Modeling_ind_1.R

yaxin

2021-04-12

```r
# 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$time_spend_company > 5, -c(5)]
y <- X$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"     "management"  "marketing"
##  [6] "IT"          "product_mng" "accounting"  "RandD"       "hr"
```

```
colnames(dummySalary)
```

```
## [1] "medium" "low"    "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] "management"            "marketing"             "IT"
## [13] "product_mng"           "accounting"            "RandD"
## [16] "hr"                    "low"                   "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
```

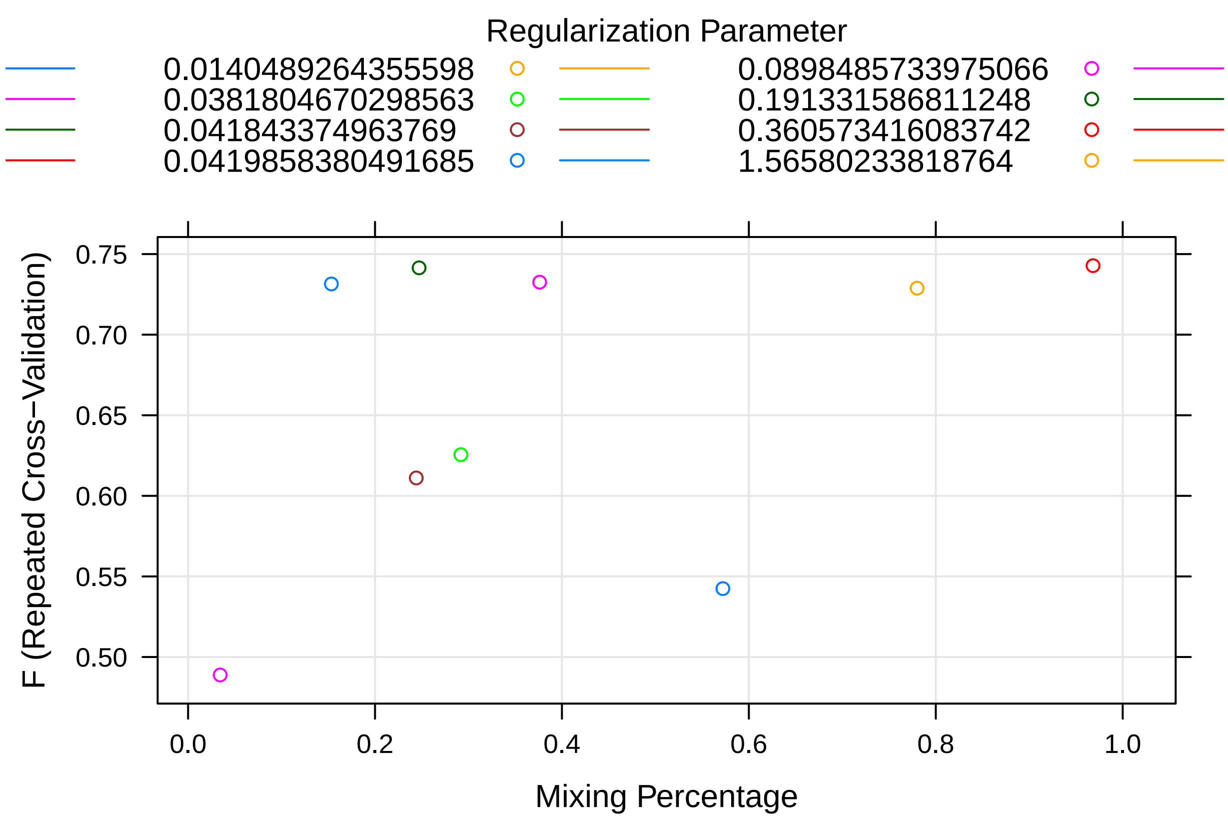
```
## Set seed for replication purpose
set.seed(7860)
index <- createDataPartition(y, p = 0.8, list = FALSE)
X_train <- X[index, ]
X_test <- cbind(X[-index, 1 : 5], X[-index, 7 : length(X)])
y_test <- X[-index, 'left']
X_dummy_train <- X_dummy[index, ]
X_dummy_test <- cbind(X_dummy[-index, 1 : 5], X_dummy[-index,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

