

ADS-503 Team 4 Modeling

Leonard Littleton

Lina Nguyen

Emanuel Lucban

6/25/21

```
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(5)
```

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)
```

```

# 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

```

# SET TRUE TO TRAIN MODELS
TRAIN = FALSE

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

# Register Parallel Processing
registerDoParallel(cl)

```

Neural Network

Training

```

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')
}

```

```
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 <- readRDS('rds_files/nnet_pred.rds')
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
## oldbalanceOrig     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 <- readRDS('rds_files/lda_pred.rds')
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 <- readRDS('rds_files/qda_pred.rds')
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
```

Logistic Regression

Training

```
if (TRAIN) {  
  ctrl <- trainControl(summaryFunction = prSummary, classProbs = TRUE)  
  lrModel <- train(x_train, y= y_train, method = 'glm',  
                  preProcess = c('center', 'scale'),  
                  metric = 'AUC', trControl = ctrl)  
  saveRDS(lrModel, "rds_files/lr_model.rds")  
} else {  
  lrModel <- readRDS('rds_files/lr_model.rds')  
}
```

lrModel

```
## Generalized Linear Model  
##  
## 4771965 samples  
##      18 predictor  
##      2 classes: 'no', 'yes'  
##  
## Pre-processing: centered (18), scaled (18)  
## Resampling: Bootstrapped (25 reps)  
## Summary of sample sizes: 4771965, 4771965, 4771965, 4771965, 4771965, ...  
## Resampling results:  
##  
##      AUC      Precision  Recall      F  
## 0.1244624 0.9993675 0.9999007 0.999634
```

Logistic Regression Prediction on Test Data

```
lrPred <- readRDS('rds_files/lr_pred.rds')  
lrCFM <- confusionMatrix(lrPred, y_test, positive = 'yes')  
lrCFM
```

```
## Confusion Matrix and Statistics  
##  
##           Reference  
## Prediction      no      yes  
##           no 1588485      565  
##           yes    71     1534  
##  
##           Accuracy : 0.9996  
##           95% CI : (0.9996, 0.9996)  
##           No Information Rate : 0.9987  
##           P-Value [Acc > NIR] : < 2.2e-16  
##  
##           Kappa : 0.8281  
##  
##           Mcnemar's Test P-Value : < 2.2e-16  
##  
##           Sensitivity : 0.7308242  
##           Specificity : 0.9999553  
##           Pos Pred Value : 0.9557632
```

```
##          Neg Pred Value : 0.9996444
##          Prevalence : 0.0013196
##          Detection Rate : 0.0009644
##          Detection Prevalence : 0.0010090
##          Balanced Accuracy : 0.8653898
##
##          'Positive' Class : yes
##
```

Logistic Regression Variable Importance

```
varImp(lrModel)
```

```
## glm variable importance
##
##          Overall
## log_newbalanceDest 1.000e+02
## log_amount         7.269e+01
## log_oldbalanceOrg  6.838e+01
## log_newbalanceOrig 6.382e+01
## hours_intraday     6.361e+01
## oldbalanceOrg       5.883e+01
## newbalanceOrig      5.532e+01
## amount              2.922e+01
## day_of_week         1.648e+01
## oldbalanceDest      1.479e+01
## newbalanceDest      1.457e+01
## log_oldbalanceDest  5.659e+00
## step                3.498e+00
## type.TRANSFER        6.534e-01
## type.CASH_OUT        6.514e-01
## type.PAYMENT         1.366e-01
## type.DEBIT           7.362e-03
## by_day               0.000e+00
```

Partial Least Squares Discriminant Analysis

Training

```
if (TRAIN) {
  plsdaModel <- train(x=x_train, y = y_train,
    preProcess = c("center", "scale"),
    method = "pls", metric = "AUC",
    trControl = ctrl,
    tuneGrid = expand.grid(.ncomp = 1:5),
    verbose = FALSE, trace = FALSE)
  saveRDS(plsdaModel, 'rds_files/plsda_model.rds')
} else {
  plsdaModel <- readRDS('rds_files/plsda_model.rds')
}
```

```
plsdaModel
```

```
## Partial Least Squares
##
```

```
## 4771965 samples
##      18 predictor
##      2 classes: 'no', 'yes'
##
## Pre-processing: centered (18), scaled (18)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3817572, 3817572, 3817572, 3817572, 3817572
## Resampling results across tuning parameters:
##
##  ncomp  AUC          Precision Recall  F
##  1      0.9998824  0.9987188  1      0.999359
##  2      0.9998644  0.9987188  1      0.999359
##  3      0.9999250  0.9987188  1      0.999359
##  4      0.9999585  0.9987188  1      0.999359
##  5      0.9999605  0.9987188  1      0.999359
##
## AUC was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 5.
```

