Talking Data Adtracking fraud deduction

Sowmiya

6/16/2019

Introduction

Fraud risk is everywhere, but for companies that advertise online, click fraud can happen at an overwhelming volume, resulting in misleading click data and wasted money. Ad channels can drive up costs by simply clicking on the ad at a large scale. With over 1 billion smart mobile devices in active use every month, China is the largest mobile market in the world and therefore suffers from huge volumes of fraudulent traffic.

In this machine learning project, ywe had build a machine learning model to determine whether a click is fraud or not.

Summary

The report is split in three sections.

1) Exploratory data analysis after the data is loaded.
2) Apply machine learning algorithms of any model with for all the features.
3) Apply machine learning algorithm for selected features through exploratory data analysis

Method

This is a classification problem statement and Decision tree is used here as its provides the best accuracy.From our analysis we understand that training data has some imbalance once we split the data. So we have used SMOTE and re-model the data so that there is an improvement in accuracy and specificity is achieved.

Import the data and install related libraries

```
train_path <- ("https://raw.githubusercontent.com/sowmi121/Talkingdata-Adtracking-Fraud-deduction/maste
test_path <- ("https://raw.githubusercontent.com/sowmi121/Talkingdata-Adtracking-Fraud-deduction/master/

if(!require(tidyverse)) install.packages("tidyverse", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3
```

```
## -- Attaching packages ---------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
## Warning: package 'tidyr' was built under R version 3.5.3
```

```
## Warning: package 'readr' was built under R version 3.5.2
```

```
## Warning: package 'purrr' was built under R version 3.5.3

## Warning: package 'dplyr' was built under R version 3.5.3

## Warning: package 'stringr' was built under R version 3.5.3

## Warning: package 'forcats' was built under R version 3.5.3

## -- Conflicts --------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(lubridate)) install.packages("lubridate", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: lubridate

## Warning: package 'lubridate' was built under R version 3.5.3

##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```
if(!require(data.table)) install.packages("data.table", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: data.table

## Warning: package 'data.table' was built under R version 3.5.3

##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart
```

```
if(!require(ggplot2)) install.packages("ggplot2", repos ="http://cran.us.r-project.org")
if(!require(grid)) install.packages("grid", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: grid
```

```
if(!require(gridExtra)) install.packages("gridExtra", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: gridExtra

## Warning: package 'gridExtra' was built under R version 3.5.3

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
if(!require(DMwR)) install.packages("DMwR", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: DMwR

## Warning: package 'DMwR' was built under R version 3.5.3
```

Read the data file

Training data contains a click record, with the following features.

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

The test data is similar, with the following differences:

click_id: reference for making predictions is_attributed: not included

```
train <- fread(paste0(train_path,"train_sample.csv"), sep=",",
               na.strings = "",
               stringsAsFactors=T,
               nrows = 100000,
               data.table = F)
str(train)
```

```
## 'data.frame':    100000 obs. of  8 variables:
##  $ ip             : int  87540 105560 101424 94584 68413 93663 17059 121505 192967 143636 ...
##  $ app            : int  12 25 12 13 12 3 1 9 2 3 ...
##  $ device         : int  1 1 1 1 1 1 1 1 2 1 ...
##  $ os             : int  13 17 19 13 1 17 17 25 22 19 ...
##  $ channel        : int  497 259 212 477 178 115 135 442 364 135 ...
##  $ click_time     : Factor w/ 4309 levels "11/6/2017 16:00",..: 1883 757 1022 1611 4250 3792 3787 538
##  $ attributed_time: Factor w/ 223 levels "11/6/2017 17:19",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ is_attributed  : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
test <- fread(paste0(test_path,"test.csv"),
              sep=",",
              na.strings = "",
              stringsAsFactors=T,
              nrows = 100000,
              data.table = F)
str(test)
```

```
## 'data.frame':    100000 obs. of  7 variables:
##  $ click_id  : int  0 1 2 3 4 5 6 7 9 8 ...
##  $ ip        : int  5744 119901 72287 78477 123080 110769 12540 88637 14932 123701 ...
##  $ app       : int  9 9 21 15 12 18 3 27 18 12 ...
##  $ device    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ os        : int  3 3 19 13 13 13 1 19 10 53 ...
##  $ channel   : int  107 466 128 111 328 107 137 153 107 424 ...
##  $ click_time: Factor w/ 2 levels "10/11/2017 4:00",..: 1 1 1 1 1 1 1 1 1 1 ...
```

