kernlab")
## install.packages("glmnet")



# Load data
setwd('D:\\Yaxin\\HKBU BM\\Courses\\Sem 2\\ECON7860 Big Data Analytics for Business (S11)\\Group Projec
rawData <- read.csv2("HR_comma_sep.csv", sep = ',')



# Transform feature types
transform_feature <- function(X) {
  X$satisfaction_level <- as.numeric(X$satisfaction_level)
  X$last_evaluation <- as.numeric(X$last_evaluation)
  X$Work_accident <- as.factor(X$Work_accident)
  X$promotion_last_5years <- as.factor(X$promotion_last_5years)
  X$sales <- as.factor(X$sales)
  X$salary <- as.factor(X$salary)
  X$left <- factor(ifelse(X$left == 0, 'no', 'yes'), levels = c('yes', 'no'))
  return(X)
}

rawData <- transform_feature(rawData)
summary(rawData)
```

```
##  satisfaction_level last_evaluation  number_project  average_montly_hours
##  Min.   :0.0900     Min.   :0.3600   Min.   :2.000   Min.   : 96.0
##  1st Qu.:0.4400     1st Qu.:0.5600   1st Qu.:3.000   1st Qu.:156.0
##  Median :0.6400     Median :0.7200   Median :4.000   Median :200.0
##  Mean   :0.6128     Mean   :0.7161   Mean   :3.803   Mean   :201.1
##  3rd Qu.:0.8200     3rd Qu.:0.8700   3rd Qu.:5.000   3rd Qu.:245.0
##  Max.   :1.0000     Max.   :1.0000   Max.   :7.000   Max.   :310.0
##
##  time_spend_company Work_accident  left       promotion_last_5years
##  Min.   : 2.000     0:12830       yes: 3571   0:14680
```

```
##  1st Qu.: 3.000     1: 2169       no :11428   1:  319
##  Median : 3.000
##  Mean   : 3.498
##  3rd Qu.: 4.000
##  Max.   :10.000
##
##          sales         salary
##  sales      :4140   high  :1237
##  technical  :2720   low   :7316
##  support    :2229   medium:6446
##  IT         :1227
##  product_mng: 902
##  marketing  : 858
##  (Other)    :2923
```

```
## Partition the dataset by "time_over_5"
X <- rawData[rawData$time_spend_company < 6, -c(5)]
y <- X$left
tag <- colnames(X)
tag
```

```
## [1] "satisfaction_level"    "last_evaluation"       "number_project"
## [4] "average_montly_hours"  "Work_accident"         "left"
## [7] "promotion_last_5years" "sales"                 "salary"
```

```
# Feature engineering
## Create dummy variables for "sales" and "salary"
library(psych)
```

```
## Warning: package 'psych' was built under R version 4.0.4
```

```
dummySales <- dummy.code(X$sales)
dummySalary <- dummy.code(X$salary)
colnames(dummySales)
```

```
##  [1] "sales"       "technical"   "support"     "IT"          "product_mng"
##  [6] "marketing"   "RandD"       "accounting"  "hr"          "management"
```

```
colnames(dummySalary)
```

```
## [1] "low"    "medium" "high"
```

```
### Set "sales" and "low" as the default values respectively
dummySales <- dummySales[ , -c(1)]
dummySalary <- dummySalary[ , -c(1)]

X_dummy <- cbind(X[ , -c(8, 9)], dummySales, dummySalary)
tag_dummy <- colnames(X_dummy)
tag_dummy
```

```
##  [1] "satisfaction_level"    "last_evaluation"       "number_project"
##  [4] "average_montly_hours"  "Work_accident"         "left"
##  [7] "promotion_last_5years" "technical"             "support"
## [10] "IT"                    "product_mng"           "marketing"
## [13] "RandD"                 "accounting"            "hr"
## [16] "management"            "medium"                "high"
```

```
# Train(80%)-test(20%)-split (stratified as "left" is unbalanced)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.4

## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

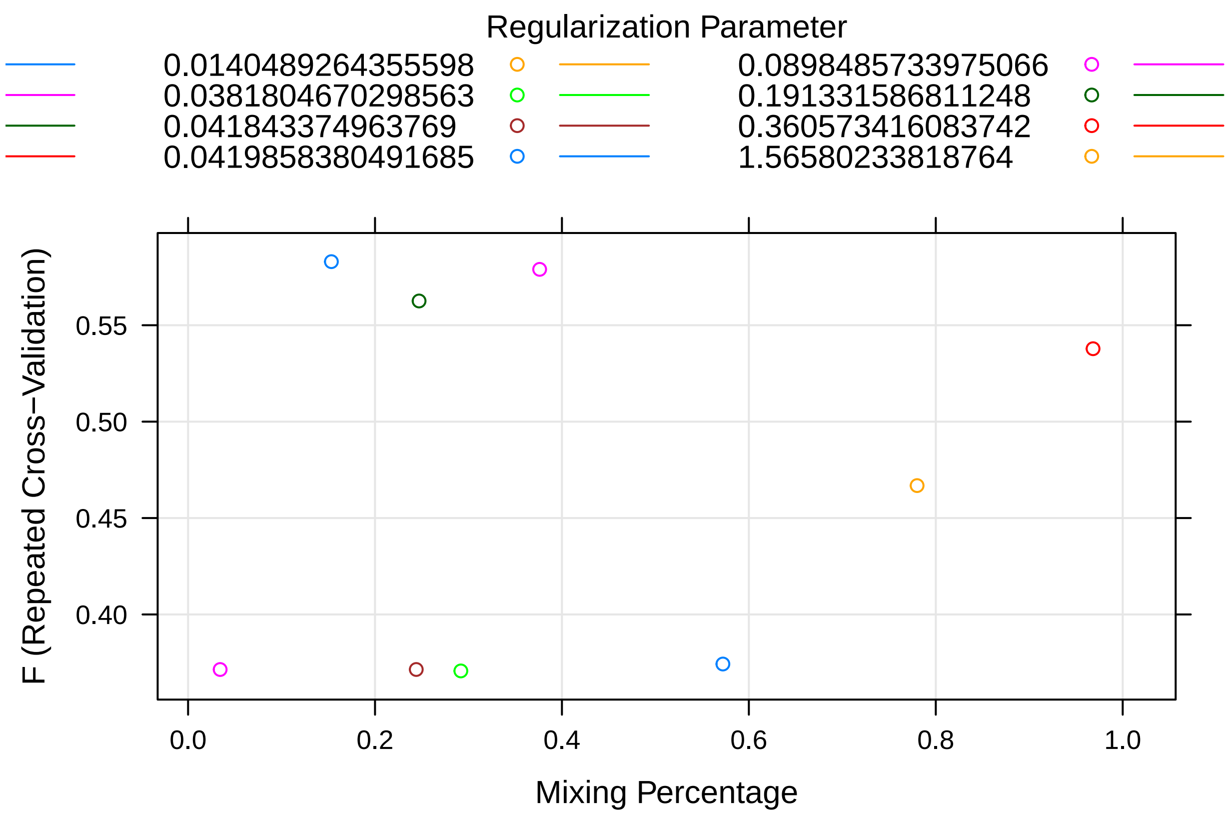
```
## Set seed for replication purpose
set.seed(7860)
index <- createDataPartition(y, p = 0.8, list = FALSE)
X_train <- X[index, ]
X_test <- cbind(X[-index, 1 : 5], X[-index, 7 : length(X)])
y_test <- X[-index, 'left']
X_dummy_train <- X_dummy[index, ]
X_dummy_test <- cbind(X_dummy[-index, 1 : 5], X_dummy[-index, 7 : length(X_dummy)])




# Modeling with extracted factors, 5-fold nested CV with random search
models <- c('svmLinear', 'glmnet', 'rf', 'knn')
n_cluster <- 10 ## Please set the number of multiprocessing slaves accordingly

for (m in models) {
  assign(paste0(m, '_best'), list('model' = c(), 'f1_val' = c(),
                                  'confm' = c()))

  tune <- 15
  control <- trainControl(method = 'repeatedcv', number = 5, repeats = 2,
                          summaryFunction = prSummary, classProbs = TRUE,
                          search="random", verboseIter = TRUE)
  set.seed(7860)

  require(doParallel)
  cl <- makePSOCKcluster(n_cluster, outfile = '')
  registerDoParallel(cl)
```

```
  if (m == 'rf') {
    m1 <- train(left ~ ., data = X_train, method = m,
                metric = 'F', tuneLength = tune, trControl = control)
    rf_best[['model']] <- m1
    rf_best[['f1_val']] <- F_meas(predict(m1, X_test), y_test)
    rf_best[['confm']] <- confusionMatrix(predict(m1, X_test), y_test)
  } else if (m == 'glmnet') {
    m1 <- train(left ~ ., data = cbind(scale(X_dummy_train[ , 1 : 4]), X_dummy_train[ , 5 : length(X_du
                method = m, family = 'binomial',
                metric = 'F', tuneLength = tune, trControl = control)
    glmnet_best[['model']] <- m1
    glmnet_best[['f1_val']] <- F_meas(predict(m1, cbind(scale(X_dummy_test[ , 1 : 4]), X_dummy_test[ , 5
    glmnet_best[['confm']] <- confusionMatrix(predict(m1, cbind(scale(X_dummy_test[ , 1 : 4]), X_dummy_t
  } else if (m == 'knn') {
    m1 <- train(left ~ ., data =  cbind(scale(X_dummy_train[ , 1 : 4]), X_dummy_train[ , 5 : length(X_du
                metric = 'F', tuneLength = tune, trControl = control, tuneGrid = expand.grid(k = c(2, 3
    knn_best[['model']] <- m1
    knn_best[['f1_val']] <- F_meas(predict(m1, cbind(scale(X_dummy_test[ , 1 : 4]), X_dummy_test[ , 5 :
    knn_best[['confm']] <- confusionMatrix(predict(m1, cbind(scale(X_dummy_test[ , 1 : 4]), X_dummy_tes
  } else {
    m1 <- train(left ~ ., data =  cbind(scale(X_dummy_train[ , 1 : 4]), X_dummy_train[ , 5 : length(X_du
                metric = 'F', tuneLength = tune, trControl = control)
    svmLinear_best[['model']] <- m1
    svmLinear_best[['f1_val']] <- F_meas(predict(m1, cbind(scale(X_dummy_test[ , 1 : 4]), X_dummy_test[
    svmLinear_best[['confm']] <- confusionMatrix(predict(m1, cbind(scale(X_dummy_test[ , 1 : 4]), X_dumm
  }

  stopImplicitCluster()
  stopCluster(cl)
}
```

