Modeling_org.R

yaxin

2021-04-10

```
# 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
```

```
# Separate target variable
X <- rawData
y <- X$left
tag <- colnames(X)
tag
```

```
##  [1] "satisfaction_level"    "last_evaluation"       "number_project"
##  [4] "average_montly_hours"  "time_spend_company"    "Work_accident"
##  [7] "left"                  "promotion_last_5years" "sales"
## [10] "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(9, 10)], dummySales, dummySalary)
tag_dummy <- colnames(X_dummy)
tag_dummy
```

```
##  [1] "satisfaction_level"    "last_evaluation"       "number_project"
##  [4] "average_montly_hours"  "time_spend_company"    "Work_accident"
##  [7] "left"                  "promotion_last_5years" "technical"
## [10] "support"               "IT"                    "product_mng"
## [13] "marketing"             "RandD"                 "accounting"
## [16] "hr"                    "management"            "medium"
## [19] "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 : 6], X[-index, 8 : length(X)])
y_test <- X[-index, 'left']
X_dummy_train <- X_dummy[index, ]
X_dummy_test <- cbind(X_dummy[-index, 1 : 6], X_dummy[-index, 8 : length(X_dummy)])
```

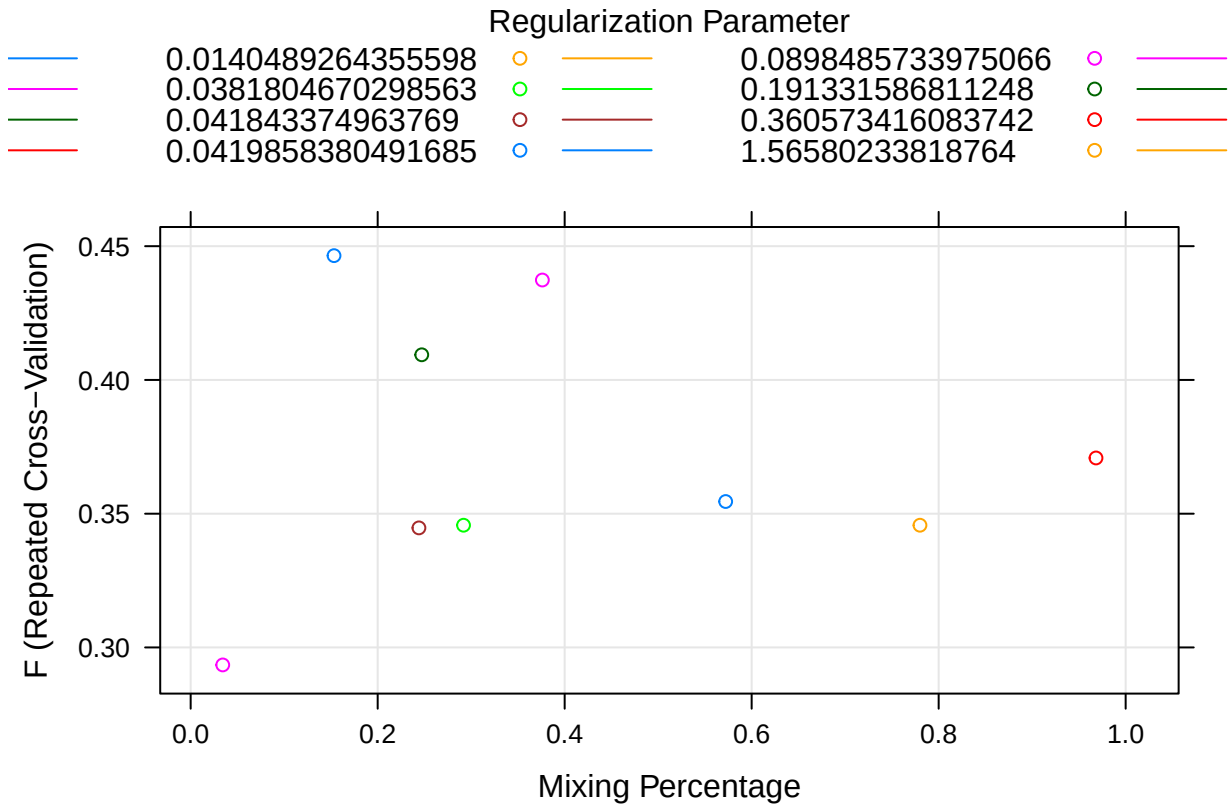
```
# Modeling with extracted factors, 5-fold nested CV with random search
models <- c('svmLinear', 'glmnet', 'rf', 'knn')
n_cluster <- 10 ## Please set the number of multiprocessing slaves accordingly

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 : 5]), X_dummy_train[ , 6 : length(X_dum
                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 : 5]), X_dummy_test[ , 
    glmnet_best[['confm']] <- confusionMatrix(predict(m1, cbind(scale(X_dummy_test[ , 1 : 5]), X_dummy_
  } else if (m == 'knn') {
    m1 <- train(left ~ ., data =  cbind(scale(X_dummy_train[ , 1 : 5]), X_dummy_train[ , 6 : 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 : 5]), X_dummy_test[ , 6 :
    knn_best[['confm']] <- confusionMatrix(predict(m1, cbind(scale(X_dummy_test[ , 1 : 5]), X_dummy_tes
  } else {
    m1 <- train(left ~ ., data =  cbind(scale(X_dummy_train[ , 1 : 5]), X_dummy_train[ , 6 : 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 : 5]), X_dummy_test[
    svmLinear_best[['confm']] <- confusionMatrix(predict(m1, cbind(scale(X_dummy_test[ , 1 : 5]), X_dumm
  }

  stopImplicitCluster()
  stopCluster(cl)
}
```