```r
  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)
}
```

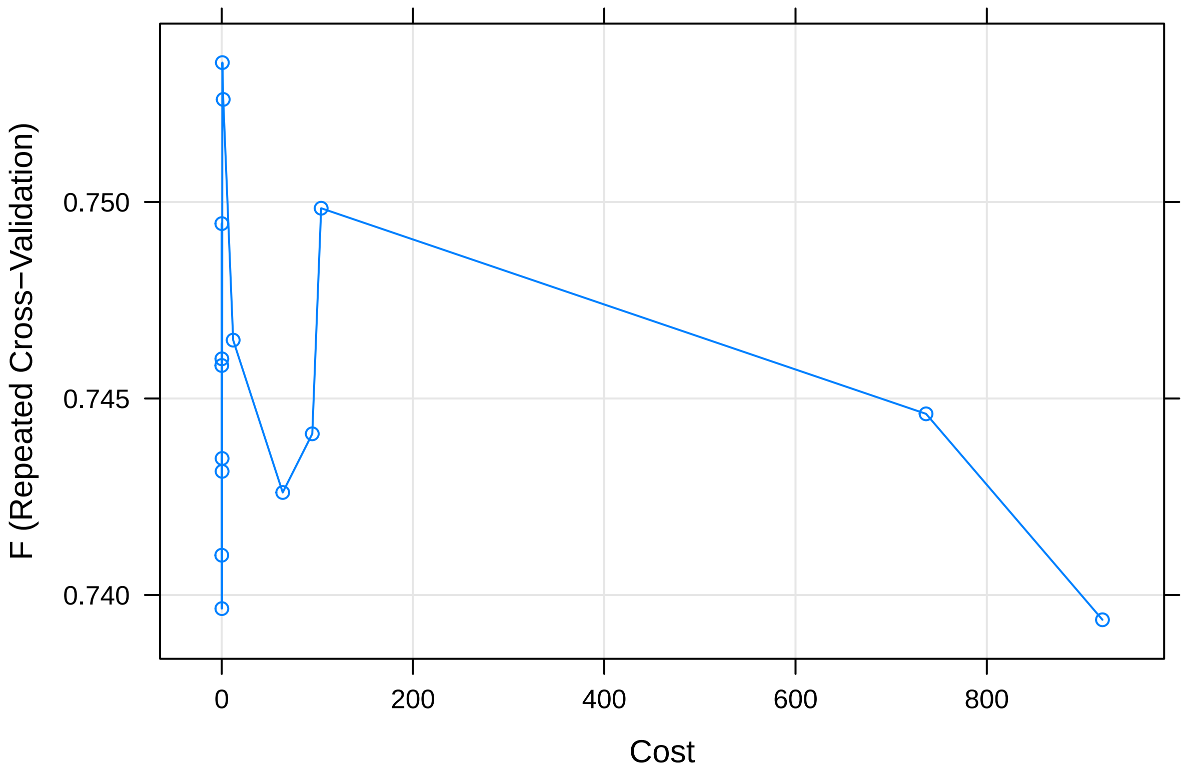
```
## Loading required package: doParallel