There is no difference between train and test data except we need to predict target (is_attributed) in test and attributed_time (Time taken to download Application) is not given in test data)

```
colSums(is.na(train))
```

```
##              ip             app          device              os
##               0               0               0               0
##         channel      click_time attributed_time   is_attributed
##               0               0           99773               0
```

There is no missing value at all, data is very clean and clear

```
colSums(train=='')
```

```
##              ip             app          device              os
##               0               0               0               0
##         channel      click_time attributed_time   is_attributed
##               0               0              NA               0
```

Attributes_time (Time taken to download) having blank entries, this is logically correct

Lets check the target variable how many are not downloaded in train data

```
table(train$is_attributed)
```

```
##
##     0     1
## 99773   227
```

Our assumption is correct since blank entries in Attributes_time is matching with Application not downlaoded in train data.As it's logically correct, we don't need do any further action on this and also notice that, this variable is not present in test data, so no point of keeping it in the train data too

```
train$attributed_time=NULL
train$click_time <- as.POSIXct(as.character(train$click_time), format = "%d/%m/%Y %H:%M")
```

Get the data from click_time in train data and split it into additional columns

```
train$year=year(train$click_time)
train$month=month(train$click_time)
train$days=weekdays(train$click_time)
train$hour=hour(train$click_time)
```

Data Cleaning

After getting new feature, let's remove original "click_time" variable

```
train$click_time=NULL
```

Check the unique number for each of the feature

```
apply(train,2, function(x) length(unique(x)))
```

```
##            ip           app        device            os       channel
##         34857           161           100           130           161
## is_attributed          year         month          days          hour
##             2             1             4             4            24
```

By looking into unique value, we can see data collected for one month in a year, so no point of keeping month and year variables

```
train$month=NULL
train$year=NULL
```

