ADS-503 Team 4 Final Project

Leonard Littleton Lina Nguyen Emanuel Lucban

6/13/21

```
library(readr)
library(data.table)
library(mlbench)
library(ggplot2)
library(tidyr)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(e1071)
library(caret)
```

```
## Loading required package: lattice
```

```
library(naniar)
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'

## The following objects are masked from 'package:caret':
##
##     MAE, RMSE

## The following object is masked from 'package:base':
##
##     Recall
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(MASS)
```

```
##
```

```
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# Parallel Processing
library(doParallel)
```

```
## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel
```

```
cl <- makePSOCKcluster(8)
```

Synthetic Financial Datasets For Fraud Detection

Data Preparation

```
synth_df <- fread('./data/PS_20174392719_1491204439457_log.csv', header = TRUE)
synth_df <- subset(synth_df, select = -c(isFlaggedFraud, nameOrig, nameDest))

synth_df <- cbind(synth_df, data.frame(hours_intraday = synth_df$step %% 24,
                                       by_day = round(synth_df$step / 24),
                                       day_of_week = round(synth_df$step / 24) %% 7))

# Log transform  (amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest)
cont_vars <- c('amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest')

# add small constant to prevent inf values
log_scaled <- sapply(data.frame(synth_df)[, cont_vars], function(x) log(x + 1))
colnames(log_scaled) <- lapply(cont_vars, function(x) paste('log_', x, sep=''))


synth_df <- cbind(synth_df, log_scaled)
synth_df$type <- as.factor(synth_df$type)
dmy <- dummyVars(" ~ type", data = synth_df, sep = '.', fullRank = TRUE)
synth_df <- cbind(synth_df, data.frame(predict(dmy, newdata = synth_df)))
# drop type
synth_df <- subset(synth_df, select = -c(type))
```

Split Data into Training and Test Datasets using Stratified Random Sampling

```
set.seed(42)
```

```r
# split x and y
x <- subset(synth_df, select = -c(isFraud))
y <- synth_df$isFraud

data_part <- createDataPartition(y = y, p = 0.75, list = FALSE)

x_train <- x[data_part, ]
y_train <- y[data_part]
x_test <- x[-data_part, ]
y_test <- y[-data_part]

y_train <- as.factor(y_train)
y_test <- as.factor(y_test)
levels(y_train) <- c('no', 'yes')
levels(y_test) <- c('no', 'yes')
```

Modeling

```r
# SET TRUE TO TRAIN MODELS
TRAIN = FALSE
```

```r
# train control
ctrl <- trainControl(method = "cv",
                     number = 5,
                     summaryFunction = prSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

# Register Parallel Processing
registerDoParallel(cl)
```

Neural Network

Training

```r
if (TRAIN){
  nnTuning <- expand.grid(size = c(1:5),
                                decay = c(0.01, 0.05, 0.1))
  nnetModel <- train(x_train, y_train,
                     method = "nnet",
                     metric = 'AUC',
                     trControl= ctrl,
                     tuneGrid = nnTuning,
                     preProcess=c("scale","center"))

  saveRDS(nnetModel, "rds_files/nnet_model.rds")
} else {
  nnetModel <- readRDS('rds_files/nnet_model.rds')
}
```

```r
nnetModel
```

```
## Neural Network
##
```

```
## 4771965 samples
##      18 predictor
##       2 classes: 'no', 'yes'
##
## Pre-processing: scaled (18), centered (18)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3817572, 3817571, 3817572, 3817572, 3817573
## Resampling results across tuning parameters:
##
##   size  decay  AUC         Precision  Recall     F
##   1     0.01   0.03841136  0.9991485  0.9998995  0.9995237
##   1     0.05   0.08480826  0.9995132  0.9999060  0.9997095
##   1     0.10   0.05056086  0.9991267  0.9999748  0.9995505
##   2     0.01   0.00000000  0.9987188  1.0000000  0.9993590
##   2     0.05   0.46270579  0.9993290  0.9999692  0.9996489
##   2     0.10   0.20514610  0.9991269  0.9999287  0.9995275
##   3     0.01   0.15922016  0.9987188  1.0000000  0.9993590
##   3     0.05   0.11514449  0.9989077  0.9999973  0.9994522
##   3     0.10   0.07911778  0.9989279  0.9999914  0.9994593
##   4     0.01   0.00000000  0.9987188  1.0000000  0.9993590
##   4     0.05   0.03219574  0.9987188  1.0000000  0.9993590
##   4     0.10   0.00000000  0.9987188  1.0000000  0.9993590
##   5     0.01   0.00000000  0.9987188  1.0000000  0.9993590
##   5     0.05   0.10122763  0.9988911  0.9999713  0.9994309
##   5     0.10   0.00000000  0.9987188  1.0000000  0.9993590
##
## AUC was used to select the optimal model using the largest value.
## The final values used for the model were size = 2 and decay = 0.05.
```

