Aggregated Data Script (ML Technique Comparisons)

Paula McMahon

August 2019

```
knitr::opts_chunk$set(echo = TRUE)
library(readxl)
library(corrplot)
library(tree)
library(randomForest)
library(e1071)
library(MASS)
library(moments)
library(rpart)
library(rpart.plot)
library(class)
library(caret)
library(gbm)
require(devtools)
require(ggbiplot)
```

Read in data

```
# Read in DORMANT customer data
dormant_customers <-
  read_excel("~/college/ST606_Project/data_files/dormant-transactions.xlsx")

# Read in CURRENT customer data
current_customers <-
  read_excel("~/college/ST606_Project/data_files/stayed-shopping-transactions.xlsx")
```

Merge current and dormant datasets

```
# Add a variable "Churn" to the dormant dataset and set it equal to 1
# i.e. the customer has stopped shopping.
# Add a variable "Churn" to the current dataset and set it equal to 0
# i.e. the customer is a current shopper.
current_customers$Churn = 0
dormant_customers$Churn = 1

# Not all of the variables in the dataset will be used:
# OrderRef is a unique reference number and is not of any value.
# ExpectedGoodsCharge is 98% correlated with ActualCharge, so drop
# ExpectedGoodsCharge from analysis.
# OverallSubstitutionPolicy is a categorical variable and not valuable
# for aggregated analysis.
# TotalOrderLines is an extra variable in current dataset but not dormant,
# so drop from analysis.
# TotalItemsApprovedPicks is 100% correlated with TotalPickedLines so
```

```r
# drop TotalItemsApprovedPicks from analysis
# TotalQtyOrdered is 99% correlated with TotalOrderItems so drop
# TotalQtyOrdered from analysis.
# AvailabilityPostSubPercentage is 99.9% correlated with
# AvailabilityPreSubPercentage so drop AvailabilityPostSubPercentage from analysis.

merged_customers <-
  rbind(dormant_customers[c("sequenceID","StoreID", "ExpectedFulfillmentCharge",
                            "SlotStartDate","ActualCharge","TotalOrderItems",
                            "TotalPickedLines","TotalQtySubbed","TotalQtyOOS",
                            "TotalPickTimeSeconds","AvailabilityPreSubPercentage",
                            "PercentageOutOfStocks","PercentageSubstitutions",
                            "ValueOfSubstitutions","ValueOfOutOfStocks",
                            "RelatedCallsCount","Churn")],
        current_customers[c("sequenceID","StoreID","ExpectedFulfillmentCharge",
                            "SlotStartDate","ActualCharge","TotalOrderItems",
                            "TotalPickedLines","TotalQtySubbed","TotalQtyOOS",
                            "TotalPickTimeSeconds","AvailabilityPreSubPercentage",
                            "PercentageOutOfStocks","PercentageSubstitutions",
                            "ValueOfSubstitutions","ValueOfOutOfStocks",
                            "RelatedCallsCount","Churn")])
```

Correlation Matrix Plots

```r
M <- cor(current_customers[c(4:5,8:23)])
corrplot(M, method="circle", type="upper", diag=FALSE, tl.cex=0.65,
         tl.offset=0.4, tl.srt=45,
         title="Correlation matrix plots showing very high levels of multicollinearity",
         mar = c(0,0,1,0))


# Correlation Matrix Plots of the MERGED data
M <- cor(merged_customers[c(3,5:16)])
corrplot(M, method="circle", type="upper", diag=FALSE, tl.cex=0.65,
         tl.offset=0.4, tl.srt=45,
         title="Correlation matrix plots using the reduced set of variables",
         mar = c(0,0,1,0))
corrplot(M, method="ellipse", type="upper", diag=FALSE, tl.cex=0.65,
         tl.offset=0.4, tl.srt=45,
         title="Correlation matrix plots using the reduced set of variables",
         mar = c(0,0,1,0))
corrplot(M, method="number", type="upper", diag=FALSE, tl.cex=0.65,
         tl.offset=0.4, tl.srt=45, number.cex=0.65,
         title="Correlation matrix plots using the reduced set of variables (%'s)",
         mar = c(0,0,1,0))
```

```r
# Create factors
merged_customers$Churn <- as.factor(merged_customers$Churn)
merged_customers$StoreID <- as.factor(merged_customers$StoreID)
dim(merged_customers)
```