Exploratory Data Analysis

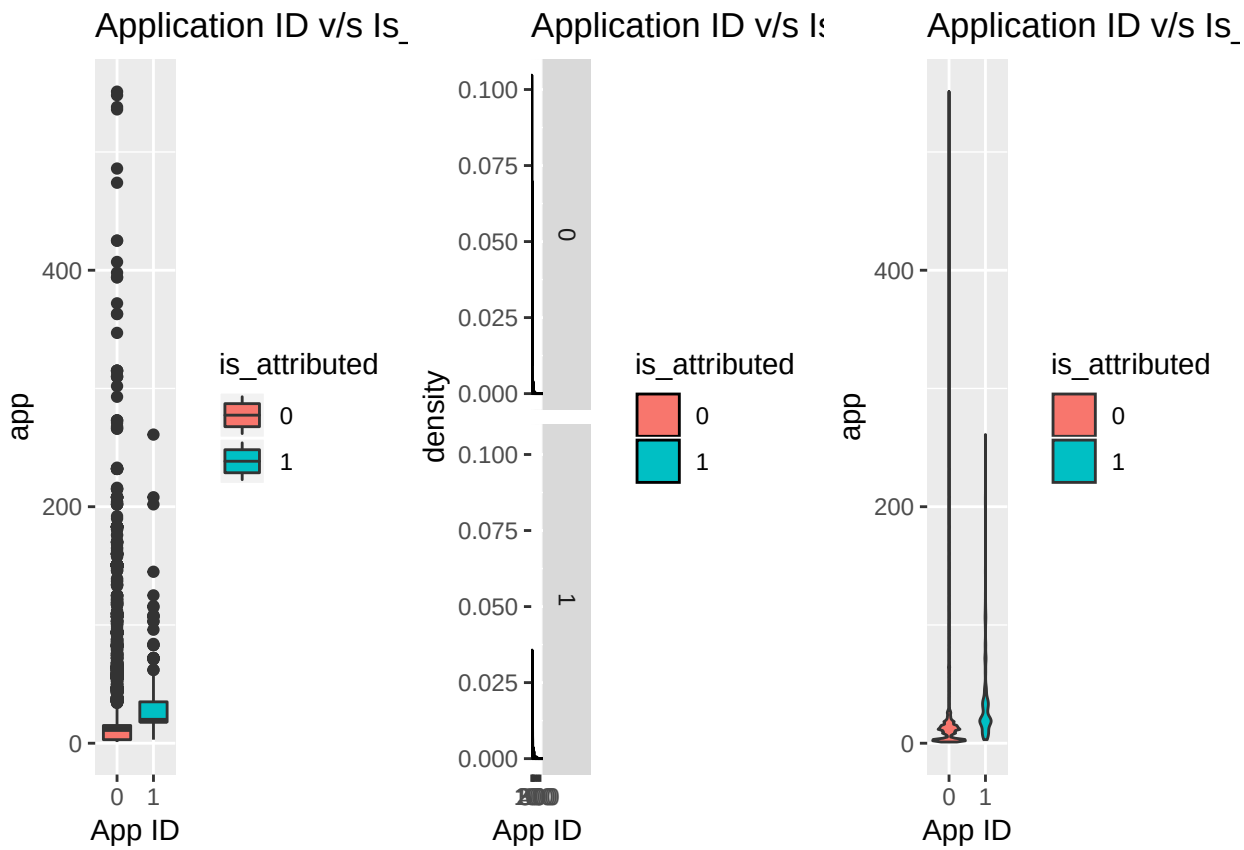
Convert variables into respective data type

```
train$is_attributed=as.factor(train$is_attributed)
train$days=as.factor(train$days)
train$os = as.factor(train$os)
train$device = as.factor(train$device)
train$channel = as.factor(train$channel)
train$hour = as.factor(train$hour)
str(train)
```

```
## 'data.frame':    100000 obs. of  8 variables:
##  $ ip           : int  87540 105560 101424 94584 68413 93663 17059 121505 192967 143636 ...
##  $ app          : int  12 25 12 13 12 3 1 9 2 3 ...
##  $ device       : Factor w/ 100 levels "0","1","2","4",..: 2 2 2 2 2 2 2 2 3 2 ...
##  $ os           : Factor w/ 130 levels "0","1","2","3",..: 14 18 20 14 2 18 18 26 23 20 ...
##  $ channel      : Factor w/ 161 levels "3","4","5","13",..: 160 68 53 147 46 21 35 127 101 35 ...
##  $ is_attributed: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ days         : Factor w/ 4 levels "Friday","Monday",..: 4 4 4 4 2 2 2 4 1 1 ...
##  $ hour         : Factor w/ 24 levels "0","1","2","3",..: 10 14 19 5 10 2 2 11 10 13 ...
```

