Churn Rates Minimization

Kelso Quan

May 15, 2019

Executive Summary

Churn rates is an important aspect to any financial tech or commercial company. Churning is when the customer cancels their subscription or ceases doing business with the company. The goal of this analysis is to minimize churn rates. In this analysis, complete case analysis was used. The number of observations was 27000 dropped to 8453 and the number of features used to find strong indicators of churn rate went from 31 to 28. The analysis used decision tree, random forest, discrete, real, and gentle Ada Boosting to model the best way to predict churn rates. Every model had very similar misclassification errors with the best model being a classification tree. The best features to indicate churn rates pertained to credit card information, rewards from credit cards, and user's financial information.

1 Introduction

Churn rates are of interest to the Financial Tech companies and commercial businesses that rely on subscriptions and sale of services. Churning means cancellation of a subscription which is bad for the company. All companies would like to minimize their customers' churn rates to maximize profit. Churn is a binary response with a customer either churned or not churned. The data set has 31 features and 27000 observations. Given the data, the analysis tries to minimize churn rates by determining which features contribute heavily to customers cancelling their subscriptions and which model is the best.

2 Methods

In this analysis, decision tree, discrete, real, and gentle Ada Boosting along with random forest were used. R/Rstudio was used to perform the analysis. For the boosting techniques, the data was divided into 2/3 training set and 1/3 test set with cross validation being used. For the random forest, the data was split 60% training set and 40% test set. tuneRF was a function used to determine how many features were optimal for the random forest model which turned out to be 5 with the minimal OOB error. For decision tree, cross validation was used to grow a large tree, then 1SE rule was applied to prune the tree into the best subtree. The data set had 31180 Na's. Due to heavy computational times, the analysis was done as a complete case analysis. After omitting observations with Na's, there were 8453 observations left in the data set. The analysis eliminated user (id), android_user, and app_web_user. User id was eliminated because it did not provide information in determining someone's churn rate. The variables android_user and app_web_user were also deleted because they were negatively correlated and binary. Being binary variables of negatively correlated variables signified that if the user was not an android_user meant the user was ios_user and if the user was not using application, then they were a web_user.

3 Results

Table 1: Table of the response variable

	count
!churn	5239
churn	3214

3.1 Exploratory Data Analysis

Table 1 shows the response was binary. About 60% of the users did not churn and 40% of the users did churn. This proportion was used to split the training and test data for the random forest model. In addition, there were several features that were so skewed in distributions that it was better for those variables to become binary features. These features include registered_phones, deposits, withdrawal, purchase partners, purchases, cc taken, cc_disliked, cc_liked, and cc_application begin. cc is short for credit card. Furthermore, the analysis had to define type 1 and type 2 errors. Type 1, false positive, is predicting that the customers had churned when actually they had not churned. Type 2, false negative, is predicting that the customers had not churned when really they had churned.

In table 2, there are strong correlations between cc_recommended, rewards_earned, and reward_rate. There was little to no correlation between age and credit_score. Age and credit_score were weakly, positively correlated to cc_recommended, rewards_earned, and reward_rate features.

Table 2: Correlation plot of non-binary, numerical features

	age	credit_score	cc_recommended	rewards_earned	reward_rate
age	1.00	0.02	0.14	0.13	0.12
credit_score	0.02	1.00	0.13	0.15	0.17
cc_recommended	0.14	0.13	1.00	0.89	0.85
rewards_earned	0.13	0.15	0.89	1.00	0.96
reward_rate	0.12	0.17	0.85	0.96	1.00

3.2 Model Fitting and Results

This analysis did a preliminary logistic regression to examine if it was a good model for the data. It did not preform well. Consequentially, the analysis progressed into a classification tree. The best split for the decision tree was purchase partners. That was followed by the features: rewards_earned, reward_rate, cc_recommended, and credit_score. But a single decision tree is not stable, so an ensemble of trees were created.