Data Manipulation

Find the number of days between each transaction date and a "base" date. This variable will be called DaysSinceLastShop.

```
# The last shopping date in the dataset is 31/05/2018.
# So use a date after this, as the "basedate"
basedate <- "01/06/18"
basedate <- as.Date(basedate, "%d/%m/%y")

# create a new variable, length = number of rows in dataset
merged_customers$DaysSinceLastShop = numeric(30977)

# subtract each SlotStartDate from the basedate and give the answer in days
for (i in 1:nrow(merged_customers)) merged_customers$DaysSinceLastShop[i] <-
as.vector(difftime(basedate, merged_customers$SlotStartDate[i], units="days"))
# We want days to be an integer value
merged_customers$DaysSinceLastShop <- ceiling(merged_customers$DaysSinceLastShop)
```

```
# In order to run logistic regression, we need to aggregate by customer (sequenceID).
# Use the aggregate function on the merged dataset, by "mean"
# This will only run on the numerical columns, so add back in the Churn column afterwards.

merged_customers_agg =
  aggregate(merged_customers[c("ExpectedFulfillmentCharge","ActualCharge",
                               "TotalOrderItems","TotalPickedLines","TotalQtySubbed",
                               "TotalQtyOOS","TotalPickTimeSeconds",
                               "AvailabilityPreSubPercentage","PercentageOutOfStocks",
                               "PercentageSubstitutions","ValueOfSubstitutions",
                               "ValueOfOutOfStocks","RelatedCallsCount",
                               "DaysSinceLastShop")],
            by=list(merged_customers$sequenceID), FUN=mean)


# Add back in the Churn response variable, 1-1000 (dormant=1), 1001-1905 (current=0)
for (i in 1:1000) merged_customers_agg$Churn[i] = 1
for (i in 1001:1905) merged_customers_agg$Churn[i] = 0
# make it a factor
merged_customers_agg$Churn <- as.factor(merged_customers_agg$Churn)
```

Data Split

```
set.seed(123)
s <- sample(nrow(merged_customers_agg), round(.5*(nrow(merged_customers_agg))))
merged_customers_train <- merged_customers_agg[s,]     # training set
merged_customers_test  <- merged_customers_agg[-s,]    # test set
dim(merged_customers_train)
dim(merged_customers_test)
```

Classification Trees

rpart

```
fitrpart <- rpart(Churn ~ ExpectedFulfillmentCharge+ActualCharge+TotalOrderItems+
                TotalPickedLines+TotalQtySubbed+TotalQtyOOS+TotalPickTimeSeconds+
                AvailabilityPreSubPercentage+PercentageOutOfStocks+
                PercentageSubstitutions+ValueOfSubstitutions+ValueOfOutOfStocks+
                RelatedCallsCount, data=merged_customers_train)

summary(fitrpart)
varImp(fitrpart)
rpart.plot(fitrpart)
plotcp(fitrpart)
printcp(fitrpart)

# To calculate the test MSE use the test set merged_customers_test
pred = predict(fitrpart, merged_customers_test, type="class")
tab <- table(pred, merged_customers_test$Churn)
tab

# The table elements are:
# tab[1]      tab[3]
# tab[2]      tab[4]

# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)

# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

Bagging

```
set.seed(123)
fitbag = randomForest(Churn ~ ExpectedFulfillmentCharge+ActualCharge+TotalOrderItems+
                TotalPickedLines+TotalQtySubbed+TotalQtyOOS+TotalPickTimeSeconds+
                AvailabilityPreSubPercentage+PercentageOutOfStocks+
                PercentageSubstitutions+ValueOfSubstitutions+ValueOfOutOfStocks+
                RelatedCallsCount, data=merged_customers_train,
                mtry=14, importance =TRUE)
fitbag
varImpPlot(fitbag)
importance(fitbag)
```