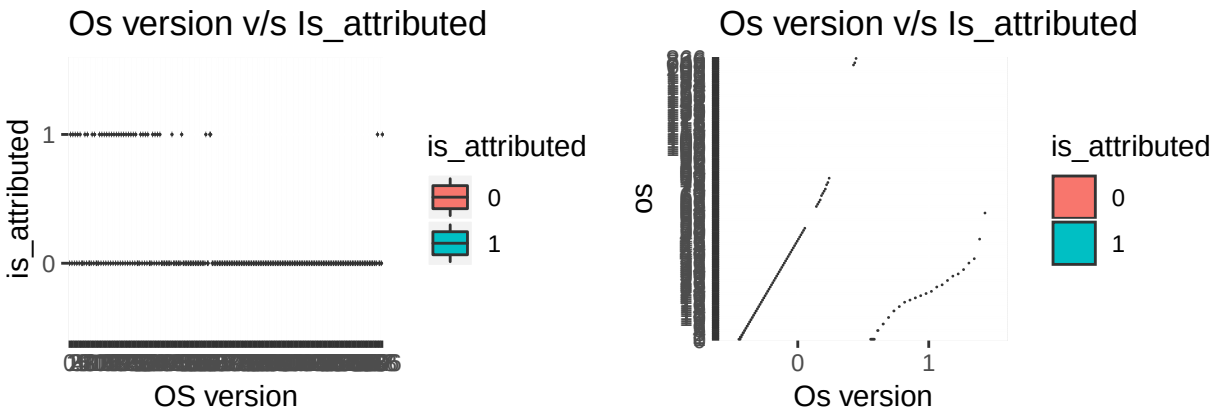
App was downloaded v/s App id for marketing

```
p1 <- ggplot(train,aes(x=is_attributed,y=app,fill=is_attributed))+
  geom_boxplot()+
  ggtitle("Application ID v/s Is_attributed")+
  xlab("App ID") +
  labs(fill = "is_attributed")

p2 <- ggplot(train,aes(x=app,fill=is_attributed))+
  geom_density()+facet_grid(is_attributed~.)+
  scale_x_continuous(breaks = c(0,50,100,200,300,400))+
  ggtitle("Application ID v/s Is_attributed")+
  xlab("App ID") +
  labs(fill = "is_attributed")

p3=ggplot(train,aes(x=is_attributed,y=app,fill=is_attributed))+
  geom_violin()+
  ggtitle("Application ID v/s Is_attributed")+
  xlab("App ID") +
  labs(fill = "is_attributed")


grid.arrange(p1,p2, p3, nrow=1,ncol=3)
```



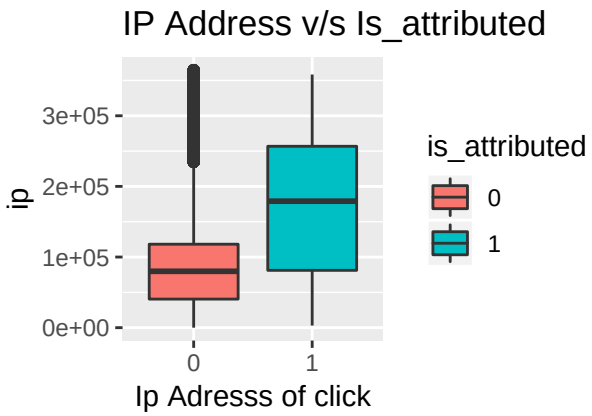
App was downloaded vs OS version id of user mobile phone

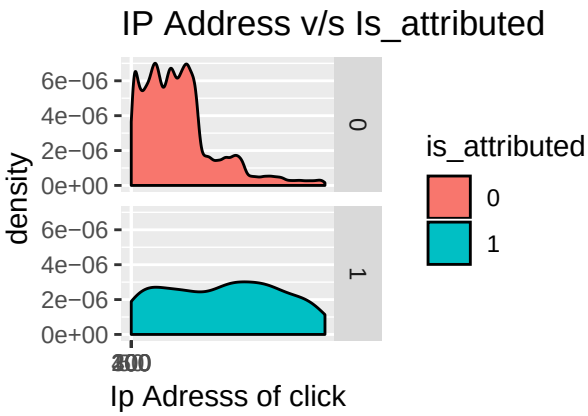
```
p4=ggplot(train,aes(x=os,y=is_attributed,fill=is_attributed))+
  geom_boxplot()+
  ggtitle("Os version v/s Is_attributed")+
  xlab("OS version") +
  labs(fill = "is_attributed")

p6=ggplot(train,aes(x=is_attributed,y=os,fill=is_attributed))+
  geom_violin()+
  ggtitle("Os version v/s Is_attributed")+
  xlab("Os version") +
  labs(fill = "is_attributed")
grid.arrange(p4, p6, nrow=2,ncol=2)
```