PLSDA on Test Data

```
plsdaPred <- readRDS('rds_files/plsda_pred.rds')
plsdaCFM <- confusionMatrix(plsdaPred, y_test, positive = 'yes')
plsdaCFM
```

```
## 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
##
```

PLSDA Variable Importance

```

varImp(plsdaModel)

##
## Attaching package: 'pls'
## The following object is masked from 'package:caret':
##
##      R2
## The following object is masked from 'package:corrplot':
##
##      corrplot
## The following object is masked from 'package:stats':
##
##      loadings
## pls variable importance
##
##              Overall
## amount          100.000
## log_oldbalanceOrg 78.252
## type.TRANSFER     61.690
## log_newbalanceOrig 55.866
## log_newbalanceDest 49.261
## type.PAYMENT      47.320
## log_amount        44.899
## hours_intraday    42.615
## step              40.622
## log_oldbalanceDest 40.594
## by_day            39.698
## oldbalanceOrg     35.692
## type.CASH_OUT     33.679
## oldbalanceDest    22.278
## newbalanceOrig    14.196
## newbalanceDest     6.730
## type.DEBIT        2.192
## day_of_week       0.000

```

Support Vector Machines

Training

```

if (TRAIN) {
  svmRdcd <- sigest(as.matrix(x_train))
  svmTuning <- expand.grid(.sigma = svmRdcd[1], .C = 1)
  svmModel <- train(x_train, y= y_train,
                    method = 'svmRadial',
                    preProcess = c('center', 'scale'),
                    metric = 'AUC',
                    trControl = ctrl,
                    tuneGrid = svmTuning,
                    fit = FALSE)
  saveRDS(svmModel, 'rds_files/svm_model.rds')
} else {

```

```
svmModel <- readRDS('rds_files/svm_model.rds')
}
```

```
svmModel
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 4771965 samples
##      18 predictor
##      2 classes: 'no', 'yes'
##
## Pre-processing: scaled (18), centered (18)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3817572, 3817572, 3817572, 3817572, 3817572
## Resampling results:
##
##      ROC          Sens          Spec
## 0.9956985 0.9999631 0.7440299
##
## Tuning parameter 'sigma' was held constant at a value of 0.01837413
##
## Tuning parameter 'C' was held constant at a value of 1
```

SVM on Test Data

```
svmPred <- readRDS('rds_files/svm_pred.rds')
svmCFM <- confusionMatrix(svmPred, y_test, positive = 'yes')
svmCFM
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction      no      yes
##      no 1588494      500
##      yes    62     1599
##
##              Accuracy : 0.9996
##              95% CI : (0.9996, 0.9997)
##      No Information Rate : 0.9987
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.8504
##
##      McNemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.761791
##              Specificity : 0.999961
##              Pos Pred Value : 0.962673
##              Neg Pred Value : 0.999685
##              Prevalence : 0.001320
##              Detection Rate : 0.001005
##      Detection Prevalence : 0.001044
##              Balanced Accuracy : 0.880876
##
##      'Positive' Class : yes
##
```

SVM Variable Importance

```
varImp(svmModel)
```

```
## ROC curve variable importance
##
##              Importance
## oldbalanceOrg      100.00
## log_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
```

Nearest Shrunk Centroids

Training

```
if (TRAIN) {
  nscGrid <- data.frame(.threshold = 0:15)
  nscBio <- train(x=x_train, y = y_train,
                 preProcess = c('center', 'scale'),
                 method = "pam",
                 metric = "AUC",
                 trControl = ctrl,
                 tuneGrid = nscGrid)
  saveRDS(nscModel, 'rds_files/nsc_model.rds')
} else {
  nscModel <- readRDS('rds_files/nsc_model.rds')
}
```

```
nscModel
```

```
## Nearest Shrunk Centroids
##
## 4771965 samples
##      18 predictor
##      2 classes: 'no', 'yes'
##
## Pre-processing: scaled (18), centered (18)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3817572, 3817573, 3817571, 3817572, 3817572
## Resampling results across tuning parameters:
```

```
##
## threshold ROC Sens Spec
## 0 0.9323871 0.9987912 0.06934934
## 1 0.9317410 0.9988023 0.06885847
## 2 0.9310041 0.9988073 0.06738601
## 3 0.9301870 0.9988122 0.06526009
## 4 0.9292877 0.9988178 0.06411537
## 5 0.9282892 0.9988246 0.06280671
## 6 0.9272498 0.9988323 0.06100758
## 7 0.9262361 0.9988430 0.05904506
## 8 0.9252065 0.9988539 0.05708241
## 9 0.9242789 0.9988726 0.05593755
## 10 0.9232327 0.9988883 0.05381123
## 11 0.9220452 0.9989064 0.05135784
## 12 0.9210972 0.9989223 0.05004931
## 13 0.9203243 0.9989395 0.04841372
## 14 0.9194647 0.9989557 0.04759579
## 15 0.9185028 0.9989775 0.04514241
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was threshold = 0.
```