```
pred = predict(fitbag, merged_customers_test, type="class")
tab <- table(pred, merged_customers_test$Churn)
tab

# The table elements are:
# tab[1]      tab[3]
# tab[2]      tab[4]

# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)

# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

Random Forest

```
fitrf <- randomForest(Churn ~ ExpectedFulfillmentCharge+ActualCharge+TotalOrderItems+
               TotalPickedLines+TotalQtySubbed+TotalQtyOOS+TotalPickTimeSeconds+
               AvailabilityPreSubPercentage+PercentageOutOfStocks+
               PercentageSubstitutions+ValueOfSubstitutions+ValueOfOutOfStocks+
               RelatedCallsCount, data=merged_customers_train,
               mtry=3, importance=TRUE)
summary(fitrf)
varImpPlot(fitrf)
importance(fitrf)

pred = predict(fitrf, merged_customers_test, type="class")
tab <- table(pred, merged_customers_test$Churn)
tab

# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)
```

```
# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

Boosting

```
set.seed(123)
# default shrinkage value = 0.001, use "shrinkage=" to change.
fitboost = gbm(as.character(Churn) ~ ExpectedFulfillmentCharge+ActualCharge+
                 TotalOrderItems+TotalPickedLines+TotalQtySubbed+TotalQtyOOS+
                 TotalPickTimeSeconds+AvailabilityPreSubPercentage+
                 PercentageOutOfStocks+PercentageSubstitutions+
                 ValueOfSubstitutions+ValueOfOutOfStocks+RelatedCallsCount,
               data=merged_customers_train, distribution="bernoulli",
               n.trees=6000,
               interaction.depth=4)

summary(fitboost)
pred = predict(fitboost, newdata=merged_customers_test, n.trees=6000, type="response")
pred <- factor(ifelse(pred < 0.5, 0, 1))

tab <- table(pred, as.character(merged_customers_test$Churn))
tab

mean(pred != as.character(merged_customers_test$Churn))
```

Support Vector Machines

```
set.seed(123)
# use training data
tuneP <- tune.svm(Churn ~ ExpectedFulfillmentCharge+ActualCharge+TotalOrderItems+
               TotalPickedLines+TotalQtySubbed+TotalQtyOOS+TotalPickTimeSeconds+
               AvailabilityPreSubPercentage+PercentageOutOfStocks+
               PercentageSubstitutions+ValueOfSubstitutions+ValueOfOutOfStocks+
               RelatedCallsCount,data = merged_customers_train,
               kernel = "radial", degree = 1:3, cost = c(.01, .1, 1, 5, 10))
tuneP

fit3 <- tuneP$best.model
summary(fit3)

# use test data
pred <- predict(fit3, newdata=merged_customers_test)
tab <- table(pred, merged_customers_test$Churn)
tab

# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])
```

```r
# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)

# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

LDA

```r
fitlda <- lda(Churn ~ ExpectedFulfillmentCharge+ActualCharge+TotalOrderItems+
              TotalPickedLines+TotalQtySubbed+TotalQtyOOS+TotalPickTimeSeconds+
              AvailabilityPreSubPercentage+PercentageOutOfStocks+
              PercentageSubstitutions+ValueOfSubstitutions+ValueOfOutOfStocks+
              RelatedCallsCount, data=merged_customers_train)
fitlda
summary(fitlda)
pred <- predict(fitlda, merged_customers_test)$class
tab <- table(pred, merged_customers_test$Churn)
tab

# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)

# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

QDA

