Modeling_ind.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"
```

```
## Create indicator variable for "time_spend_company"
time_over_5 <- factor(ifelse(X$time_spend_company > 5, 1, 0))
X <- cbind(X[ , 1 : 4], time_over_5, X[ , 6 : length(X)])
X_dummy <- cbind(X_dummy[ , 1 : 4], time_over_5, X_dummy[ , 6 : length(X_dummy)])
tag1 <- colnames(X)
tag1
```

```
##  [1] "satisfaction_level"    "last_evaluation"       "number_project"
##  [4] "average_montly_hours"  "time_over_5"           "Work_accident"
##  [7] "left"                  "promotion_last_5years" "sales"
## [10] "salary"
```

```
# 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 : 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 = 921 on full training set
```

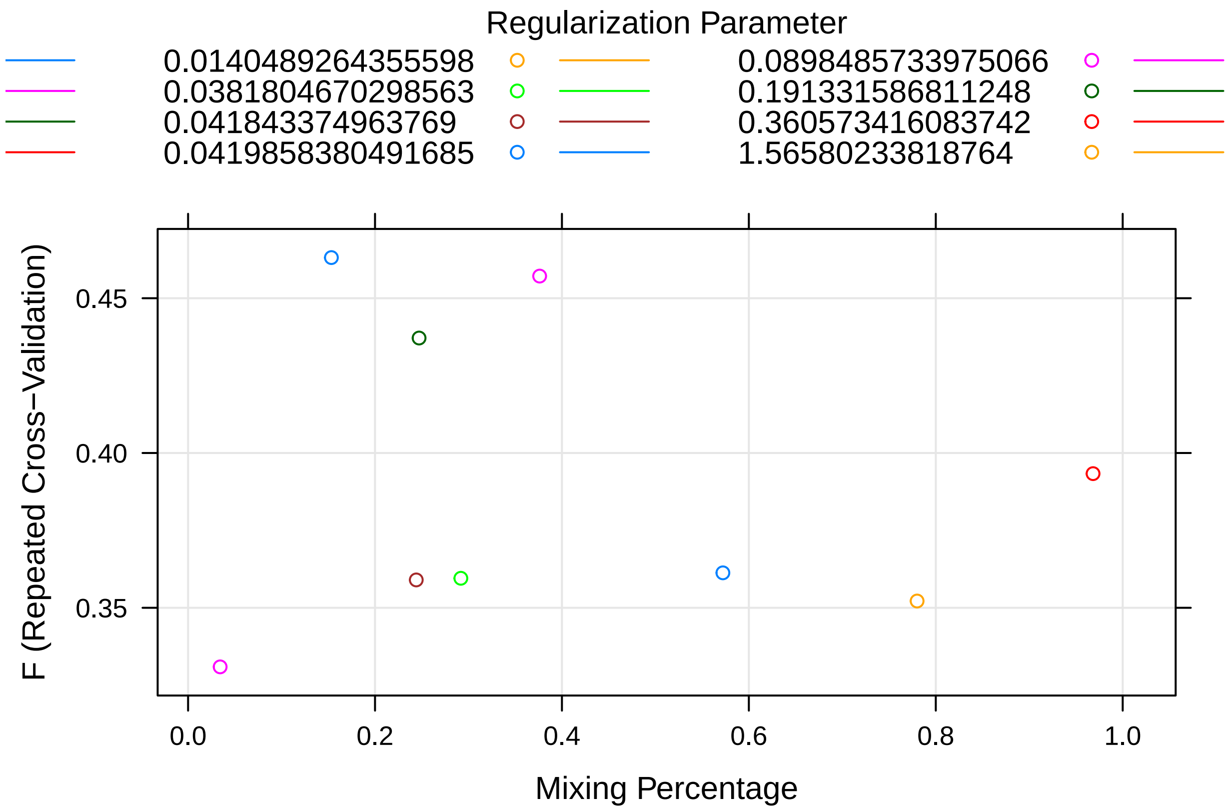
```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
## Aggregating results
```

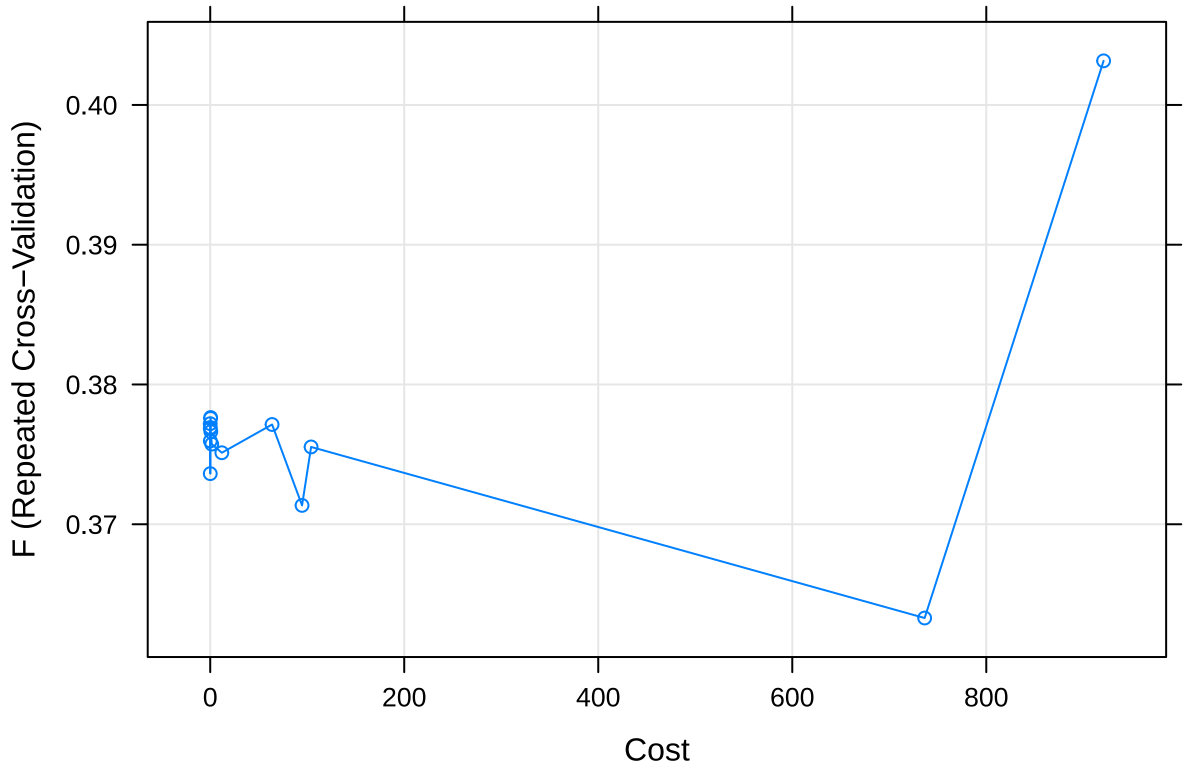
```
## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

```
## 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
```

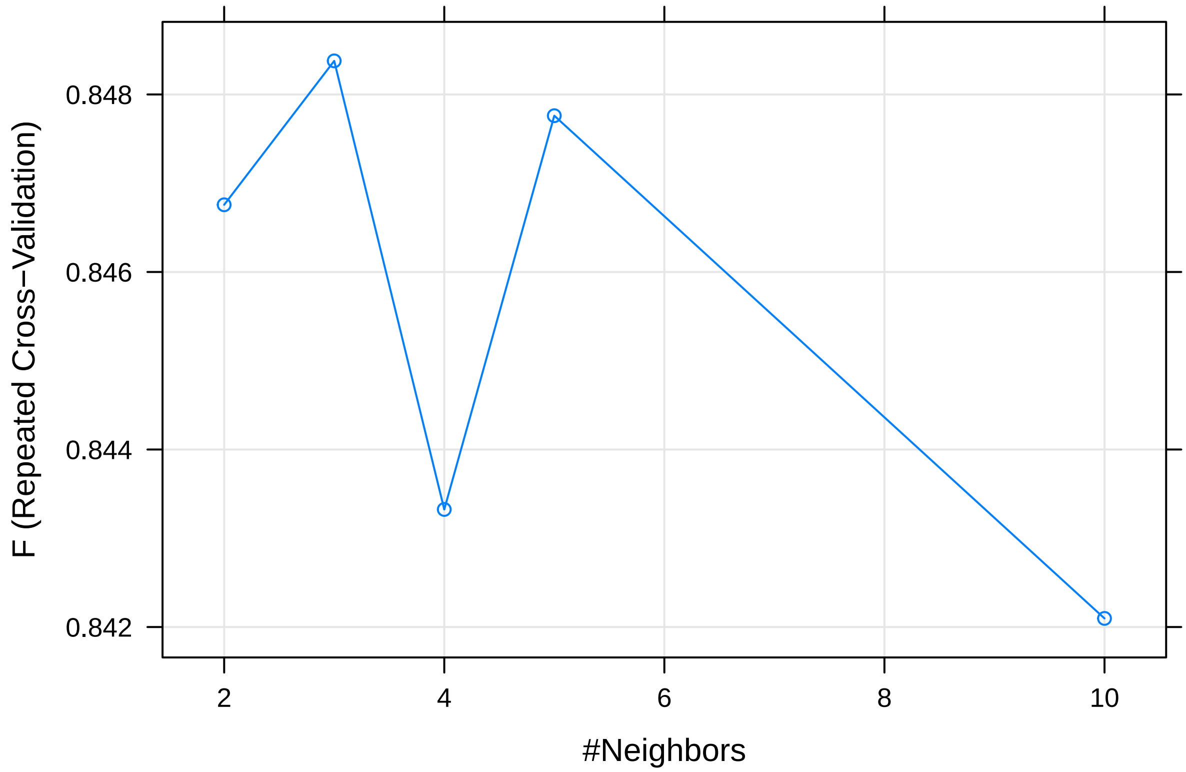
```
results <- as.data.frame(cbind(glmnet_best, svmLinear_best, knn_best, rf_best))

plot(results$glmnet_best$model)
```



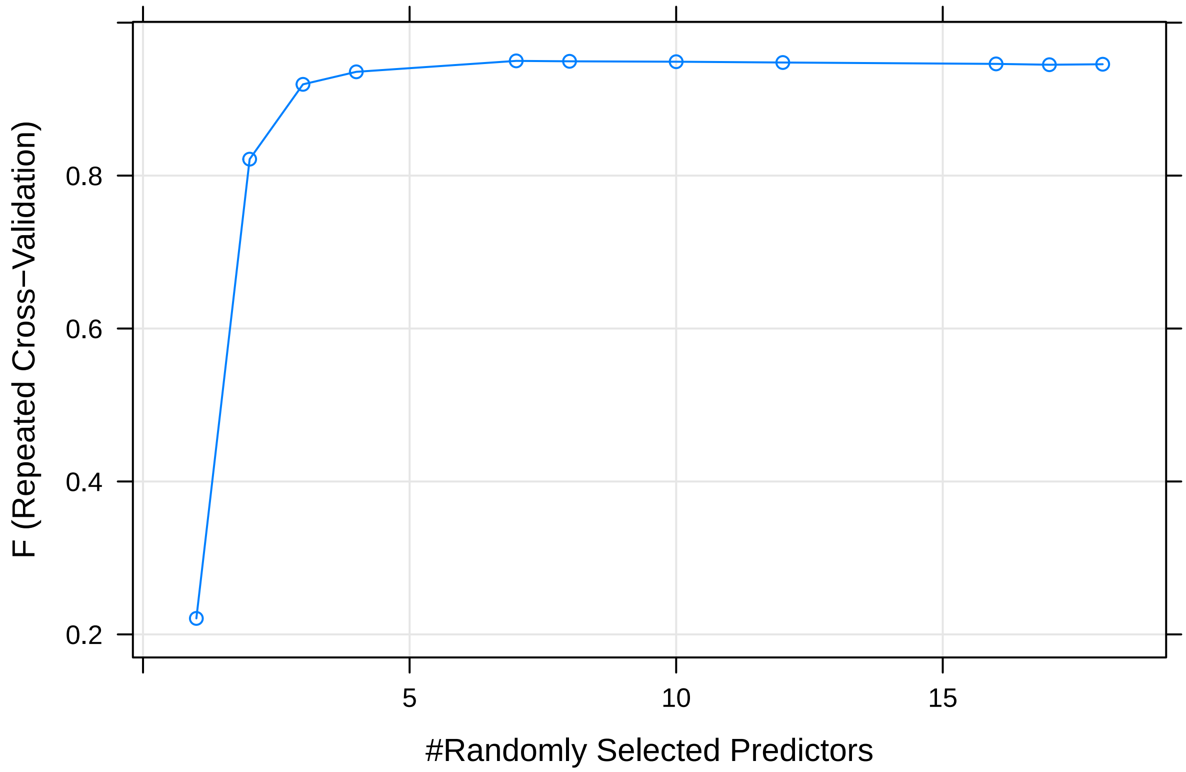

```
plot(results$svmLinear_best$model)
```




```
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
##
## 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  0.5978424  0.8278177  0.2070356  0.3309151
##   0.08249613  0.360573416  0.5770630        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  0.5764040  0.6574883  0.3579011  0.4631284
##   0.24414545  0.041843375  0.5995441  0.7022834  0.2413432  0.3589979
##   0.24717296  0.005363840  0.5809894  0.6607318  0.3271016  