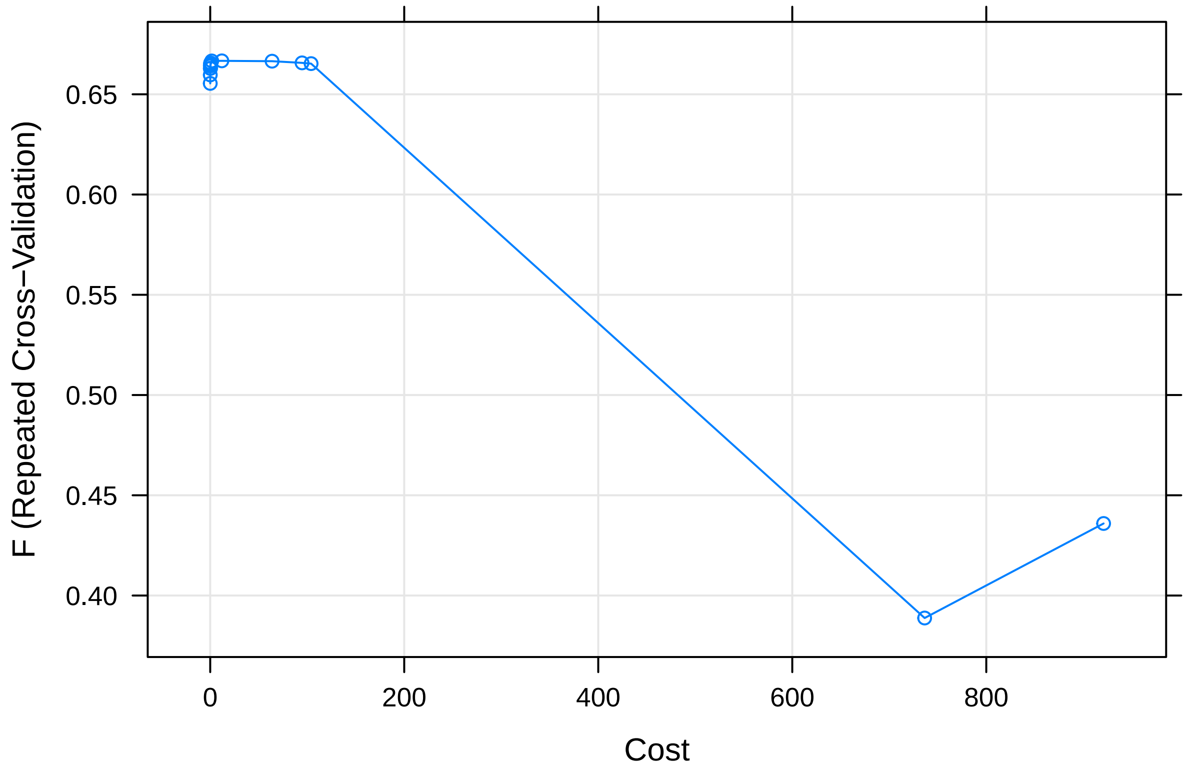
```
## Loading required package: doParallel

## Warning: package 'doParallel' was built under R version 4.0.4

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 4.0.4

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 4.0.4

## Loading required package: parallel

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.

## Aggregating results
## Selecting tuning parameters
## Fitting C = 12 on full training set
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.

## Aggregating results

## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results

## Selecting tuning parameters
## Fitting alpha = 0.153, lambda = 0.00124 on full training set
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 7 on full training set
## Aggregating results
## Selecting tuning parameters
## Fitting k = 3 on full training set
```