```r
fitqda <- qda(Churn ~ ExpectedFulfillmentCharge+ActualCharge+TotalOrderItems+
              TotalPickedLines+TotalQtySubbed+TotalQtyOOS+TotalPickTimeSeconds+
              AvailabilityPreSubPercentage+PercentageOutOfStocks+
              PercentageSubstitutions+ValueOfSubstitutions+ValueOfOutOfStocks+
              RelatedCallsCount, data=merged_customers_train)
fitqda
summary(fitqda)
pred <- predict(fitqda, merged_customers_test)$class
tab <- table(pred, merged_customers_test$Churn)
tab
```

```r
# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)

# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

KNN

```r
merged_cust_tr_sc <- scale(merged_customers_train[2:14])
merged_cust_te_sc <- scale(merged_customers_test[2:14])
pred <- knn(merged_cust_tr_sc, merged_cust_te_sc[1:952,],
            merged_customers_test$Churn[-953], k=50)
tab <- table(pred, merged_customers_test$Churn[-953])
tab

# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn[-953])

# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

Principal Components Analysis

```r
# Generate a version of DaysSinceLastShop with the "sd" function applied to it.
# This will help to isolate the different types of shoppers.
na_count <-
  tapply(merged_customers$DaysSinceLastShop, merged_customers$sequenceID, sd)

# identify specific groups so they can be colour coded in ggbiplot
pcaGroup <- character(1903)
for (i in 1:nrow(merged_customers_agg[-c(182,579),]))
```

```
# customer has shopped once and is classified as having left
ifelse(is.na(na_count[i]) & merged_customers_agg$Churn[i] == 1,
       pcaGroup[i] <- "ShopOnceChurn",
# customer has shopped once and is classified as having not left
ifelse(is.na(na_count[i]) & merged_customers_agg$Churn[i] == 0,
       pcaGroup[i] <- "ShopOnceNoChurn",
# customer has shopped more than once and is classified as having left
ifelse(!is.na(na_count[i]) & merged_customers_agg$Churn[i] == 1,
       pcaGroup[i] <- "ShopMoreChurn",
# customer has shopped more than once and is classified as having left
pcaGroup[i] <- "ShopMoreNoChurn")))

sum(pcaGroup == "ShopOnceChurn")
sum(pcaGroup == "ShopOnceNoChurn")
sum(pcaGroup == "ShopMoreChurn")
sum(pcaGroup == "ShopMoreNoChurn")
pcaGroup <- as.factor(pcaGroup)

# Currently PCA is running without customers 182 and 579 for now as they have
# very high substitutions/out of stock values. They are non regular transactions
# and running temporarily without them gives a slightly clearer plot.
fitpca <- prcomp(merged_customers_agg[-c(182,579),2:14], scale. = TRUE)
summary(fitpca)
fitpca
biplot(fitpca)

ggbiplot(fitpca, obs.scale = 1, var.scale = 1, group=pcaGroup,
         varname.size = 2, labels.size=1, varname.adjust=1,
         ellipse = TRUE, circle = FALSE, alpha=0) +
  scale_color_discrete(name = '') +
  geom_point(aes(colour=pcaGroup), size = 0.01) +
  coord_equal(ratio = 0.5) +
  theme(legend.direction ='vertical',
        legend.position = 'right')
```

Logistic Regression

```
fitlr <- glm(Churn ~ ExpectedFulfillmentCharge+ActualCharge+TotalOrderItems+
               TotalPickedLines+TotalQtySubbed+TotalQtyOOS+TotalPickTimeSeconds+
               AvailabilityPreSubPercentage+PercentageOutOfStocks+
               PercentageSubstitutions+ValueOfSubstitutions+ValueOfOutOfStocks+
               RelatedCallsCount, family=binomial, data=merged_customers_train)

summary(fitlr)
varImp(fitlr)

pred <- predict(fitlr, newdata=merged_customers_test[-953], type="response")
pred <- factor(ifelse(pred < 0.5, 0, 1))

# generate a confusion matrix
tab <- table(merged_customers_test$Churn, pred)
tab
```

```r
# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)

# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

Use drop1

```r
# Try the drop1 function - it compares the overall model (fitlr) with the model resulting
# from removing that one specific variable.
drop1(fitlr, test="Chisq")
```

