**CKME136 - Capstone Project**

**Initial Results**

*TalkingData AdTracking Fraud Detection Challenge*

*https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection*

Phi Huynh - 500777278

[*https://github.com/PHLHY/Capstone-*](https://github.com/PHLHY/Capstone-)

The following steps were done as determined by the previous Literature Review and Data Descriptions report.

**Step 1: Data cleaning:** No missing data noted

**Step 2: Exploratory analysis:** Imbalance data noted, correlation completed, started with some visualizations

**Step 3: Feature selection:** PCA and forward selection were done.

**Step 4: Classification algorithms:** ROSE oversampling/undersampling utilized for imbalance data set. Random forest was done for the initial algorithm.

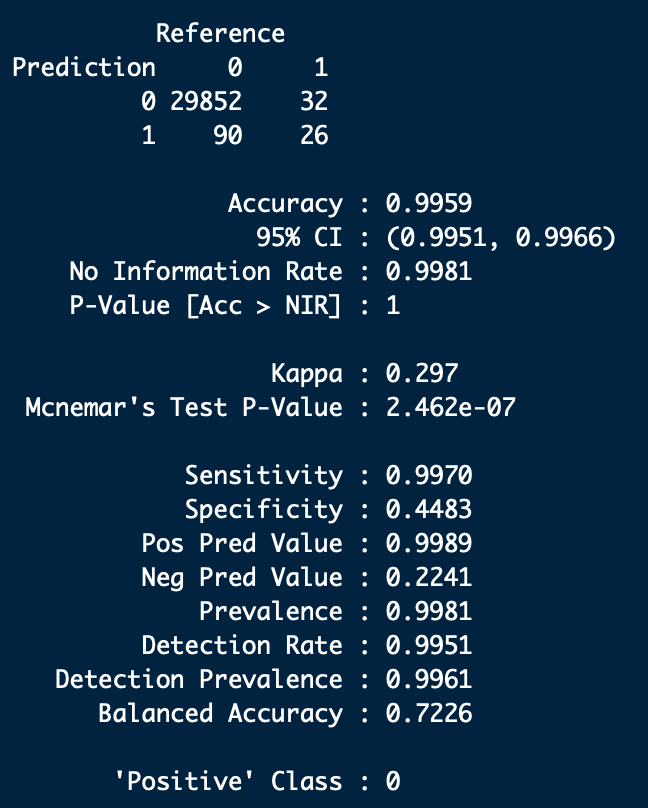
**Step 5: Evaluation of models:** Crosstable completed at this time. Please see following page for initial result.

**Problems and Limitations**

* Dataset was originally trimmed from original amount to 1,000,000. However, there were still lots of slow down and memory errors. Thus, dataset was further trimmed to 100,000.
* Data visualization needs to be fixed at this point. Graphs requiring organization and show the top values of each attributes
* Variable selections not considering time attribute currently
* Initial result shows low specificity. Fine tuning of algorithm needed.

**Plans for Future Submission**

* Complete visualization and answers questions posed for exploratory analysis
* Review feature selection again for selection of variables
* Consider different imbalance techniques
* Fine tune Random Forest algorithm and ensure cross validation is done.
* Trial different algorithms and create chart showcasing the different results
* Final report and preparation for presentation



**Coding**

#Used for quicker dataset loading due to the big datasets

library(data.table)

library(plyr)

#data visualization

library(ggplot2)

#corrplot

library(corrplot)

#Cross validation and feature selection

library(caret)

library(MASS)

library(leaps)

#Class imbalance

install.packages("ROSE")

library(ROSE)

#classifier models

library(caret)

install.packages("randomForest")

library(randomForest)

#loading up datasets

train <- fread("all/train.csv", showProgress = T)

test <- fread("all/test.csv", showProgress = T)

#quick look at the data

head(train)

tail(train)

str(train)

#checking for missing values broken down by variables

colSums(is.na(test))

colSums(is.na(train))

#Note attribute\_time having blank entries which makes sense since they did not download app (target variable). Proven below where the number matches

colSums(train=="")

table(train$is\_attributed)

#taking a look at the dataset of target variable. Noted that it is skewed (0.24% shows target attribute)

table(train$is\_attributed)

#to control randomization for future processing

set.seed(575)

#sampling to make this datasets smaller for easier computation. Note computer limitations and crashing on R.

#Would usually do a 70/30 split, however, original percentage differences between test and train is 90/10 split

s.train <- train[sample(nrow(train), 100000), ]

s.test <- test[sample(nrow(test), 10000), ]

check\_index <- sample(1:nrow(s.train), 0.7 \* nrow(s.train))

traincheck.set <- s.train[check\_index,]

testcheck.set <- s.train[-check\_index,]

#target variable. Still skewed. 0.25% shows target attribute. Similar to original dataset.