Neural Network Prediction on Test Data

```
nnetPred <- predict(nnetModel, newdata = x_test)
nnetCFM <- confusionMatrix(nnetPred, y_test, positive = 'yes')
nnetCFM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      no     yes
##        no  1588556    2099
##        yes       0       0
##
##                Accuracy : 0.9987
##                  95% CI : (0.9986, 0.9987)
##     No Information Rate : 0.9987
##     P-Value [Acc > NIR] : 0.5058
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.99868
##               Prevalence : 0.00132
##           Detection Rate : 0.00000
##     Detection Prevalence : 0.00000
##        Balanced Accuracy : 0.50000
##
##         'Positive' Class : yes
##
```

Neural Network Variable Importance

```
varImp(nnetModel)
```

```
## nnet variable importance
##
##                     Overall
## newbalanceOrig     100.000
## oldbalanceOrg       94.576
## type.DEBIT          55.842
## oldbalanceDest      45.186
## log_newbalanceOrig  43.786
## log_oldbalanceDest  41.047
## type.CASH_OUT       39.638
## log_oldbalanceOrg   32.570
## log_newbalanceDest  31.992
## type.TRANSFER       26.864
## type.PAYMENT        25.894
## newbalanceDest      23.269
## log_amount          20.260
## amount              19.157
## day_of_week         18.376
## by_day              13.672
## step                 5.901
## hours_intraday       0.000
```

Linear Discriminant Analysis

Training

```
if (TRAIN) {
  ldaModel <- train(x_train, y_train,
                    method = "lda",
                    metric = 'AUC',
                    trControl= ctrl,
                    preProcess=c("scale","center"),
                    tuneLength = 10,
                    verbose = FALSE, trace = FALSE)

  saveRDS(ldaModel, "rds_files/lda_model.rds")
} else {
  ldaModel <- readRDS('rds_files/lda_model.rds')
}
```

```
ldaModel
```

```
## Linear Discriminant Analysis
```

```
##
## 4771965 samples
##      18 predictor
##       2 classes: 'no', 'yes'
##
## Pre-processing: scaled (18), centered (18)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3817572, 3817571, 3817572, 3817572, 3817573
## Resampling results:
##
##   AUC           Precision  Recall     F
##   0.0005841567  0.9992177  0.9999041  0.9995608
```

LDA Prediction on Test Data

```
ldaPred <- predict(ldaModel, newdata = x_test)
ldaCFM <- confusionMatrix(ldaPred, y_test, positive = 'yes')
ldaCFM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      no     yes
##        no  1588403    1253
##        yes     153     846
##
##                Accuracy : 0.9991
##                  95% CI : (0.9991, 0.9992)
##     No Information Rate : 0.9987
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5458
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4030491
##             Specificity : 0.9999037
##          Pos Pred Value : 0.8468468
##          Neg Pred Value : 0.9992118
##              Prevalence : 0.0013196
##          Detection Rate : 0.0005319
##    Detection Prevalence : 0.0006280
##       Balanced Accuracy : 0.7014764
##
##        'Positive' Class : yes
##
```