Table 3: Misclassifications Error of Methods Used

Method	False Positive	False Negative	Total Misclassifications
Single Tree	15.21%	46.03%	27.10%
Gentle AdaBoost	8.13%	68.3%	31%
Real AdaBoost	11.79%	58%	29.34%
Discrete AdaBoost	13.27%	55.6%	29.34%
Random Forest	18.49%	46.40%	29.10%

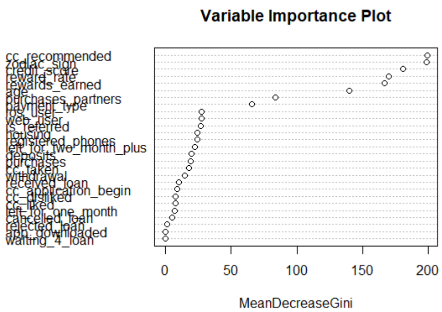
From table 3, the models have very similar misclassification errors. When looking at the total misclassification column, those error rates are strictly below 30%. The best model being the single classification tree with the lowest total misclassifications. Gentle AdaBoost had the lowest false positive rates among all the models.

The figures (a), (b), (c), (d) shows the variable importance when it comes to predicting churn rates. Financial factors such as cc_recommended, rejected_loans, and other credit card features are significant in predicting churn rates. Any features that were closer to the right side of the importance plot were considered influential to the users' churn rate. The variables concentrated on the left side of the variable importance plot were considered less significant when it came to determining whether a customer was going to churn or not.

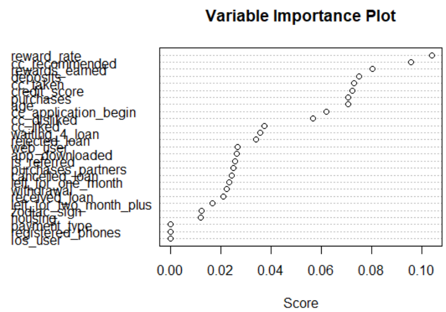
4 Conclusion

In determining an user's churn rate, it is best to consider their credit card history and financial information. cc_recommended, purchase_partners, reward_rate, and rejected_loan were of the highest variable importance when determining whether a subscriber stays in business or cancels their subscription. The analysis was slightly biased because the training-test data set ratio was different between random forest and the three methods of AdaBoost. In addition, all the models had very similar results with the decision tree being the most interpretable. The other models required variable importance plots to indicate which features contributed most to churn rates. In future analysis, the training-test data set ratio will be the same. Another constraint was the sheer amount of Na's in the data set. A considerable portion of the data was lost when complete case analysis was used rather than imputation or another method to handle missing values. Financial Tech companies should consider an user's financial history and credit card information to determine whether that user is at risk of churning.

Quan/Documents/SchoolWork/Stat702/final/rfVarImpPlot.png Quan/Documents/SchoolWork/Stat702/final/realVarPlot.png

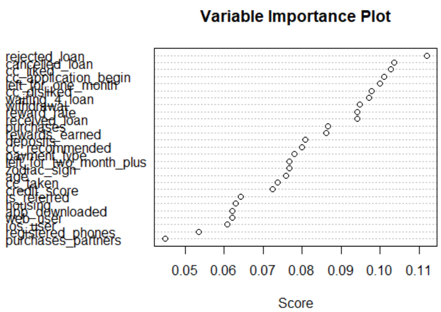


(a) Random Forest

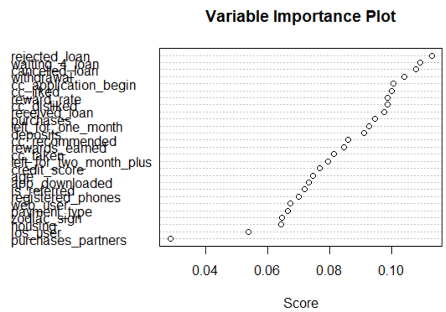


(b) Real AdaBoost

Quan/Documents/SchoolWork/Stat702/final/gentleVarPlot.png Quan/Documents/SchoolWork/Stat702/final/disVarPlot.png



(c) Gentle AdaBoost



(d) Discrete AdaBoost

5 References

Khan, Adnan
https://www.kaggle.com/nanda1331/churn-rate-minimization
stackexchange.com
overleaf.com