0.4371391
##   0.29186267  0.038180467  0.5992936  0.6919188  0.2430939  0.3595342
##   0.37615343  0.002088687  0.5778859  0.6585455  0.3505511  0.4571495
##   0.57222747  0.041985838  0.5720001  0.7066776  0.2429200  0.3612896
##   0.73301844  2.449870885  0.0000000        NaN  0.0000000        NaN
##   0.77100396  0.191331587  0.5329404        NaN  0.0000000        NaN
##   0.78001638  0.014048926  0.5957931  0.6166058  0.2467702  0.3521881
##   0.96830053  0.005986254  0.5867548  0.6313787  0.2861499  0.3933344
##   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] 0.4330275
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  236  140
##        no   478 2145
##
##                Accuracy : 0.7939
##                  95% CI : (0.779, 0.8083)
##     No Information Rate : 0.7619
##     P-Value [Acc > NIR] : 1.606e-05
##
##                   Kappa : 0.3216
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.33053
##             Specificity : 0.93873
##          Pos Pred Value : 0.62766
##          Neg Pred Value : 0.81777
##              Prevalence : 0.23808
##          Detection Rate : 0.07869
##    Detection Prevalence : 0.12538
##       Balanced Accuracy : 0.63463
##
##        '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  0.5978251  0.6159196  0.2726697  0.3771953
##     0.07368128  0.5982951  0.6181018  0.2682988  0.3736204
##     0.09840031  0.5986567  0.6230300  0.2707491  0.3768356
##     0.13818800  0.5987977  0.6198423  0.2703982  0.3759589
##     0.15388959  0.5988409  0.6206067  0.2712751  0.3769227
##     0.39561848  0.5990424  0.6222821  0.2716238  0.3775710
##     0.40826969  0.5990663  0.6219923  0.2717996  0.3776369
##     0.64974256  0.5991235  0.6216020  0.2707497  0.3766250