LDA Variable Importance

```
varImp(ldaModel)
```

```
## ROC curve variable importance
##
##                     Importance
## log_oldbalanceOrg       100.00
## oldbalanceOrg           100.00
## log_amount               93.07
## amount                   93.07
## type.TRANSFER            66.00
## newbalanceOrig           65.53
## log_newbalanceOrig       65.53
## type.PAYMENT             54.10
## step                     54.00
## by_day                   52.76
## hours_intraday           49.38
## log_oldbalanceDest       41.67
## oldbalanceDest           41.67
## type.CASH_OUT            23.98
## day_of_week              15.74
## newbalanceDest           11.49
## log_newbalanceDest       11.49
## type.DEBIT                0.00
```

Quadratic Discriminant Analysis

Training

```
if (TRAIN) {
  qdaModel <- train(x_train, y_train,
                    method = "qda",
                    metric = 'AUC',
                    trControl= ctrl,
                    preProcess=c("pca", "scale","center"),
                    tuneLength = 10,
                    verbose = FALSE, trace = FALSE)
  saveRDS(qdaModel, "rds_files/qda_model.rds")
} else {
  qdaModel <- readRDS('rds_files/qda_model.rds')
}
```

```
qdaModel
```

```
## Quadratic Discriminant Analysis
##
## 4771965 samples
##      18 predictor
##       2 classes: 'no', 'yes'
##
## Pre-processing: principal component signal extraction (18), scaled
##  (18), centered (18)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3817572, 3817571, 3817572, 3817572, 3817573
## Resampling results:
##
##   AUC        Precision  Recall     F
##   0.2011124  0.9997956  0.9934996  0.9966375
```

QDA Prediction on Test Data

```
qdaPred <- predict(qdaModel, newdata = x_test)
qdaCFM <- confusionMatrix(qdaPred, y_test, positive = 'yes')
qdaCFM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      no     yes
##        no  1578607     295
##        yes    9949    1804
##
##                Accuracy : 0.9936
##                  95% CI : (0.9934, 0.9937)
##     No Information Rate : 0.9987
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2588
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.859457
##             Specificity : 0.993737
##          Pos Pred Value : 0.153493
##          Neg Pred Value : 0.999813
##              Prevalence : 0.001320
##          Detection Rate : 0.001134
##    Detection Prevalence : 0.007389
##       Balanced Accuracy : 0.926597
##
##        'Positive' Class : yes
##
```

QDA Variable Importance

```
varImp(qdaModel)
```

```
## ROC curve variable importance
##
##                    Importance
## log_oldbalanceOrg      100.00
## oldbalanceOrg          100.00
## log_amount              93.07
## amount                  93.07
## type.TRANSFER           66.00
## newbalanceOrig          65.53
## log_newbalanceOrig      65.53
## type.PAYMENT            54.10
## step                    54.00
## by_day                  52.76
## hours_intraday          49.38
## log_oldbalanceDest      41.67
## oldbalanceDest          41.67
## type.CASH_OUT           23.98
## day_of_week             15.74
## newbalanceDest          11.49
## log_newbalanceDest      11.49
## type.DEBIT               0.00
```