NSC on Test Data

```
nscPred <- readRDS('rds_files/nsc_pred.rds')
nscCFM <- confusionMatrix(nscPred, y_test, positive = 'yes')
nscCFM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no    yes
##           no 1586646 1947
##           yes  1910   152
##
##           Accuracy : 0.9976
##           95% CI : (0.9975, 0.9977)
##           No Information Rate : 0.9987
##           P-Value [Acc > NIR] : 1.0000
##
##           Kappa : 0.0718
##
## Mcnemar's Test P-Value : 0.5621
##
##           Sensitivity : 7.242e-02
##           Specificity : 9.988e-01
##           Pos Pred Value : 7.371e-02
##           Neg Pred Value : 9.988e-01
##           Prevalence : 1.320e-03
##           Detection Rate : 9.556e-05
##           Detection Prevalence : 1.296e-03
##           Balanced Accuracy : 5.356e-01
##
##           'Positive' Class : yes
##
```

NSC Variable Importance

```
varImp(nscModel)
```

```
## pam variable importance
##
##               Importance
## amount          0.127151
## type.TRANSFER    0.088481
## log_amount       0.066681
## log_oldbalanceOrg 0.056623
## step            0.051994
## by_day           0.050711
## hours_intraday   0.050355
## log_newbalanceOrig 0.047248
## type.PAYMENT     0.041969
## log_oldbalanceDest 0.027505
## type.CASH_OUT    0.018248
## oldbalanceOrg    0.015725
## newbalanceOrig   0.012498
## log_newbalanceDest 0.011205
## day_of_week      0.008764
## oldbalanceDest   0.008650
## type.DEBIT       0.003743
## newbalanceDest   0.000000
```

Gradient Boosting Model

Training

```
if (TRAIN) {
  gbmModel <- train(x=x_train, y = y_train,
                    preProcess = c('center', 'scale'),
                    method = "pam",
                    metric = "AUC",
                    trControl = ctrl,
                    tuneGrid = nscGrid)
  saveRDS(gbmModel, 'rds_files/gbm_model.rds')
} else {
  gbmModel <- readRDS('rds_files/gbm_model.rds')
}
```

```
gbmModel
```

```
## Stochastic Gradient Boosting
##
## 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, 3817572, 3817572, 3817572, 3817572
```



```
## Resampling results:
##
##      ROC      Sens      Spec
##  0.975896  0.9999442  0.6234864
##
## Tuning parameter 'n.trees' was held constant at a value of 500
## Tuning
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.01
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 5
```

GBM Prediction on Test Data

```
gbmPred <- readRDS('rds_files/gbm_pred.rds')
gbmCFM <- confusionMatrix(gbmPred, y_test, positive = 'yes')
gbmCFM
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction      no      yes
##           no 1588471      781
##           yes    85     1318
##
##              Accuracy : 0.9995
##              95% CI : (0.9994, 0.9995)
##      No Information Rate : 0.9987
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.7525
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.6279181
##              Specificity : 0.9999465
##              Pos Pred Value : 0.9394155
##              Neg Pred Value : 0.9995086
##              Prevalence : 0.0013196
##              Detection Rate : 0.0008286
##      Detection Prevalence : 0.0008820
##              Balanced Accuracy : 0.8139323
##
##              'Positive' Class : yes
##
```