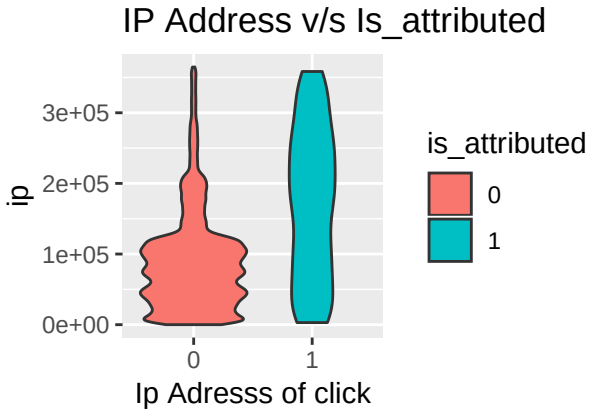


App was downloaded v/s ip address of click

```
p7=ggplot(train,aes(x=is_attributed,y=ip,fill=is_attributed))+
  geom_boxplot()+
  ggtitle("IP Address v/s Is_attributed")+
  xlab("Ip Adresss of click") +
  labs(fill = "is_attributed")


p8=ggplot(train,aes(x=ip,fill=is_attributed))+
  geom_density()+facet_grid(is_attributed~.)+
  scale_x_continuous(breaks = c(0,50,100,200,300,400))+
  ggtitle("IP Address v/s Is_attributed")+
  xlab("Ip Adresss of click") +
  labs(fill = "is_attributed")



p9=ggplot(train,aes(x=is_attributed,y=ip,fill=is_attributed))+
  geom_violin()+
  ggtitle("IP Address v/s Is_attributed")+
  xlab("Ip Adresss of click") +
  labs(fill = "is_attributed")

grid.arrange(p7,p8, p9, nrow=2,ncol=2)
```

