**CKME136- Capstone Project**

Literature Review and Data Descriptions

Sheena Chauhan

Talking Data Ad tracking fraud detection Challenge

<https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data>

Click fraud is the act of generating fraudulent clicks on pay-per-click advertisements which results in artificially inflating web traffic and potential loss of revenue to the advertiser.

Below are few questions that are posted to assist with this challenge:

1)     Is there specific type of mobile applications which are downloaded more often by the users?

2)     Is there specific day and time when mobile users download those specific applications?

3)     Is there any relation between the mobile devices used and fraudulent clicks done?

4) Which clicks were not fraudulent clicks

The following steps were done previously in Literature Review and Data Descriptions report.

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

**Step 2: Exploratory analysis:** Completed Correlation, Imbalance data noted, started with some visualizations

**Step 3: Feature selection:** Forward selection were done.

**Step 4: Classification algorithms:** ROSE oversampling/undersampling utilized for imbalance data set. For Initial algorithm, I chose Random forest and was completed.

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

**Problems and Limitations**

* This is a huge dataset, I originally loaded amount to 1,000,000. However, due to lots of slowdowns and memory errors I further trimmed to 100,000.
* Algorithms still need fine tuning.
* Data visualization needs to be fixed at this point.
* Variable selections not considering time attribute currently
* Initial result shows low specificity.

**Plans for Future Submission**

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

Attributes of these datasets includes:  
1) ip

2) app

3) device

4) os

5) channel

6) click\_time

7) attributed\_time

8) is\_attributed (target attribute)

**Description of Attributes:**

**ip**: ip address of click.

* **app**: app id for marketing.
* **device**: device type id of user mobile phone (e.g., iphone 6 plus, iphone 7, huawei mate 7, etc.)
* **os**: os version id of user mobile phone
* **channel**: channel id of mobile ad publisher
* **click\_time**: timestamp of click (UTC)
* **attributed\_time**: if user download the app for after clicking an ad, this is the time of the app download
* **is\_attributed**: the target that is to be predicted, indicating the app was downloaded

**Initial Coding as below:**

Due to big data sets, I have used below libraries. I installed all these packages first.

library(data.table)

library(plyr)

* corrplot

library(corrplot)

* data visualization

library(ggplot2)

* 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

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

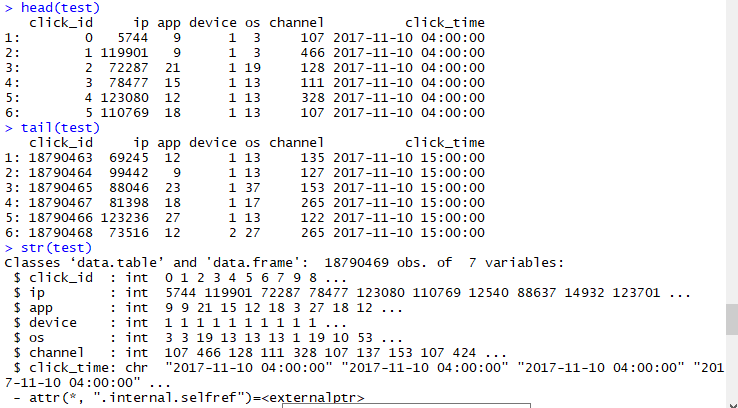
train <- fread("Ryerson/train.csv ", showProgress = T)

* 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 with blank entries shows they did not download the app (target variable). Proven below where the number matches

colSums(train=="")

table(train$is\_attributed)

* Looking at the dataset of target variable, it is skewed (0.24% shows target attribute)

table(train$is\_attributed)

* Control randomization

set.seed(575)

* For easier computation and due to computer limitation, I’ve sampled the dataset. usually it’s 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))

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

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

* splitting click\_time into different columns for better analysis by removing click\_time and year and month since they are the same for all and added seconds:

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

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

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

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

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

table(TrainData.set$year)

table(TrainData.set$month)

TrainData.set$click\_time=NULL

TrainData.set$year=NULL

TrainData.set$month=NULL

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

will need to balance dataset (undersample/oversample)

table(TrainData.set$is\_attributed)

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

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

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

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

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

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

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

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

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

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

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

* Changed “is\_attributed” and to factor

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

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

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

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

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

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

* Removed attribute\_time for correlation (pearson)

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

* Changed is\_attributed back to numeric for correlation

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

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

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

* PCA if selected

pc\_TrainData.set <- princomp(cor.TrainData.set, cor=TRUE, score=TRUE)

summary(pc\_TrainData.set)

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

98.99

plot(pc\_TrainData.set)

* feature selection (forward) if selected

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

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

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

* now is\_attributed is balanced (34919 - 0, 35081 - 1)
* changing back to factor

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

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

rf.TrainData.set <- randomForest(formula = is\_attributed ~ ., data = balanced\_cor.TrainData.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.TrainData.set

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

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

predict.rf <- predict(rf.TrainData.set, cor.TestData.set)

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

* predicting on test set given

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

#9893 -0, 107 - 1

table(predicttest.rf)