Appendix: R Code

```r
setwd("~/SchoolWork/Stat702/final")
churn.data = read.csv("churn.csv", header = T)
names(churn.data)
str(churn.data)
dim(churn.data)
sum(is.na(churn.data))
set.seed(702)


#Take log transform of age
churn.data$age<-log(churn.data$age)

#Set age as numerical
churn.data$age<-as.numeric(churn.data$age)

#Set response variable as a factor and add labels
churn.data$churn <- factor(churn.data$churn, levels=0:1, labels=c("!churn", "churn"))

#Set housing as factor
churn.data$housing <- factor(churn.data$housing)

#app_web_user correlated to web_user
#remove app web_user for now we will modify based on step wise model building
cor.test(~app_web_user+web_user, data=churn.data)
churn.data$app_web_user<-NULL

#ios_user correlated to android_user
#remove android user for now, we will modify based on step wise model building
cor.test(~ios_user+android_user, data=churn.data)
churn.data$android_user<-NULL

#remove user feature
churn.data$user<-NULL

#Set zodiac_sign as factor
churn.data$zodiac_sign<-as.factor(churn.data$zodiac_sign)


#Change features to factor (5:9) and (11:13)

#Features 5:9 (deposits, withdrawal, purchase partners, purchases, cc taken)
churn.data[,5:9]=sapply(churn.data[,5:9],function(x) replace(x,x>0,1))

#Features 11:13 (cc_disliked,cc_liked,cc_application begin)
churn.data[,11:13]=sapply(churn.data[,11:13],function(x) replace(x,x>0,1))

#set deposits as factor
churn.data$deposits<-as.factor(churn.data$deposits)

#set withdrawal as factor
churn.data$withdrawal<-as.factor(churn.data$withdrawal)

#set purchases as factor
churn.data$purchases<-as.factor(churn.data$purchases)

#set purchases_partners as factor
churn.data$purchases_partners<-as.factor(churn.data$purchases_partners)

#set cc taken as factor
churn.data$cc_taken<-as.factor(churn.data$cc_taken)

#set cc_disliked as factor
churn.data$cc_disliked<-as.factor(churn.data$cc_disliked)
```

```r
#set cc_liked as factor
churn.data$cc_liked<-as.factor(churn.data$cc_liked)

#set cc_application_begin as factor
churn.data$cc_application_begin<-as.factor(churn.data$cc_application_begin)

#set app downladed as factor
churn.data$app_downloaded<-as.factor(churn.data$app_downloaded)

#set web user as factor
churn.data$web_user<-as.factor(churn.data$web_user)

#Set ios user as factor
churn.data$ios_user<-as.factor(churn.data$ios_user)

#registered phone as binary and to factor
churn.data[,17]=sapply(churn.data[,17],function(x) replace(x,x>0,1))
churn.data$registered_phones<-as.factor(churn.data$registered_phones)

#waiting for loan as factor
churn.data$waiting_4_loan<-as.factor(churn.data$waiting_4_loan)

#cancelled loan as factor
churn.data$cancelled_loan<-as.factor(churn.data$cancelled_loan)

#Received loan as factor
churn.data$received_loan<-as.factor(churn.data$received_loan)

#set rejected loan as factor
churn.data$rejected_loan<-as.factor(churn.data$rejected_loan)

#set left for one month as factor
churn.data$left_for_one_month<-as.factor(churn.data$left_for_one_month)

#set left for more than twomonths to factor
churn.data$left_for_two_month_plus<-as.factor(churn.data$left_for_two_month_plus)

#set is_referred as factor
churn.data$is_referred<-as.factor(churn.data$is_referred)

churn.data <- na.omit(churn.data)
dim(churn.data)
summary(churn.data)
sum(is.na(churn.data))

set1<-churn.data[churn.data$churn=="churn",]
set0<-churn.data[churn.data$churn=="!churn",]

dim(set1)  # 3214   28
dim(set0)  # 5239   28
3214*2/3   # 2142.667
5239*2/3   #3492

training1 <- sample(1:3214,2143)
test1 <- (1:3214)[-training1]
sum((1:3214) == sort(c(training1,test1)))

training0 <- sample(1:5239,3492)
test0 <- (1:5239)[-training0]
sum((1:5239 == sort(c(training0,test0)))) #2788

train <- rbind(set1[training1,], set0[training0,])
test <- rbind(set1[test1,], set0[test0,])

numeric.var <- sapply(churn.data, is.numeric)