#will need to balance dataset (undersample/oversample)

table(traincheck.set$is\_attributed)

#splitting click\_time into different columns for better analysis

#removing click\_time and year and month since they are the same for all

#consider adding in seconds?

traincheck.set$click\_time<-as.POSIXct(traincheck.set$click\_time, format = "%Y-%m-%d %H:%M")

traincheck.set$year=year(traincheck.set$click\_time)

traincheck.set$month=month(traincheck.set$click\_time)

traincheck.set$days=weekdays(traincheck.set$click\_time)

traincheck.set$hour=hour(traincheck.set$click\_time)

table(traincheck.set$year)

table(traincheck.set$month)

traincheck.set$click\_time=NULL

traincheck.set$year=NULL

traincheck.set$month=NULL

#changing is\_attributed and to factor

traincheck.set$is\_attributed = factor(traincheck.set$is\_attributed)

#variables frequency, need to look at ggplot2 for desc and top 15

count.trainip <- count(s.train, "ip")

ggplot(traincheck.set, aes(x=ip), color="steelblue") + geom\_bar()

count.trainapp <- count(s.train, "app")

ggplot(traincheck.set, aes(x=app), color="steelblue") + geom\_bar()

count.traindevice <- count(s.train, "device")

ggplot(traincheck.set, aes(x=device), color="steelblue") + geom\_bar()

count.trainos <- count(s.train, "os")

ggplot(traincheck.set, aes(x=os), color="steelblue") + geom\_bar()

count.trainchannel <- count(s.train, "channel")

ggplot(traincheck.set, aes(x=channel), color="steelblue") + geom\_bar()

#changing days to numeric (monday = 1, Tuesday =2, wednesday-3, thursday = 4). Remember to switch to test as well later

traincheck.set$days <- gsub("Thursday", "4", traincheck.set$days)

traincheck.set$days <- gsub("Wednesday", "3", traincheck.set$days)

traincheck.set$days <- gsub("Tuesday", "2", traincheck.set$days)

traincheck.set$days <- gsub("Monday", "1", traincheck.set$days)

#remove attribute\_time for correlation (pearson)

cor.traincheck.set <- traincheck.set[,c(-6,-8,-9)]

#changing is\_attributed back to numeric for correlation

cor.traincheck.set$is\_attributed <- as.numeric(as.character(cor.traincheck.set$is\_attributed))

#cor (pearson), note negative weak correlation for channel and app

corrplot(cor(cor.traincheck.set, method="spearman"), method="number")

#PCA if selected

pc\_traincheck.set <- princomp(cor.traincheck.set, cor=TRUE, score=TRUE)

summary(pc\_traincheck.set)

#We usually dont consider anything less than 0.5 for variances. Thus we should consider at least 5 components

#98.99

plot(pc\_traincheck.set)

#feature selection (forward) if selected

full <- lm(is\_attributed~ip+app+device+os+channel, data=cor.traincheck.set)

null <- lm(is\_attributed~1, data=cor.traincheck.set)

stepF <- stepAIC(null,scope=list(lower=null, upper=full), direction ="forward", trace=TRUE)

summary(stepF)

#thus, all variables should be selected as they are all significant

#to correct imbalance using over and under sampling

balanced\_cor.traincheck.set <- ovun.sample(is\_attributed ~ ., data = cor.traincheck.set, method = "both", p=0.5, N=70000, seed = 1)$data

table(balanced\_cor.traincheck.set$is\_attributed)

# now is\_attributed is balanced (34919 - 0, 35081 - 1)

#changing back to factor

balanced\_cor.traincheck.set$is\_attributed = factor(balanced\_cor.traincheck.set$is\_attributed)

#random forest note:note enough memory with 1000000, had to switch it to 70000

rf.traincheck.set <- randomForest(formula = is\_attributed ~ ., data = balanced\_cor.traincheck.set, importance = TRUE)

#using default mtry, aware that you can fine tune mtry using caret randomforest instead

#predicting

#first factor and making it similar to balanced\_cor.traincheck.set

cor.testcheck.set <- testcheck.set[,c(-6,-8,-9)]

cor.testcheck.set$is\_attributed = factor(cor.testcheck.set$is\_attributed)

predict.rf <- predict(rf.traincheck.set, cor.testcheck.set)

confusionMatrix(predict.rf, cor.testcheck.set$is\_attributed)

#predicting on test set given

predicttest.rf <- predict(rf.traincheck.set, cor.s.test)

#9893 -0, 107 - 1

table(predicttest.rf)