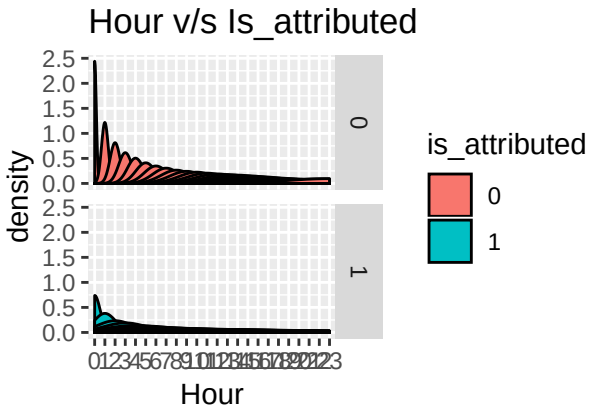


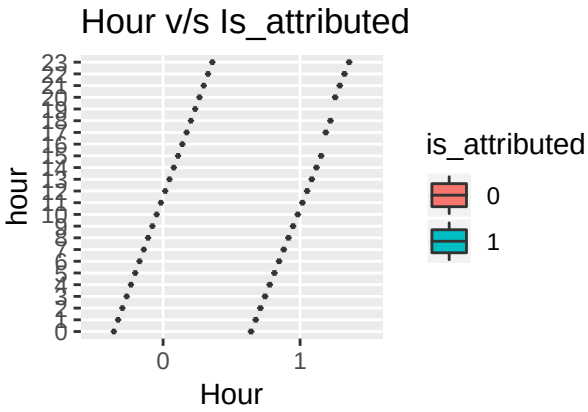


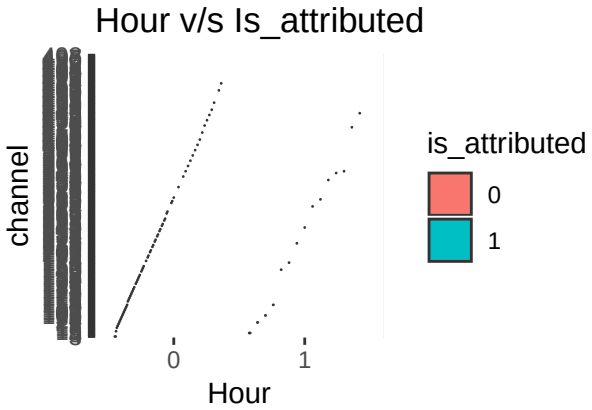


Does specific hour play any role in downloading

```
p16=ggplot(train,aes(x=hour,fill=is_attributed))+
  geom_density()+facet_grid(is_attributed~.)+
  ggtitle("Hour v/s Is_attributed ")+
  xlab("Hour") +
  labs(fill = "is_attributed")

p17=ggplot(train,aes(x=is_attributed,y=hour,fill=is_attributed))+
  geom_boxplot()+
  ggtitle("Hour v/s Is_attributed")+
  xlab("Hour") +
  labs(fill = "is_attributed")

p18=ggplot(train,aes(x=is_attributed,y=channel,fill=is_attributed))+
  geom_violin()+
  ggtitle("Hour v/s Is_attributed")+
  xlab("Hour") +
  labs(fill = "is_attributed")

grid.arrange(p16,p17, p18, nrow=2,ncol=2)
```



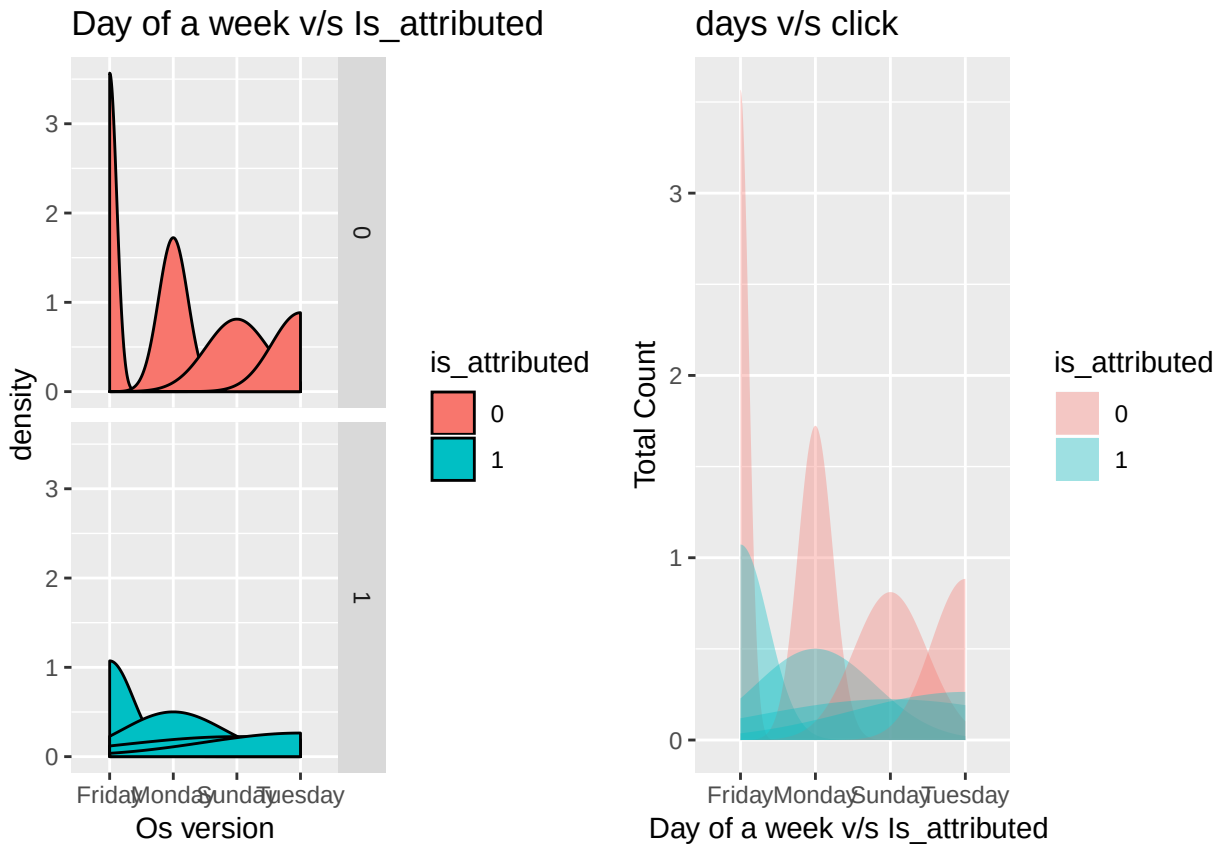




There is slight difference in both the distribution, we can say least important feature