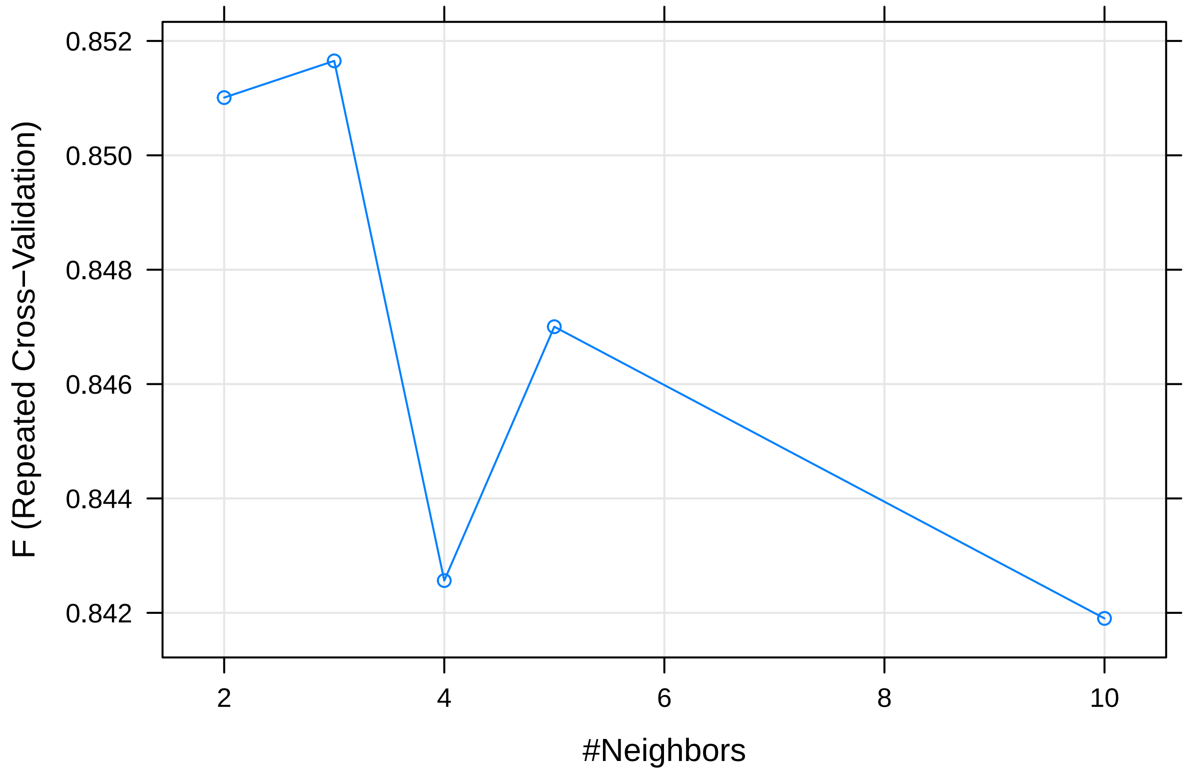
```
results <- as.data.frame(cbind(glmnet_best, svmLinear_best, knn_best, rf_best))

plot(results$glmnet_best$model)
```