##     1.56081702  0.5991627  0.6211610  0.2698747  0.3757322
##    11.98711010  0.5992446  0.6218698  0.2691738  0.3751156
##    63.79080326  0.5998981  0.6265558  0.2432994  0.3771364
##    94.68470164  0.5970787  0.6191147  0.2656118  0.3713476
##   103.98601383  0.5992500  0.6204117  0.2698740  0.3755295
##   736.48273800  0.4672218  0.6814965  0.1039190  0.3632910
##   920.95367219  0.5226205  0.6595962  0.1474811  0.4031545
##
## F was used to select the optimal model using the largest value.
## The final value used for the model was C = 920.9537.
##
## $f1_val
## [1] NA
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes    0    0
##        no   714 2285
##
##                Accuracy : 0.7619
##                  95% CI : (0.7463, 0.7771)
##     No Information Rate : 0.7619
##     P-Value [Acc > NIR] : 0.51
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
```

```
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.7619
##              Prevalence : 0.2381
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        '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  0.1149322  0.8069732  0.8909706  0.8467571
##    3  0.1727101  0.8086860  0.8923674  0.8483781
##    4  0.2039229  0.8087259  0.8811679  0.8433248
##    5  0.2307771  0.8194162  0.8783701  0.8477608
##   10  0.3251063  0.8170888  0.8689163  0.8420971
##
## 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] 0.8586456
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  654  154
##        no    60 2131
##
##                Accuracy : 0.9286
##                  95% CI : (0.9188, 0.9376)
##     No Information Rate : 0.7619
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.8118