Does Particular day play any role in downloading application?

```
p19=ggplot(train,aes(x=days,fill=is_attributed))+
  geom_density()+facet_grid(is_attributed~.)+
  ggtitle("Day of a week v/s Is_attributed ")+
  xlab("Os version") +
  labs(fill = "is_attributed")


p20=ggplot(train,aes(x=days,fill=is_attributed))+geom_density(col=NA,alpha=0.35)+
  ggtitle("days v/s click")+
  xlab("Day of a week v/s Is_attributed ") +
  ylab("Total Count") +
  labs(fill = "is_attributed")

grid.arrange(p19,p20, ncol=2)
```



Machine learning algorithms and cross validation

Now we will apply decission tree and cross validation to the training data set and check the accuracy.

Running the model using all the features in the data set

```
set.seed(1234)
library(caret)
cv.10 <- createMultiFolds(train$is_attributed, k = 10, times = 10)


ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 10,
                     index = cv.10)

set.seed(1234)

Model_CDT <- train(x = train[,-6], y = train[,6], method = "rpart",
                   tuneLength = 20,
                   trControl = ctrl)

PRE_VDTS=predict(Model_CDT$finalModel,data=train,type="class")
confusionMatrix(PRE_VDTS,train$is_attributed)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
```

```
##          0 99763   162
##          1    10    65
##
##                Accuracy : 0.9983
##                  95% CI : (0.998, 0.9985)
##     No Information Rate : 0.9977
##     P-Value [Acc > NIR] : 8.123e-05
##
##                   Kappa : 0.4298
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9999
##             Specificity : 0.2863
##          Pos Pred Value : 0.9984
##          Neg Pred Value : 0.8667
##              Prevalence : 0.9977
##          Detection Rate : 0.9976
##    Detection Prevalence : 0.9992
##       Balanced Accuracy : 0.6431
##
##        'Positive' Class : 0
##
```

```
varImp(Model_CDT)
```

```
## rpart variable importance
##
##         Overall
## channel  100.00
## os        78.42
## device    72.30
## ip        29.49
## app       23.34
## hour      18.73
## days       0.00
```