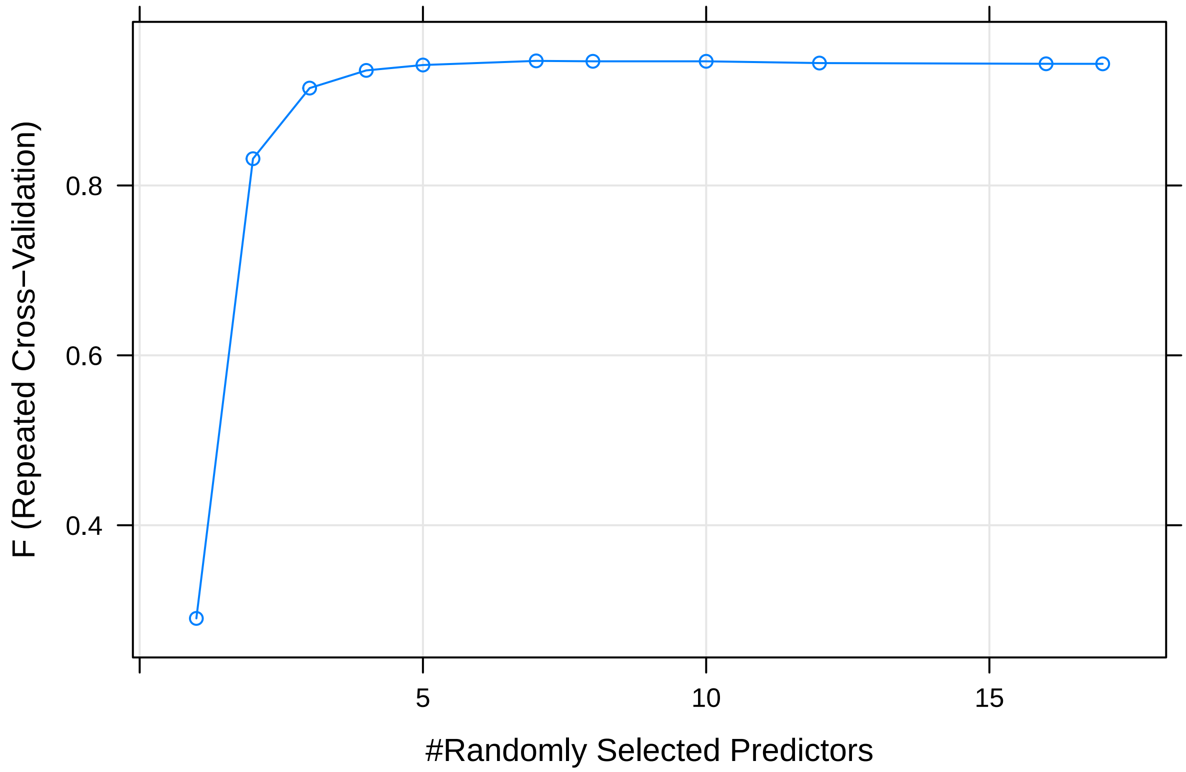

```
plot(results$svmLinear_best$model)
```




```
plot(results$knn_best$model)
```




```
plot(results$rf_best$model)
```




```
for (i in 1 : 4) {
  cat(rep('\n', 3))
  print(results[[i]])
  cat(rep('\n', 3))
}
```

```
##
##
##
## $model
## glmnet
##
## 10974 samples
##    17 predictor
##     2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 8779, 8779, 8779, 8779, 8780, 8779, ...
## Resampling results across tuning parameters:
##
##   alpha       lambda       AUC        Precision  Recall     F
##   0.03431740  0.089848573  0.6354293  0.7564402  0.2462825  0.3714456
##   0.08249613  0.360573416  0.6007119        NaN  0.0000000        NaN
##   0.11032045  5.835644047  0.0000000        NaN  0.0000000        NaN
##   0.14298028  5.900723701  0.0000000        NaN  0.0000000        NaN
```

```
##   0.15333117  0.001237096  0.6132379  0.7074950  0.4960967  0.5829779
##   0.24414545  0.041843375  0.6328153  0.6459675  0.2607807  0.3714583
##   0.24717296  0.005363840  0.6159915  0.7076353  0.4671004  0.5625732
##   0.29186267  0.038180467  0.6322243  0.6393197  0.2611524  0.3707096
##   0.37615343  0.002088687  0.6142002  0.7077905  0.4901487  0.5789760
##   0.57222747  0.041985838  0.6372825  0.6462069  0.2635688  0.3742908
##   0.73301844  2.449870885  0.0000000        NaN  0.0000000        NaN
##   0.77100396  0.191331587  0.5770621        NaN  0.0000000        NaN
##   0.78001638  0.014048926  0.6241281  0.6653814  0.3596654  0.4668294
##   0.96830053  0.005986254  0.6184896  0.6953754  0.4386617  0.5378155
##   0.98979898  1.565802338  0.0000000        NaN  0.0000000        NaN
##
## F was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.1533312 and lambda
##  = 0.001237096.
##
## $f1_val
## [1] 0.5547703
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  314  146
##        no   358 1925
##
##                Accuracy : 0.8163
##                  95% CI : (0.8012, 0.8306)
##     No Information Rate : 0.755
##     P-Value [Acc > NIR] : 8.457e-15
##
##                   Kappa : 0.4441
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4673
##             Specificity : 0.9295
##          Pos Pred Value : 0.6826
##          Neg Pred Value : 0.8432
##              Prevalence : 0.2450
##          Detection Rate : 0.1145
##    Detection Prevalence : 0.1677
##       Balanced Accuracy : 0.6984
##
##        'Positive' Class : yes
##
##
##
##
##
##
##
##
```

```