## 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

## Aggregating results
## Selecting tuning parameters
## Fitting C = 0.65 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 = 0.968, lambda = 0.00599 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
## Selecting tuning parameters
## Fitting mtry = 4 on full training set
## Aggregating results
## Selecting tuning parameters
## Fitting k = 2 on full training set
```

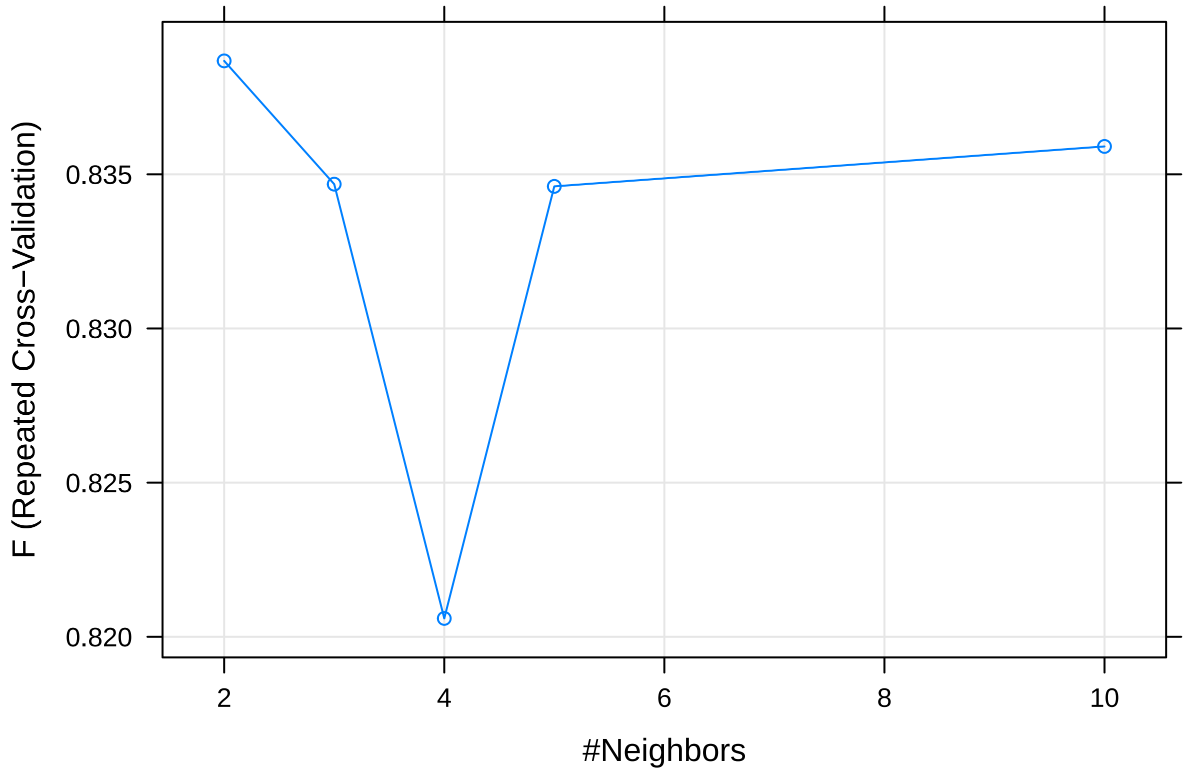
```r
results <- as.data.frame(cbind(glmnet_best, svmLinear_best, knn_best, rf_best))

plot(results$glmnet_best$model)
```



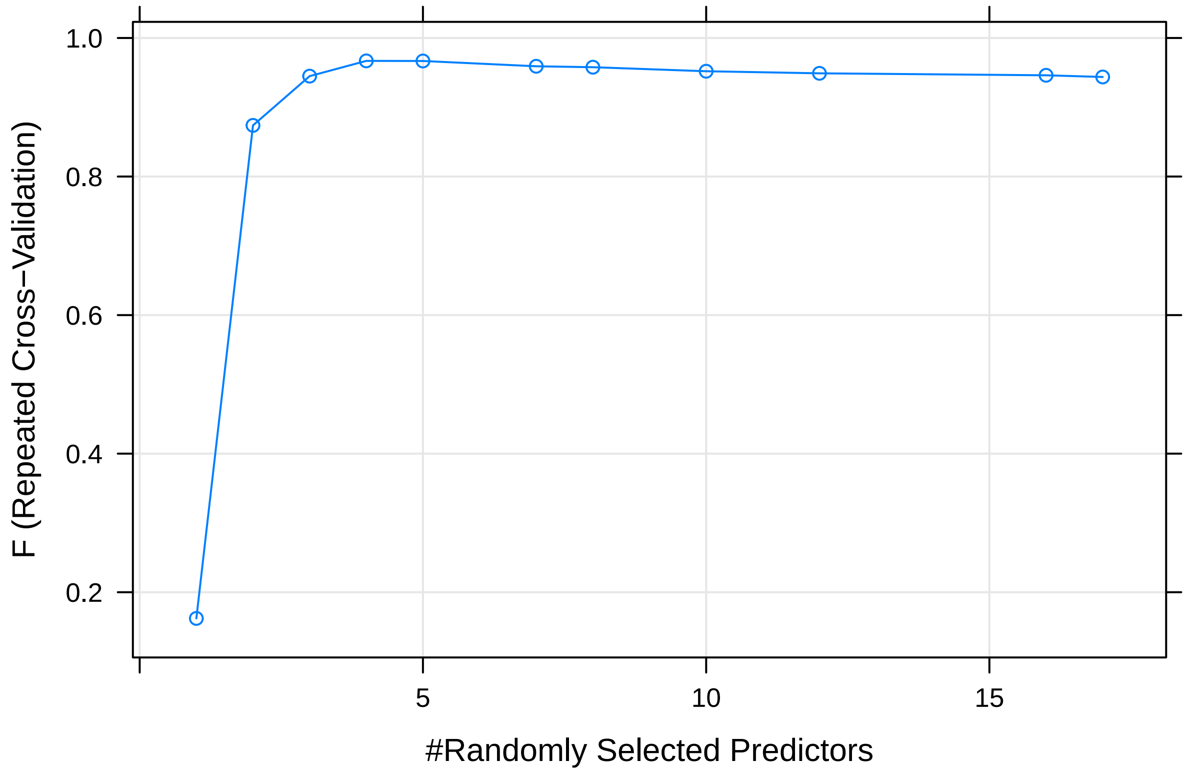
```
plot(results$svmLinear_best$model)
```