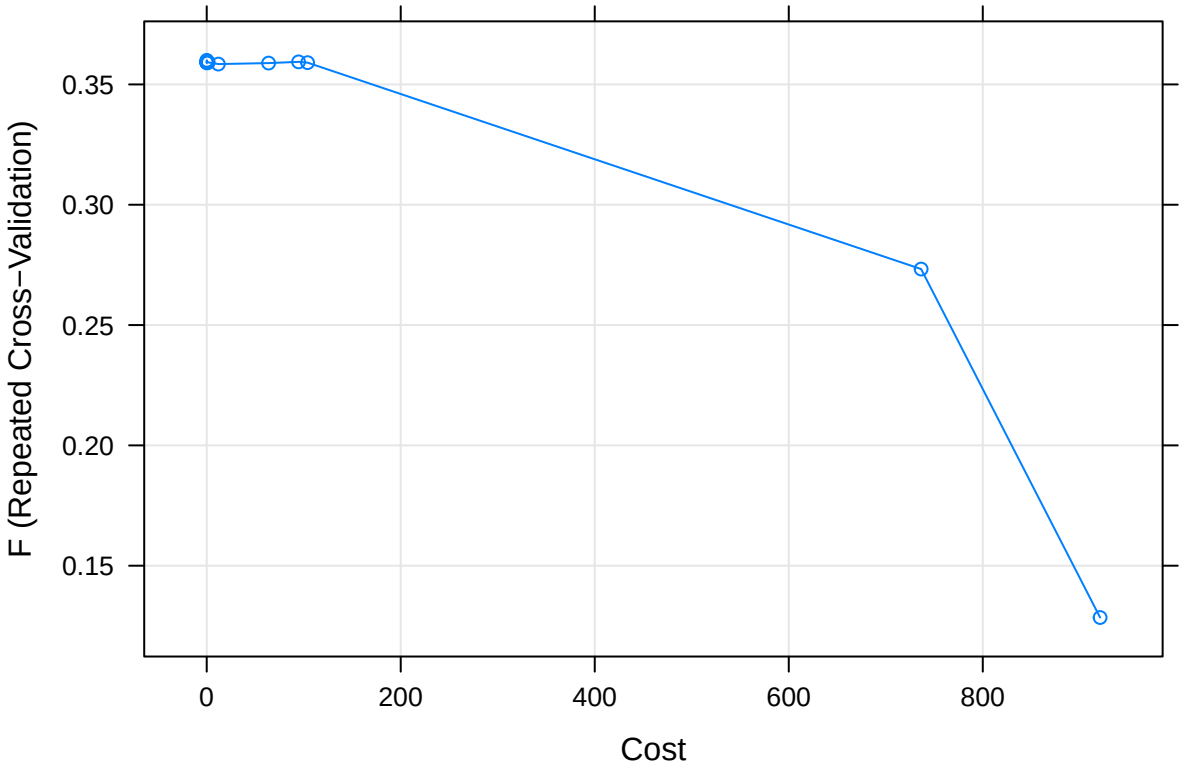
```
## Loading required package: doParallel

## 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 = 0.0984 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 = 2 on full training set
```

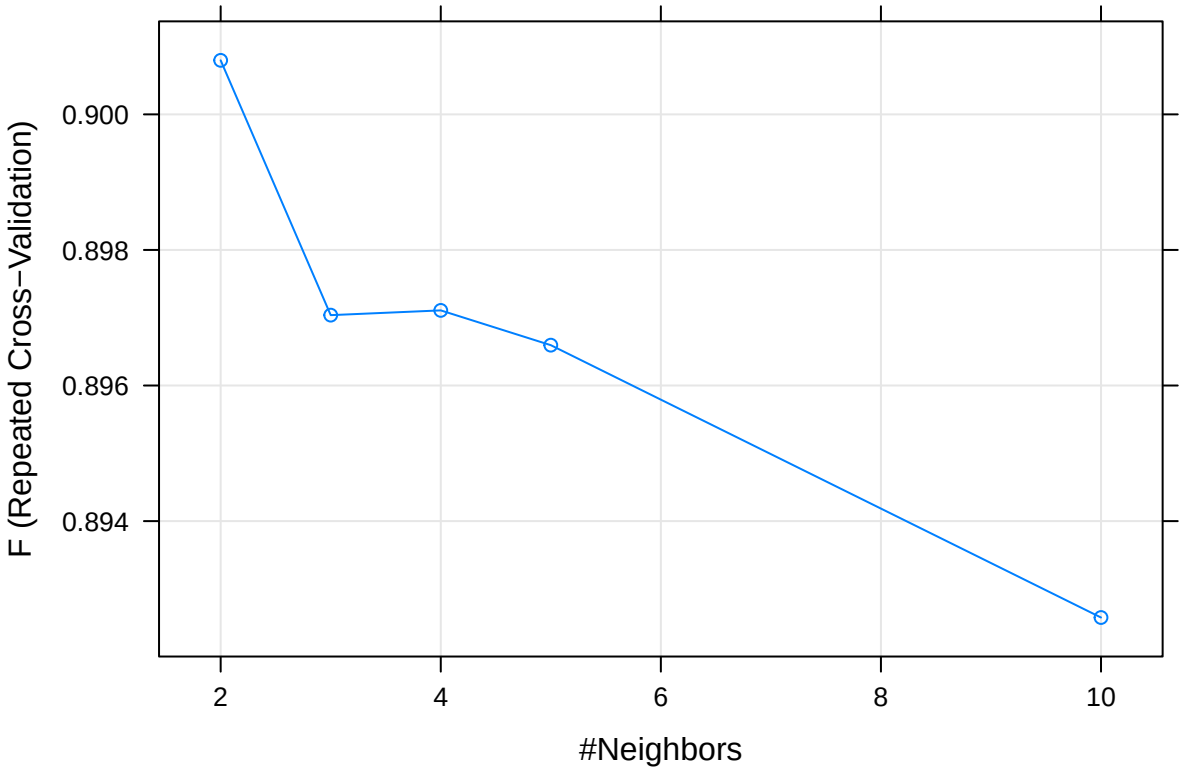
```
results <- as.data.frame(cbind(glmnet_best, svmLinear_best, knn_best, rf_best))

plot(results$glmnet_best$model)
```



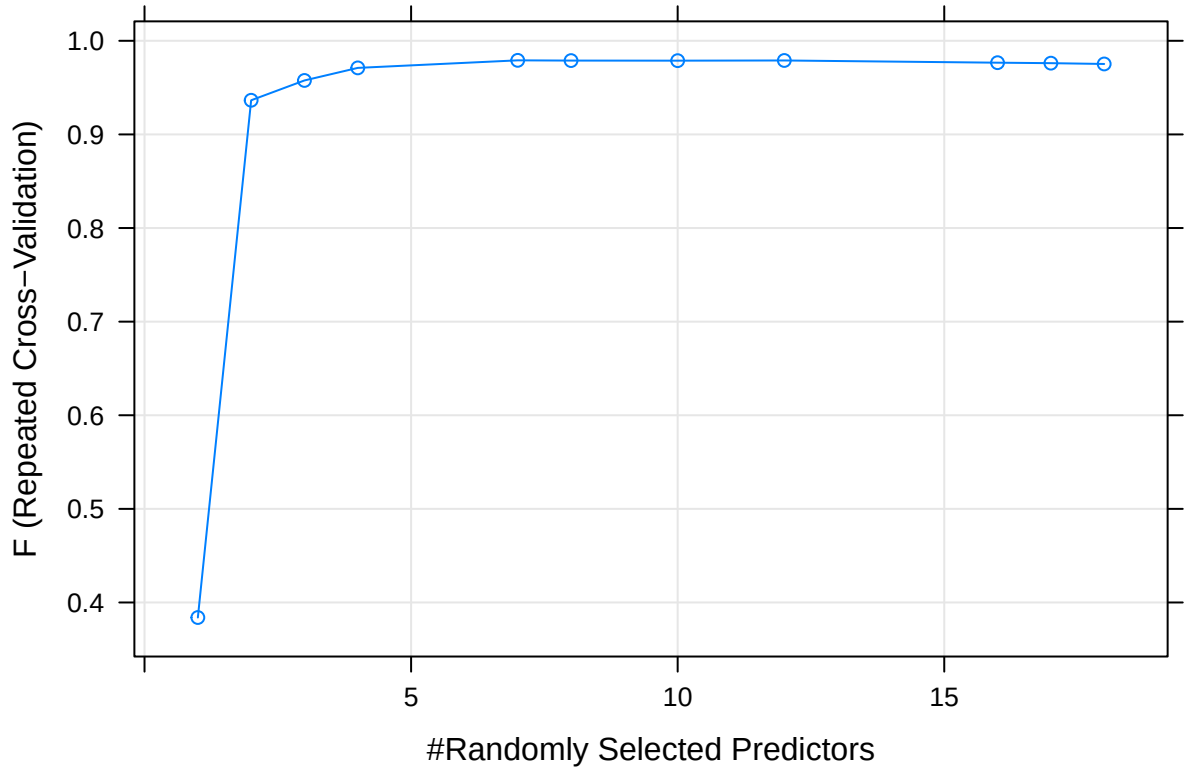

```
plot(results$svmLinear_best$model)
```




```
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
##
## 12000 samples
##    18 predictor
##     2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 9599, 9600, 9600, 9600, 9601, 9600, ...
## Resampling results across tuning parameters:
##
##   alpha       lambda       AUC        Precision  Recall     F
##   0.03431740  0.089848573  0.5388516  0.6861101  0.1869095  0.2934622
##   0.08249613  0.360573416  0.5405829        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  0.5252566  0.6079448  0.3531769  0.4464667
##   0.24414545  0.041843375  0.5393145  0.6208542  0.2387166  0.3446955
##   0.24717296  0.005363840  0.5268066  0.5906081  0.3138014  0.4093979
##   0.29186267  0.038180467  0.5396244  0.6133864  0.2408172  0.3456930
##   0.37615343  0.002088687  0.5257697  0.6036659  0.3433762  0.4373502
##   0.57222747  0.041985838  0.5522347  0.6626281  0.2422195  0.3545505
##   0.73301844  2.449870885  0.0000000        NaN  0.0000000        NaN
##   0.77100396  0.191331587  0.5329404        NaN  0.0000000        NaN
##   0.78001638  0.014048926  0.5350505  0.5648876  0.2492196  0.3456904
##   0.96830053  0.005986254  0.5296256  0.5631106  0.2770508  0.3708367
##   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] 0.4214351
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  232  155
##        no   482 2130
##
##                Accuracy : 0.7876
##                  95% CI : (0.7725, 0.8021)
##     No Information Rate : 0.7619
##     P-Value [Acc > NIR] : 0.0004523
##
##                   Kappa : 0.3051
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.32493
##             Specificity : 0.93217
##          Pos Pred Value : 0.59948
##          Neg Pred Value : 0.81547
##              Prevalence : 0.23808
##          Detection Rate : 0.07736
##    Detection Prevalence : 0.12904
##       Balanced Accuracy : 0.62855
##
##        'Positive' Class : yes
##
##
##
##
##
##
##
##
##
```