##
##  Mcnemar's Test P-Value : 2.053e-10
##
##             Sensitivity : 0.9160
##             Specificity : 0.9326
##          Pos Pred Value : 0.8094
##          Neg Pred Value : 0.9726
##              Prevalence : 0.2381
##          Detection Rate : 0.2181
##    Detection Prevalence : 0.2694
##       Balanced Accuracy : 0.9243
##
##        '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    0.8450443  0.9977625  0.1322784  0.2209106
##    2    0.9509879  0.9855131  0.7045902  0.8214880
##    3    0.9660519  0.9647894  0.8781891  0.9193528
##    4    0.8676781  0.9560223  0.9163420  0.9357037
##    7    0.4985774  0.9569444  0.9432960  0.9500074
##    8    0.4618902  0.9554648  0.9436457  0.9494454
##   10    0.4150500  0.9531331  0.9448731  0.9489268
##   12    0.3824791  0.9517627  0.9441723  0.9478962
##   16    0.3468032  0.9479437  0.9443465  0.9460821
##   17    0.3407206  0.9459145  0.9441714  0.9449895
##   18    0.3365850  0.9464936  0.9448716  0.9456151
##
## 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] 0.9579832
##
## $confm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  yes   no
##        yes  684   30
##        no    30 2255
##
##                Accuracy : 0.98
##                  95% CI : (0.9743, 0.9847)
##     No Information Rate : 0.7619
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9449
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9580
##             Specificity : 0.9869
##          Pos Pred Value : 0.9580
##          Neg Pred Value : 0.9869
##              Prevalence : 0.2381
##          Detection Rate : 0.2281
##    Detection Prevalence : 0.2381
##       Balanced Accuracy : 0.9724
##
##        'Positive' Class : yes
##
##
##
##
##
```

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