```
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
##
## 1027 samples
##   17 predictor
##    2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 822, 822, 821, 821, 822, 821, ...
## Resampling results across tuning parameters:
##
##   alpha       lambda       AUC        Precision  Recall     F
##   0.03431740  0.089848573  0.8182920  0.9640891  0.3338681  0.4888865
##   0.08249613  0.360573416  0.7999695        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  0.8239647  0.7575614  0.7112299  0.7314827
##   0.24414545  0.041843375  0.8196371  0.8991803  0.4678253  0.6111017
##   0.24717296  0.005363840  0.8268326  0.7781371  0.7112299  0.7414642
##   0.29186267  0.038180467  0.8192147  0.8886713  0.4885918  0.6255861
##   0.37615343  0.002088687  0.8255250  0.7596960  0.7112299  0.7325697
##   0.57222747  0.041985838  0.8070457  0.9191497  0.3873440  0.5424405
##   0.73301844  2.449870885  0.0000000        NaN  0.0000000        NaN
##   0.77100396  0.191331587  0.4079394        NaN  0.0000000        NaN
##   0.78001638  0.014048926  0.8244181  0.8241240  0.6579323  0.7288491
##   0.96830053  0.005986254  0.8285448  0.7892837  0.7053476  0.7428176
##   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 = 0.9683005 and lambda
##  = 0.005986254.
##
## $f1_val
## [1] 0.72
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction yes  no
##        yes  27   7
##        no   14 207
##
##                Accuracy : 0.9176
##                  95% CI : (0.8769, 0.9483)
##     No Information Rate : 0.8392
##     P-Value [Acc > NIR] : 0.0001726
##
##                   Kappa : 0.6722
##
##  Mcnemar's Test P-Value : 0.1904303
##
##             Sensitivity : 0.6585
##             Specificity : 0.9673
##          Pos Pred Value : 0.7941
##          Neg Pred Value : 0.9367
##              Prevalence : 0.1608
##          Detection Rate : 0.1059
##    Detection Prevalence : 0.1333
##       Balanced Accuracy : 0.8129
##
##        'Positive' Class : yes
##
##
##
##
##
##
##
##
```

```
## $model
## Support Vector Machines with Linear Kernel
##
## 1027 samples
##   17 predictor
##    2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 822, 822, 821, 821, 822, 821, ...
## Resampling results across tuning parameters:
##
##   C             AUC        Precision  Recall     F
##     0.04464867  0.8184956  0.7734722  0.7170232  0.7410115
##     0.07368128  0.8204057  0.7758861  0.7229947  0.7458394
##     0.09840031  0.8219985  0.7697447  0.7347594  0.7494493
##     0.13818800  0.8218191  0.7731029  0.7141711  0.7396531
##     0.15388959  0.8220517  0.7724129  0.7258467  0.7460088
##     0.39561848  0.8191100  0.7721789  0.7200535  0.7431460
##     0.40826969  0.8185492  0.7751091  0.7169340  0.7434739
##     0.64974256  0.8195639  0.7760782  0.7378788  0.7535457
##     1.56081702  0.8185487  0.7824856  0.7287879  0.7526092
##    11.98711010  0.8179792  0.7772031  0.7229947  0.7464844
##    63.79080326  0.8166356  0.7747801  0.7172014  0.7426080
##    94.68470164  0.8166024  0.7836226  0.7141711  0.7441011
##   103.98601383  0.8166950  0.7837313  0.7229947  0.7498403
##   736.48273800  0.8166645  0.7761573  0.7201426  0.7446090
##   920.95367219  0.8166853  0.7868393  0.7020499  0.7393690
##
## F was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.6497426.
##
## $f1_val
## [1] 0.7272727
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction yes  no
##        yes  28   8
##        no   13 206
##
##                Accuracy : 0.9176
##                  95% CI : (0.8769, 0.9483)
##     No Information Rate : 0.8392
##     P-Value [Acc > NIR] : 0.0001726
##
##                   Kappa : 0.679
##
##  Mcnemar's Test P-Value : 0.3827331
##
##             Sensitivity : 0.6829
##             Specificity : 0.9626
```