Pairwise Interactions

```r
fitlr2 <- glm(Churn ~ (ExpectedFulfillmentCharge+ActualCharge+TotalOrderItems+
                       TotalPickedLines+TotalQtySubbed+TotalQtyOOS+
                       TotalPickTimeSeconds+AvailabilityPreSubPercentage+
                       PercentageOutOfStocks+PercentageSubstitutions+
                       ValueOfSubstitutions+ValueOfOutOfStocks+RelatedCallsCount)^2,
              family=binomial, data=merged_customers_train)
summary(fitlr2)
varImp(fitlr2)

pred <- predict(fitlr2, newdata=merged_customers_test, type="response")
pred <- factor(ifelse(pred < 0.5, 0, 1))

# generate a confusion matrix
tab <- table(merged_customers_test$Churn, pred)
tab

# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)
```

```
# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

3rd Degree Interactions

```
fitlr3 <- glm(Churn ~ (ExpectedFulfillmentCharge+ActualCharge+TotalOrderItems+
                       TotalPickedLines+TotalQtySubbed+TotalQtyOOS+
                       TotalPickTimeSeconds+AvailabilityPreSubPercentage+
                       PercentageOutOfStocks+PercentageSubstitutions+
                       ValueOfSubstitutions+ValueOfOutOfStocks+RelatedCallsCount)^3,
            family=binomial, data=merged_customers_train)
summary(fitlr3)
varImp(fitlr3)

pred <- predict(fitlr3, newdata=merged_customers_test, type="response")
pred <- factor(ifelse(pred < 0.5, 0, 1))

# generate a confusion matrix
tab <- table(merged_customers_test$Churn, pred)
tab

# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)

# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

Conditional Logistic Regression

Create a subset of those that shopped more than once. This is a "conditional" analysis i.e. given that a customer shopped more than once, what are features that best predict that they might leave.

```
# In order to run logistic regression, we need to aggregate by customer (sequenceID).
# This will only run on the numerical columns, so add back in the Churn column afterwards.
merged_customers_multiple =
  aggregate(merged_customers[c("ExpectedFulfillmentCharge","ActualCharge",
                               "TotalOrderItems","TotalPickedLines",
                               "TotalQtySubbed","TotalQtyOOS","TotalPickTimeSeconds",
                               "AvailabilityPreSubPercentage","PercentageOutOfStocks",
                               "PercentageSubstitutions","ValueOfSubstitutions",
                               "ValueOfOutOfStocks","RelatedCallsCount",
                               "DaysSinceLastShop")],
```

```
            by=list(merged_customers$sequenceID), FUN=mean)

# Add back in the Churn response variable, 1-1000 (dormant=1), 1001-1905 (current=0)
for (i in 1:1000) merged_customers_multiple$Churn[i] = 1
for (i in 1001:1905) merged_customers_multiple$Churn[i] = 0
# make it a factor
merged_customers_multiple$Churn <- as.factor(merged_customers_multiple$Churn)
```

```
# Generate a version of DaysSinceLastShop with the "sd" function applied to it
merged_customers_multiple$DaysSinceLastShop_sd <-
  tapply(merged_customers$DaysSinceLastShop, merged_customers$sequenceID, sd)
# Generate a version of DaysSinceLastShop with the "skewness" function applied to it
merged_customers_multiple$DaysSinceLastShop_skew <-
  tapply(merged_customers$DaysSinceLastShop, merged_customers$sequenceID, skewness)
# Generate a version of DaysSinceLastShop with the "kurtosis" function applied to it
merged_customers_multiple$DaysSinceLastShop_kurt <-
  tapply(merged_customers$DaysSinceLastShop, merged_customers$sequenceID, kurtosis)
# to find out how many NA's we have...
sum(is.na(merged_customers_multiple$DaysSinceLastShop_sd))

# find out how many customers only shopped once and are not classified as leavers
sum(is.na(merged_customers_multiple$DaysSinceLastShop_sd)& merged_customers_multiple$Churn==0)

# create a dataframe without these 426 one-time-only shoppers
merged_customers_multiple <-
  subset(merged_customers_multiple, !is.na(merged_customers_multiple$DaysSinceLastShop_sd))
# check the dimensions of the subsetted data frame that will be used for conditional LR
dim(merged_customers_multiple)
```

```
# Split data
set.seed(123)
s <- sample(nrow(merged_customers_multiple), round(.5*(nrow(merged_customers_multiple))))
merged_customers_train <- merged_customers_multiple[s,]     # training set
merged_customers_test  <- merged_customers_multiple[-s,]    # test set
dim(merged_customers_train)
dim(merged_customers_test)
```

```
# add higher order moments as predictors
fitlr4 <- glm(Churn ~ DaysSinceLastShop_sd+DaysSinceLastShop_skew+
                DaysSinceLastShop_kurt+ExpectedFulfillmentCharge+ActualCharge+
                TotalOrderItems+TotalPickedLines+TotalQtySubbed+TotalQtyOOS+
                TotalPickTimeSeconds+AvailabilityPreSubPercentage+PercentageOutOfStocks+
                PercentageSubstitutions+ValueOfSubstitutions+
                ValueOfOutOfStocks+RelatedCallsCount,
            family=binomial, data=merged_customers_train)
summary(fitlr4)
varImp(fitlr4)

pred <- predict(fitlr4, newdata=merged_customers_test, type="response")
pred <- factor(ifelse(pred < 0.5, 0, 1))

# generate a confusion matrix
tab <- table(merged_customers_test$Churn, pred)
tab
```

```r
# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)

# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