```

```
corr.matrix <- cor(churn.data[,numeric.var])
library(corrplot)
corrplot(corr.matrix, main = '\n\nCorrelation Plot for Numerical Variables')
dim(train)
dim(test)
set.seed(100)
logmod <- glm(churn ~., family = binomial(link = "logit"), data = churn.data)
summary(logmod)

logmodel <- glm(churn ~.,family=binomial(link="logit"),data=train)
summary(logmodel)
logmodel1 <- glm(churn~ housing+credit_score+purchases_partners+cc_taken+
                   cc_recommended+cc_liked+cc_application_begin+web_user+
                   ios_user+registered_phones+payment_type+received_loan+
                   rejected_loan+zodiac_sign+left_for_two_month_plus+rewards_earned+
                   reward_rate+is_referred,
                 family= binomial(link = "logit"), data = train)
summary(logmodel1)
logmodel2 <- glm(churn~ housing+credit_score+purchases_partners+cc_taken+
                   cc_recommended+cc_liked+cc_application_begin+web_user+
                   registered_phones+received_loan+
                   rejected_loan+left_for_two_month_plus+rewards_earned+
                   reward_rate+is_referred,
                 family= binomial(link = "logit"), data = train)
summary(logmodel2)

testmodel  <- glm(churn~ housing+credit_score+purchases_partners+cc_taken+
                    cc_recommended+cc_liked+cc_application_begin+web_user+
                    registered_phones+received_loan+
                    rejected_loan+left_for_two_month_plus+rewards_earned+
                    reward_rate+is_referred,
                  family= binomial(link = "logit"), data = test)
summary(testmodel)

###############################################################################
#Boosting
###############################################################################

#cross validation errors 5.3.3 islr
lmat <- matrix(c(0,1,4,0), nrow=2, byrow=T)

library(ada)

#Discrete adaboost using 10 fold xval, cp=0
default=rpart.control(xval=10,cp=0)
fitdis<-ada(churn~.,data=train,iter=50,loss="e",type="discrete", control=default)
print(fitdis) #training error 0.087


#misclassification rate = (false positive + false negative)/total
pred_val.2 <- predict(fitdis, newdata= test)
table.2<-table(test$churn, pred_val.2)
miss.rate2<-(table(test$churn,pred_val.2)[1,2]+table(test$churn,pred_val.2)[2,1])/length(test$churn)
miss.rate2 #28.63% misclassification overall, discrete adaboost might overfit

#false positive and false negative rates
fp.rate2<-(table(test$churn,pred_val.2)[1,2])/(table(test$churn,pred_val.2)[1,2]+table(test$churn,pred_val.2)[1,1])
fp.rate2 #16.77% false positive


fn.rate2<-(table(test$churn,pred_val.2)[2,1])/((table(test$churn,pred_val.2)[2,1])+table(test$churn,pred_val.2)[2,2])
fn.rate2#47.03% false negatives
```

```r

#variable importance plot
varplot(fitdis)
vip <- varplot(fitdis,plot.it=FALSE,type="scores")
round(vip,4)

#Real adaboost
fitreal<-ada(churn~.,data=train,iter=50,type="real",
             control=rpart.control(maxdepth=2,cp=-1,minsplit=0))
fitreal
varplot(fitreal)
#misclassification rate = (false positive + false negative)/total
pred_val.3 <- predict(fitreal, newdata= test)
table.3<-table(test$churn, pred_val.3)
miss.rate3<-(table(test$churn,pred_val.3)[1,2]+table(test$churn,pred_val.3)[2,1])/length(
    test$churn)
miss.rate3 #for real adaboost still at 29.84%

#false positive and false negative rates
fp.rate3<-(table(test$churn,pred_val.3)[1,2])/(table(test$churn,pred_val.3)[1,2]+table(test
    $churn,pred_val.3)[1,1])
fp.rate3 #15.08% false positive


fn.rate3<-(table(test$churn,pred_val.3)[2,1])/((table(test$churn,pred_val.3)[2,1])+table(
    test$churn,pred_val.3)[2,2])
fn.rate3#59.43% false negatives


#gentle adaboost
fitgen<-ada(churn~.,data=train,test.x=test[,-1],test.y=test[,1],iter=50,
            type="gentle",
            control=rpart.control(cp=-1,maxdepth=8))