```
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.9406
##              Prevalence : 0.1608
##          Detection Rate : 0.1098
##    Detection Prevalence : 0.1412
##       Balanced Accuracy : 0.8228
##
##        'Positive' Class : yes
##
##
##
##
##
##
##
##
## $model
## k-Nearest Neighbors
##
## 1027 samples
##   17 predictor
##    2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 822, 822, 821, 821, 822, 821, ...
## Resampling results across tuning parameters:
##
##   k   AUC        Precision  Recall     F
##    2  0.1157209  0.7954497  0.8901961  0.8386785
##    3  0.2231447  0.7830693  0.8990196  0.8346801
##    4  0.2615624  0.7850426  0.8661319  0.8205955
##    5  0.3145425  0.7977169  0.8810160  0.8346081
##   10  0.5182110  0.8056070  0.8723708  0.8359043
##
## 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] 0.7764706
##
## $confm
## Confusion Matrix and Statistics
##
##           Reference
## Prediction yes  no
##        yes  33  12
##        no    8 202
##
##                Accuracy : 0.9216
##                  95% CI : (0.8815, 0.9514)
##     No Information Rate : 0.8392
##     P-Value [Acc > NIR] : 7.737e-05
##
```

```
##                  Kappa : 0.7204
##
##  Mcnemar's Test P-Value : 0.5023
##
##             Sensitivity : 0.8049
##             Specificity : 0.9439
##          Pos Pred Value : 0.7333
##          Neg Pred Value : 0.9619
##              Prevalence : 0.1608
##          Detection Rate : 0.1294
##    Detection Prevalence : 0.1765
##       Balanced Accuracy : 0.8744
##
##        'Positive' Class : yes
##
##
##
##
##
##
##
##
## $model
## Random Forest
##
## 1027 samples
##    8 predictor
##    2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 822, 822, 821, 821, 822, 821, ...
## Resampling results across tuning parameters:
##
##   mtry  AUC        Precision  Recall       F
##    1    0.8860352  1.0000000  0.008823529  0.1621622
##    2    0.9313423  0.9419988  0.819073084  0.8738309
##    3    0.9286554  0.9485249  0.943404635  0.9449030
##    4    0.9301020  0.9506899  0.985294118  0.9669327
##    5    0.9236188  0.9534109  0.982174688  0.9667989
##    7    0.9286897  0.9500323  0.970320856  0.9591579
##    8    0.9295856  0.9474927  0.970320856  0.9578329
##   10    0.9112520  0.9412639  0.964349376  0.9520077
##   12    0.8806312  0.9408671  0.958467023  0.9489849
##   16    0.8117514  0.9385866  0.955525847  0.9461528
##   17    0.8473012  0.9342303  0.955525847  0.9437392
##
## F was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
##
## $f1_val
## [1] 0.925
##
## $confm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction yes  no
##        yes  37   2
##        no    4 212
##
##                Accuracy : 0.9765
##                  95% CI : (0.9495, 0.9913)
##     No Information Rate : 0.8392
##     P-Value [Acc > NIR] : 7.85e-13
##
##                   Kappa : 0.9111
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9024
##             Specificity : 0.9907
##          Pos Pred Value : 0.9487
##          Neg Pred Value : 0.9815
##              Prevalence : 0.1608
##          Detection Rate : 0.1451
##    Detection Prevalence : 0.1529
##       Balanced Accuracy : 0.9465
##
##        'Positive' Class : yes
##
##
##
##
##
```

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