```r
# add interactions
fitlr5 <- glm(Churn ~ (DaysSinceLastShop_sd+DaysSinceLastShop_skew+
                DaysSinceLastShop_kurt+ActualCharge)^2+
                ExpectedFulfillmentCharge+RelatedCallsCount+
                TotalOrderItems+TotalPickedLines+
                TotalQtySubbed+TotalQtyOOS+TotalPickTimeSeconds+
                AvailabilityPreSubPercentage+PercentageOutOfStocks+
                PercentageSubstitutions+ValueOfSubstitutions+
                ValueOfOutOfStocks,
            family=binomial, data=merged_customers_train)

summary(fitlr5)
varImp(fitlr5)

pred <- predict(fitlr5, newdata=merged_customers_test, type="response")
pred <- factor(ifelse(pred < 0.5, 0, 1))

# generate a confusion matrix
tab <- table(merged_customers_test$Churn, pred)
tab

# What proportion of dormant customers are missclassified?
tab[2]/(tab[2]+tab[4])

# What proportion of those who have stayed shopping are missclassified?
tab[3]/(tab[3]+tab[1])

# What proportion of the predicted leavers actually left?
tab[4]/(tab[4]+tab[3])

# What is the overall error rate for the test data?
mean(pred != merged_customers_test$Churn)

# what percentage of the test observations are correctly classified?
sum(diag(tab))/sum(tab)
```

Summary of Results

```
df <- data.frame(ML_Method=c("Single Tree","Bagging","Boosting","Random Forest","SVM",
                             "LDA", "QDA", "KNN", "Logistic Regression",
                             "Logistic Regression*" ),
                 Tst_Err_Rate=c(27, 25, 25.9, 24.8, 25.2, 31.1, 34.8, 44.4, 31.3, 23))
ggplot(df, aes(x=reorder(ML_Method, Tst_Err_Rate), y=Tst_Err_Rate, fill=Tst_Err_Rate)) +
  geom_bar(stat='identity') + coord_flip() +
  scale_fill_gradient(low = "blue", high = "red") +
  ggtitle("Aggregated Data: Machine Learning Method vs Error Rates") +
  ylab("Error rate percentage on the test dataset") + xlab("Machine Learning Method") +
  labs(fill = "Test Error Rate %")
```