```
## $model
## Support Vector Machines with Linear Kernel
##
## 12000 samples
##    18 predictor
##     2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 9599, 9600, 9600, 9600, 9601, 9600, ...
## Resampling results across tuning parameters:
##
##   C             AUC        Precision  Recall      F
##     0.04464867  0.5697948  0.5573399  0.26549821  0.3592145
##     0.07368128  0.5696127  0.5584270  0.26514794  0.3591320
##     0.09840031  0.5695241  0.5595599  0.26584816  0.3600005
##     0.13818800  0.5693214  0.5582580  0.26497281  0.3589521
##     0.15388959  0.5692245  0.5589792  0.26514825  0.3592690
##     0.39561848  0.5690321  0.5596702  0.26532307  0.3595540
##     0.40826969  0.5691036  0.5594245  0.26549821  0.3596579
##     0.64974256  0.5690167  0.5596911  0.26514825  0.3593929
##     1.56081702  0.5689829  0.5591519  0.26497342  0.3591506
##    11.98711010  0.5689307  0.5582344  0.26444834  0.3584523
##    63.79080326  0.5690286  0.5599612  0.26444742  0.3588669
##    94.68470164  0.5693410  0.5607214  0.26479737  0.3593691
##   103.98601383  0.5701933  0.5594224  0.26479768  0.3590361
##   736.48273800  0.5346766  0.5308216  0.22721578  0.2732642
##   920.95367219  0.5518211  0.2660677  0.05621471  0.1284671
##
## F was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.09840031.
##
## $f1_val
## [1] 0.3542857
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  186  150
##        no   528 2135
##
##                Accuracy : 0.7739
##                  95% CI : (0.7585, 0.7888)
##     No Information Rate : 0.7619
##     P-Value [Acc > NIR] : 0.06337
##
##                   Kappa : 0.2382
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.26050
##             Specificity : 0.93435
```

```
##          Pos Pred Value : 0.55357
##          Neg Pred Value : 0.80173
##              Prevalence : 0.23808
##          Detection Rate : 0.06202
##    Detection Prevalence : 0.11204
##       Balanced Accuracy : 0.59743
##
##        'Positive' Class : yes
##
##
##
##
##
##
##
##
## $model
## k-Nearest Neighbors
##
## 12000 samples
##    18 predictor
##     2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 9599, 9600, 9600, 9600, 9601, 9600, ...
## Resampling results across tuning parameters:
##
##   k   AUC         Precision  Recall     F
##    2  0.06404068  0.8651975  0.9396210  0.9007969
##    3  0.09302575  0.8633151  0.9336687  0.8970383
##    4  0.10663038  0.8673362  0.9291196  0.8971084
##    5  0.12023188  0.8714683  0.9233451  0.8965955
##   10  0.17917546  0.8676856  0.9189696  0.8925777
##
## F was used to select the optimal model using the largest value.
## The final value used for the model was k = 2.
##
## $f1_val
## [1] 0.9089692
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  677  101
##        no    37 2184
##
##                Accuracy : 0.954
##                  95% CI : (0.9459, 0.9612)
##     No Information Rate : 0.7619
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.877
##
##  Mcnemar's Test P-Value : 8.189e-08
##
##             Sensitivity : 0.9482
##             Specificity : 0.9558
##          Pos Pred Value : 0.8702
##          Neg Pred Value : 0.9833
##              Prevalence : 0.2381
##          Detection Rate : 0.2257
##    Detection Prevalence : 0.2594
##       Balanced Accuracy : 0.9520
##
##        'Positive' Class : yes
##
##
##
##
##
##
##
##
## $model
## Random Forest
##
## 12000 samples
##     9 predictor
##     2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 9599, 9600, 9600, 9600, 9601, 9600, ...
## Resampling results across tuning parameters:
##
##   mtry  AUC        Precision  Recall     F
##    1    0.8872703  0.9945474  0.2409719  0.3839401
##    2    0.9703780  0.9892893  0.8892160  0.9364912
##    3    0.9756729  0.9884361  0.9287690  0.9576419
##    4    0.8328986  0.9914374  0.9515211  0.9710543
##    7    0.4114758  0.9942426  0.9644710  0.9791161
##    8    0.3777892  0.9935270  0.9646461  0.9788603
##   10    0.3043573  0.9929891  0.9649961  0.9787808
##   12    0.2601088  0.9924573  0.9658711  0.9789754
##   16    0.2169654  0.9881876  0.9653460  0.9766253
##   17    0.2102876  0.9871239  0.9653460  0.9761055
##   18    0.1941002  0.9855402  0.9651712  0.9752388
##
## 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] 0.9794763
##
## $confm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  692    7
##        no    22 2278
##
##                Accuracy : 0.9903
##                  95% CI : (0.9861, 0.9935)
##     No Information Rate : 0.7619
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9732
##
##  Mcnemar's Test P-Value : 0.00933
##
##             Sensitivity : 0.9692
##             Specificity : 0.9969
##          Pos Pred Value : 0.9900
##          Neg Pred Value : 0.9904
##              Prevalence : 0.2381
##          Detection Rate : 0.2307
##    Detection Prevalence : 0.2331
##       Balanced Accuracy : 0.9831
##
##        'Positive' Class : yes
##
##
##
##
##
```

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