Even though overall accurcay is very high but specificity is too low

Running the model with selected features

We now check the importance of each feature in the training data set and then remove the unwanted features

```
str(train)
```

```
## 'data.frame':    100000 obs. of  8 variables:
##  $ ip           : int  87540 105560 101424 94584 68413 93663 17059 121505 192967 143636 ...
##  $ app          : int  12 25 12 13 12 3 1 9 2 3 ...
##  $ device       : Factor w/ 100 levels "0","1","2","4",..: 2 2 2 2 2 2 2 2 3 2 ...
##  $ os           : Factor w/ 130 levels "0","1","2","3",..: 14 18 20 14 2 18 18 26 23 20 ...
##  $ channel      : Factor w/ 161 levels "3","4","5","13",..: 160 68 53 147 46 21 35 127 101 35 ...
##  $ is_attributed: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ days         : Factor w/ 4 levels "Friday","Monday",..: 4 4 4 4 2 2 2 4 1 1 ...
##  $ hour         : Factor w/ 24 levels "0","1","2","3",..: 10 14 19 5 10 2 2 11 10 13 ...
```

```
train$days=NULL
train$hour=NULL
#train$os=NULL
#train$device=NULL
str(train)
```

```
## 'data.frame':    100000 obs. of  6 variables:
##  $ ip           : int  87540 105560 101424 94584 68413 93663 17059 121505 192967 143636 ...
##  $ app          : int  12 25 12 13 12 3 1 9 2 3 ...
##  $ device       : Factor w/ 100 levels "0","1","2","4",..: 2 2 2 2 2 2 2 2 3 2 ...
##  $ os           : Factor w/ 130 levels "0","1","2","3",..: 14 18 20 14 2 18 18 26 23 20 ...
##  $ channel      : Factor w/ 161 levels "3","4","5","13",..: 160 68 53 147 46 21 35 127 101 35 ...
##  $ is_attributed: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
set.seed(1234)

Model_CDT1 <- train(x = train[,-6], y = train[,6],
                    method = "rpart",
                    tuneLength = 20,
                    trControl = ctrl)

PRE_VDTS1=predict(Model_CDT1$finalModel,data=train,type="class")
confusionMatrix(PRE_VDTS1,train$is_attributed)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 99763   162
##          1    10    65
##
##                Accuracy : 0.9983
##                  95% CI : (0.998, 0.9985)
##     No Information Rate : 0.9977
##     P-Value [Acc > NIR] : 8.123e-05
##
##                   Kappa : 0.4298
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9999
##             Specificity : 0.2863
##          Pos Pred Value : 0.9984
##          Neg Pred Value : 0.8667
##              Prevalence : 0.9977
##          Detection Rate : 0.9976
##    Detection Prevalence : 0.9992
##       Balanced Accuracy : 0.6431
##
##        'Positive' Class : 0
##
```

Second model gives the same accuracy, however there is drastic change in specificity.

Dividing the data into test and training data

```
train$app=NULL
train$ip=NULL
set.seed(5000)
ind=createDataPartition(train$is_attributed,times=1,p=0.7,list=FALSE)
train_val=train[ind,]
test_val=train[-ind,]
```

Check the proportion and its the same

```
round(prop.table(table(train$is_attributed)*100),digits = 3)
```

```
##
##     0     1
## 0.998 0.002
```

```
round(prop.table(table(train_val$is_attributed)*100),digits = 3)
```

```
##
##     0     1
## 0.998 0.002
```

```
round(prop.table(table(test_val$is_attributed)*100),digits = 3)
```

```
##
##     0     1
## 0.998 0.002
```

Notice, how well caret divided the data into 70% to 30% ratio and also it make sure that no change in the proportion of target variable

Data Balancing using Smote

```
set.seed(1234)
smote_train = SMOTE(is_attributed ~ ., data  = train_val)
table(smote_train$is_attributed)
```

```
##
##   0   1
## 636 477
```

we now use Smote_train data set and use decision tree algorithm and check on the accuracy.

```
set.seed(1234)
cv.10 <- createMultiFolds(smote_train$is_attributed, k = 10, times = 10)
# Control
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 10,
                     index = cv.10)
##Train the data
```

```r
Model_CDT <- train(x = smote_train[,-4],
                   y = smote_train[,4],
                   method = "rpart",
                   tuneLength = 20,
                   trControl = ctrl)


PRE_VDTS=predict(Model_CDT$finalModel,newdata=test_val,type="class")
confusionMatrix(PRE_VDTS,test_val$is_attributed)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 28303    10
##          1  1628    58
##
##                Accuracy : 0.9454
##                  95% CI : (0.9428, 0.9479)
##     No Information Rate : 0.9977
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.062
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9456
##             Specificity : 0.8529
##          Pos Pred Value : 0.9996
##          Neg Pred Value : 0.0344
##              Prevalence : 0.9977
##          Detection Rate : 0.9435
##    Detection Prevalence : 0.9438
##       Balanced Accuracy : 0.8993
##
##        'Positive' Class : 0
##
```

We are able to complete decision tree with 0.94% accuracy, and specificity increased to 0.85% (Remember, drastic increase in specificty after data balance)

Conculsion

Hence using Decision tree machine learning algorithm provides the highest accuracy.

Reference Github Link: [https://github.com/sowmi121/Talkingdata-Adtracking-Fraud-deduction]