## $model
## Support Vector Machines with Linear Kernel
##
## 10974 samples
##    17 predictor
##     2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 8779, 8779, 8779, 8779, 8780, 8779, ...
## Resampling results across tuning parameters:
##
##   C             AUC        Precision  Recall     F
##     0.04464867  0.6022223  0.7371450  0.5907063  0.6554272
##     0.07368128  0.6004907  0.7398197  0.5959108  0.6596572
##     0.09840031  0.5999223  0.7405653  0.6003717  0.6629115
##     0.13818800  0.5994605  0.7413602  0.6018587  0.6641282
##     0.15388959  0.5993138  0.7419980  0.6020446  0.6644928
##     0.39561848  0.5986093  0.7419756  0.5996283  0.6629700
##     0.40826969  0.5986001  0.7434283  0.6029740  0.6656199
##     0.64974256  0.5984318  0.7428439  0.6027881  0.6652099
##     1.56081702  0.5983204  0.7434946  0.6044610  0.6666199
##    11.98711010  0.5982231  0.7441789  0.6042751  0.6666691
##    63.79080326  0.5981868  0.7434881  0.6042751  0.6664944
##    94.68470164  0.5979691  0.7435708  0.6029740  0.6656624
##   103.98601383  0.5977665  0.7433567  0.6027881  0.6652898
##   736.48273800  0.4965342  0.4521771  0.4308550  0.3887858
##   920.95367219  0.5634040  0.4256875  0.3426270  0.4359383
##
## F was used to select the optimal model using the largest value.
## The final value used for the model was C = 11.98711.
##
## $f1_val
## [1] 0.6497129
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  396  151
##        no   276 1920
##
##                Accuracy : 0.8443
##                  95% CI : (0.8302, 0.8577)
##     No Information Rate : 0.755
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.551
##
##  Mcnemar's Test P-Value : 1.964e-09
##
##             Sensitivity : 0.5893
##             Specificity : 0.9271
```

```
##          Pos Pred Value : 0.7239
##          Neg Pred Value : 0.8743
##              Prevalence : 0.2450
##          Detection Rate : 0.1444
##    Detection Prevalence : 0.1994
##       Balanced Accuracy : 0.7582
##
##        'Positive' Class : yes
##
##
##
##
##
##
##
##
## $model
## k-Nearest Neighbors
##
## 10974 samples
##    17 predictor
##     2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 8779, 8779, 8779, 8779, 8780, 8779, ...
## Resampling results across tuning parameters:
##
##   k   AUC        Precision  Recall     F
##    2  0.1057112  0.8091698  0.8975836  0.8510093
##    3  0.1634299  0.8127667  0.8946097  0.8516513
##    4  0.1980941  0.8078878  0.8804833  0.8425641
##    5  0.2180272  0.8216192  0.8741636  0.8470031
##   10  0.2930483  0.8246629  0.8602230  0.8419024
##
## F was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
##
## $f1_val
## [1] 0.8521618
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  610  151