(fitgen)
varplot(fitgen)
#misclassification rate = (false positive + false negative)/total
pred_val.4 <- predict(fitgen, newdata= test)
table.4<-table(test$churn, pred_val.4)
miss.rate4<-(table(test$churn,pred_val.4)[1,2]+table(test$churn,pred_val.4)[2,1])/length(
    test$churn)
miss.rate4 #still at 29.77%

#false positive and false negative rates
fp.rate4<-(table(test$churn,pred_val.4)[1,2])/(table(test$churn,pred_val.4)[1,2]+table(test
    $churn,pred_val.4)[1,1])
fp.rate4 #18.17% false positive


fn.rate4<-(table(test$churn,pred_val.4)[2,1])/((table(test$churn,pred_val.4)[2,1])+table(
    test$churn,pred_val.4)[2,2])
fn.rate4#46.20% false negatives


# training the model
model_ada<-train(churn~., data=train,method='ada',tuneGrid=grid)
plot(model_ada)


set.seed(702)

##random forest##
library(randomForest)
dim(train)
c.tune <- tuneRF(train[2:28], train$churn, ntreeTry=50, stepFactor=2,
                 improve=0.05, trace=TRUE, plot=TRUE, dobest=FALSE, main = "mtry vs OOB
```

```
    error")
# mtry  OOBError
# 3.OOB     3 0.2917480
# 5.OOB     5 0.2857143
# 10.OOB   10 0.2897959
# 5 is the best mtry
c.tune
plot(c.tune)
train.rf <- randomForest(churn~.,data = train, mtry = 5, ntree = 501,norm.votes = F)
test.rf <- randomForest(churn~., data = test, mtry = 5, ntree = 501,norm.votes = F)
print(train.rf)
print(test.rf)
#test.rf OOB estimate of  error rate: 29.42% (misclass rate)
# OOB estimate of  error rate: 29.42%
# Confusion matrix:
#   !churn churn class.error
# !churn   1415   332   0.1900401
# churn     497   574   0.4640523

#varImpPlot(train.rf, main = "Variable Importance of Churn Rates")
varImpPlot(test.rf, main = "Variable Importance of Churn Rates")
plot(train.rf, main = "MSE vs # of bootstrap Samples")
plot(test.rf, main="MSE vs. # of boostrap Samples")
legend("center", legend = test.rf)
legend("topright",
       legend=as.character(levels(mtcars$cyl)),
       fill = rainbow(nlevels(mtcars$cyl)),
       title = "cyl")

### prediction ###
ind<-sample(2,nrow(churn.data),replace=T,prob=c(.6, .4)) #1/2 (.6/.4): training/testing
table(ind)
churn.rf<-randomForest(churn~., data=churn.data[ind==1,])
print(churn.rf) #OOB estimate of  error rate: 27.7%
varImpPlot(churn.rf)
churn.pred <- randomForest(churn~., data = churn.data[ind == 2,])
print(churn.pred)
varImpPlot(churn.pred, main = "Variable Importance Plot")
# misclasification error rate: 28.67%
# Confusion matrix:
#   !churn churn class.error
# !churn   1712   372   0.1785029
# churn     601   709   0.4587786

##rpart comparisons###
library(rpart)
library(caret)
library("e1071")

my.control <- rpart.control(xval=10, cp=0)
tree <- rpart(churn ~.,
              data=churn.data, method="class", control=my.control)
plot(tree, margin = .1, uniform = T)
text(tree, use.n = T)
printcp(tree)
plotcp(tree)
0.76602 +0.012997 #0.779017 tree 16, 30 splits, 31 terminal nodes
besttree<-prune(tree,cp=0.0018)
plot(besttree,margin=.1)
text(besttree,use.n=T)
printcp(besttree)
besttree8 <- prune(besttree, cp = .004)
plot(besttree8, uniform = T)
text(besttree8, use.n = T)
printcp(besttree8)
```

```
pred.tree<- predict(besttree,newdata=churn.data[ind==2,], type='class')
table(observed=churn.data[ind==2,"churn"],predicted=pred.tree)
(588+350)/(1760  + 313 +562  + 703) #0.2810066

nsplits= besttree$cptable[,2]
test_error= besttree$cptable[,3]
xerror= besttree$cptable[,4]
xstd= besttree$cptable[,5]
plot(nsplits, test_error, type = 'l', xlab = "Number of splits", ylab = "Test error",
     main = "Test error vs Number of splits")
plot(nsplits, test_error)
lines(nsplits, xerror, lty=3, col = "red")
lines(nsplits, xerror+xstd, lty=4, col = "blue")
lines(nsplits, xerror-xstd, lty=4, col = "blue")
legend(17,1, c("test error", "xerror", "+/- 1 xstd"), lty = c(1,3,4))
xtable(varImp(besttree))
```

Listing 1: Churn Rates