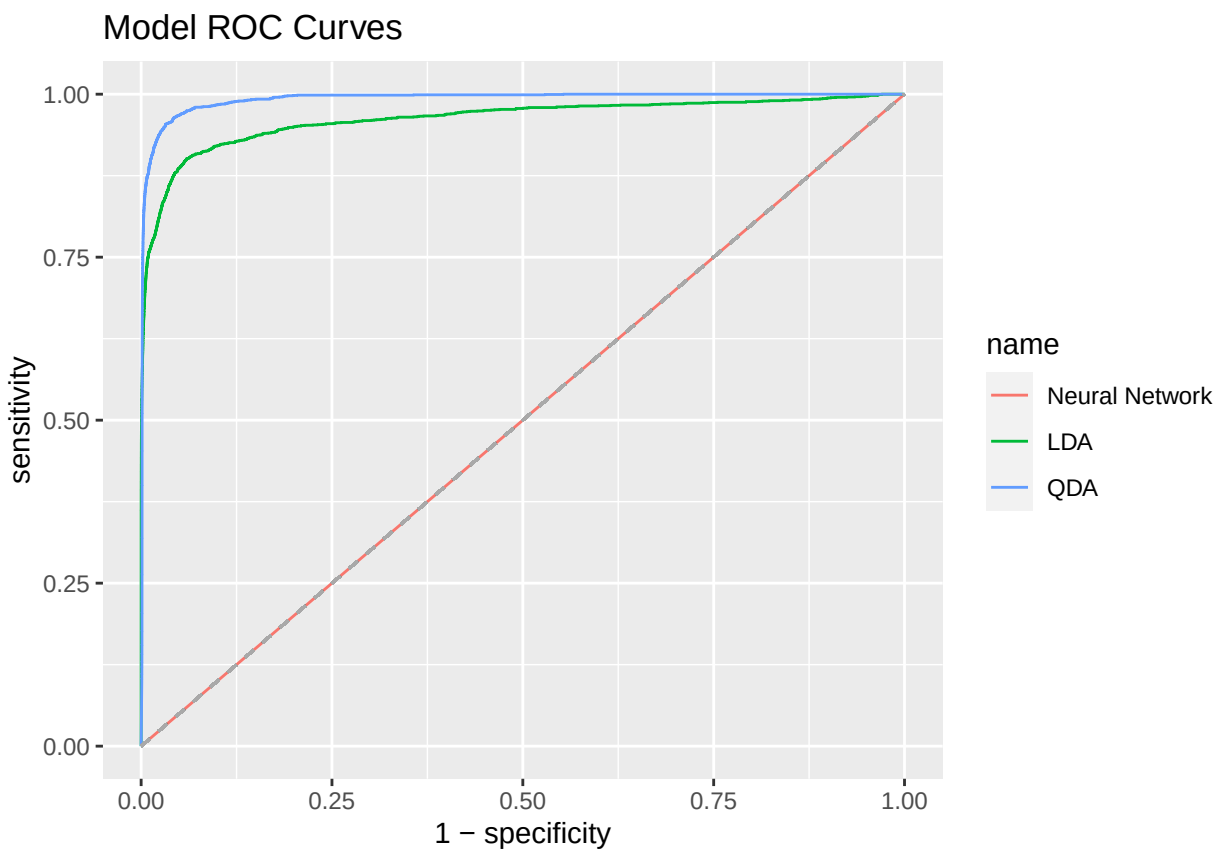
```
# Stop cluster and parallel processing
stopCluster(cl)
registerDoSEQ()
```

ROC Plots

```
roc_list <- list("Neural Network" = roc(y_test, predict(nnetModel, newdata = x_test, type = 'prob')$yes
                 "LDA" = roc(y_test, predict(ldaModel, newdata = x_test, type = 'prob')$yes),
                 "QDA" = roc(y_test, predict(qdaModel, newdata = x_test, type = 'prob')$yes))
```

```
## Setting levels: control = no, case = yes

## Setting direction: controls < cases

## Setting levels: control = no, case = yes

## Setting direction: controls < cases

## Setting levels: control = no, case = yes

## Setting direction: controls < cases
```

```
ggroc(roc_list, legacy.axes = TRUE) + scale_linetype_discrete() + ggtitle("Model ROC Curves") +
  geom_segment(aes(x = 0, xend = 1, y = 0, yend = 1),
               color="darkgrey", linetype="dashed")
```


Model ROC Curves
sensitivity
1 − specificity
name
Neural Network
LDA
QDA