##        no    62 1920
##
##                Accuracy : 0.9223
##                  95% CI : (0.9117, 0.9321)
##     No Information Rate : 0.755
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.7991
##
##  Mcnemar's Test P-Value : 1.643e-09
##
##            Sensitivity : 0.9077
##            Specificity : 0.9271
##         Pos Pred Value : 0.8016
##         Neg Pred Value : 0.9687
##             Prevalence : 0.2450
##         Detection Rate : 0.2224
##   Detection Prevalence : 0.2774
##      Balanced Accuracy : 0.9174
##
##       'Positive' Class : yes
##
##
##
##
##
##
##
##
## $model
## Random Forest
##
## 10974 samples
##     8 predictor
##     2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 8779, 8779, 8779, 8779, 8780, 8779, ...
## Resampling results across tuning parameters:
##
##   mtry  AUC        Precision  Recall     F
##    1    0.8616727  0.9967071  0.1795539  0.2902993
##    2    0.9487339  0.9869695  0.7184015  0.8314906
##    3    0.9651951  0.9708307  0.8646840  0.9146202
##    4    0.8739066  0.9599272  0.9122677  0.9354524
##    5    0.6965537  0.9571933  0.9269517  0.9418093
##    7    0.4691848  0.9563755  0.9371747  0.9466568
##    8    0.4376008  0.9547790  0.9377323  0.9461573
##   10    0.3791470  0.9530556  0.9394052  0.9461679
##   12    0.3469535  0.9497937  0.9384758  0.9440835
##   16    0.3012814  0.9471755  0.9394052  0.9432464
##   17    0.2943130  0.9450048  0.9414498  0.9431920
##
## F was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 7.
##
## $f1_val
## [1] 0.9663426
##
## $confm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  646   19
##        no    26 2052
##
##                Accuracy : 0.9836
##                  95% CI : (0.9781, 0.988)
##     No Information Rate : 0.755
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9555
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9613
##             Specificity : 0.9908
##          Pos Pred Value : 0.9714
##          Neg Pred Value : 0.9875
##              Prevalence : 0.2450
##          Detection Rate : 0.2355
##    Detection Prevalence : 0.2424
##       Balanced Accuracy : 0.9761
##
##        'Positive' Class : yes
##
##
##
##
##
```

```
save.image("D:/Yaxin/HKBU BM/Courses/Sem 2/ECON7860 Big Data Analytics for Business (S11)/Group Project/
```