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

Model Metrics on Test Data

```
# ROC values
nnetROC <- roc(y_test, readRDS('rds_files/nnet_prob.rds')$yes)

## Setting levels: control = no, case = yes
## Setting direction: controls < cases
ldaROC <- roc(y_test, readRDS('rds_files/lda_prob.rds')$yes)

## Setting levels: control = no, case = yes
## Setting direction: controls < cases
qdaROC <- roc(y_test, readRDS('rds_files/qda_prob.rds')$yes)

## Setting levels: control = no, case = yes
## Setting direction: controls < cases
lrROC <- roc(y_test, readRDS('rds_files/lr_prob.rds')$yes)

## Setting levels: control = no, case = yes
## Setting direction: controls < cases
plsdaROC <- roc(y_test, readRDS('rds_files/plsda_prob.rds')$yes)

## Setting levels: control = no, case = yes
## Setting direction: controls < cases
svmROC <- roc(y_test, readRDS('rds_files/svm_prob.rds')$yes)

## Setting levels: control = no, case = yes
## Setting direction: controls < cases
nscROC <- roc(y_test, readRDS('rds_files/nsc_prob.rds')$yes)

## Setting levels: control = no, case = yes
## Setting direction: controls < cases
gbmROC <- roc(y_test, readRDS('rds_files/gbm_prob.rds')$yes)

## Setting levels: control = no, case = yes
## Setting direction: controls < cases

#AUC
nnetAUC <- auc(nnetROC)
ldaAUC <- auc(ldaROC)
qdaAUC <- auc(qdaROC)
lrAUC <- auc(lrROC)
plsdaAUC <- auc(plsdaROC)
svmAUC <- auc(svmROC)
nscAUC <- auc(nscROC)
gbmAUC <- auc(gbmROC)

model_names <- c('Neural Network', 'Linear Discriminant Analysis', 'Quadratic Discriminant Analysis',
                 'PLS Discriminant Analysis', 'Logistic Regression', 'Support Vector Machines',
                 'Nearest Shrunken Centroids', 'Gradient Boosting Machine')

metricsdf <- rbind(nnetCFM$byClass, ldaCFM$byClass)
metricsdf <- rbind(metricsdf, qdaCFM$byClass)
metricsdf <- rbind(metricsdf, plsdaCFM$byClass)
```

```

metricsdf <- rbind(metricsdf, lrCFM$byClass)
metricsdf <- rbind(metricsdf, svmCFM$byClass)
metricsdf <- rbind(metricsdf, nscCFM$byClass)
metricsdf <- rbind(metricsdf, gbmCFM$byClass)

metricsdf <- cbind(data.frame("Models" = model_names, "AUC" = c(nnetAUC, ldaAUC, qdaAUC,
                                                                plsdaAUC, lrAUC, svmAUC,
                                                                nscAUC, gbmAUC)),
                  data.frame(metricsdf))
metricsdf[order(-metricsdf$Sensitivity), ]

```

##	Models	AUC	Sensitivity	Specificity		
## 3	Quadratic Discriminant Analysis	0.9927164	0.85945688	0.9937371		
## 6	Support Vector Machines	0.9978020	0.76179133	0.9999610		
## 5	Logistic Regression	0.9964422	0.73082420	0.9999553		
## 8	Gradient Boosting Machine	0.9777450	0.62791806	0.9999465		
## 2	Linear Discriminant Analysis	0.9621525	0.40304907	0.9999037		
## 7	Nearest Shrunk Centroids	0.9362662	0.07241544	0.9987977		
## 1	Neural Network	0.5000000	0.00000000	1.0000000		
## 4	PLS Discriminant Analysis	0.9771492	0.00000000	1.0000000		
##	Pos.Pred.Value	Neg.Pred.Value	Precision	Recall	F1	Prevalence
## 3	0.15349273	0.9998132	0.15349273	0.85945688	0.26046780	0.001319582
## 6	0.96267309	0.9996853	0.96267309	0.76179133	0.85053191	0.001319582
## 5	0.95576324	0.9996444	0.95576324	0.73082420	0.82829374	0.001319582
## 8	0.93941554	0.9995086	0.93941554	0.62791806	0.75271274	0.001319582
## 2	0.84684685	0.9992118	0.84684685	0.40304907	0.54615881	0.001319582
## 7	0.07371484	0.9987744	0.07371484	0.07241544	0.07305936	0.001319582
## 1	NaN	0.9986804	NA	0.00000000	NA	0.001319582
## 4	NaN	0.9986804	NA	0.00000000	NA	0.001319582
##	Detection.Rate	Detection.Prevalence	Balanced.Accuracy			
## 3	1.134124e-03	0.0073887801	0.9265970			
## 6	1.005246e-03	0.0010442239	0.8808762			
## 5	9.643826e-04	0.0010090183	0.8653898			
## 8	8.285895e-04	0.0008820266	0.8139323			
## 2	5.318564e-04	0.0006280432	0.7014764			
## 7	9.555812e-05	0.0012963213	0.5356065			
## 1	0.000000e+00	0.0000000000	0.5000000			
## 4	0.000000e+00	0.0000000000	0.5000000			

ROC Plots

```

# Plot takes a long time to complete
roc_list <- list("Neural Network" = nnetROC,
                "LDA" = ldaROC,
                "QDA" = qdaROC,
                "PLS Discriminant Analysis" = plsdaROC,
                "Logistic Regression" = lrROC,
                "Support Vector Machines" = svmROC,
                "Nearest Shrunk Centroids" = nscROC,
                "Gradient Boosting Machine" = gbmROC)

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")
```

