Click Traffic Fraud Detection in Mobile Application Advertisements

Naiara de Almeida Pantuza

21/01/2020

Project developed during the DSA BigData Analystics with R and Microsoft Azure course: available on the kaggle platform

DESCRIPTION:

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. China is the largest mobile market in the world and therefore suffers from huge volumes of fradulent traffic. TalkingData, China's largest independent big data service platform, covers over 70% of active mobile devices nationwide. They've built an IP blacklist and device blacklist. In this project, I am supose to build an algorithm that predicts whether a user will download an app after clicking a mobile app ad.

```
# Setting work directory
setwd("/home/naiara/Documentos/DataScience/FCD/BigDataRAzure/Project_Fraud_Detection")
```

All datasets were downloaded in kaggle. I chose to work with 10,000,000 lines to avoid processing and storage problems on my computer.

```
# Importing train and test datasets
Azure <- FALSE

if (Azure) {
  train <- maml.mapInputPort(1)
  test <- maml.mapInputPort(2)
}else {
  # Getting the first 10.000.000 rows of train dataset (train.zip file)
  train <- read.csv(unz("train.zip", "train.csv"))
  # Getting test dataset
  test <- read.csv(unz("test.zip", "test.csv"))
  }
```

Data cleaning and processing

Here I proceeded with cleaning and data processing.

```
# Checking if there are missing values.
# Only "attributed_time" has missing values, once tha apps may not be downloaded.
any(is.na(train[,-7]))
```

```
## [1] FALSE
```

```r
any(is.na(test))
```

```
## [1] FALSE
```

```r
# Modifying variable types for analysis:
# Converting categorical variables to factor
cnames <- c('ip', 'app', 'device', 'os', 'channel')
for (var in cnames) {
  train[,var] <- as.factor(train[,var])
  test[,var] <- as.factor(test[,var])
}

# Converting target variable "is_attributted" to factor
train$is_attributed <- as.factor(train$is_attributed)

# Converting temporal variables to datetime format
train$click_time <- as.POSIXct(train$click_time, tz = Sys.timezone())
train$attributed_time <- dplyr::na_if(train$attributed_time, "")
train$attributed_time <- as.POSIXct(train$attributed_time, tz = Sys.timezone())
test$click_time <- as.POSIXct(test$click_time, tz = Sys.timezone())

# Converting "click_id" from test dataset to factor
test$click_id <- as.factor(test$click_id)
```

Visual analysis and data exploration

Here I visualized training and test datasets and their summaries.

```r
# Train dataset head
head(train)
```

```
##   X     ip app device os channel          click_time attributed_time
## 1 1  83230   3      1 13     379 2017-11-06 14:32:21            <NA>
## 2 2  17357   3      1 19     379 2017-11-06 14:33:34            <NA>
## 3 3  35810   3      1 13     379 2017-11-06 14:34:12            <NA>
## 4 4  45745  14      1 13     478 2017-11-06 14:34:52            <NA>
## 5 5 161007   3      1 13     379 2017-11-06 14:35:08            <NA>
## 6 6  18787   3      1 16     379 2017-11-06 14:36:26            <NA>
##   is_attributed
## 1             0
## 2             0
## 3             0
## 4             0
## 5             0
## 6             0
```

```r
# Test dataset head
head(test)
```

```
##   click_id     ip app device os channel          click_time
## 1        0   5744   9      1  3     107 2017-11-10 04:00:00
## 2        1 119901   9      1  3     466 2017-11-10 04:00:00
## 3        2  72287  21      1 19     128 2017-11-10 04:00:00
## 4        3  78477  15      1 13     111 2017-11-10 04:00:00
```

```
## 5        4 123080  12      1 13     328 2017-11-10 04:00:00
## 6        5 110769  18      1 13     107 2017-11-10 04:00:00
```

```r
# Train dataset summary
str(train)
```

```
## 'data.frame':    10000000 obs. of  9 variables:
##  $ X              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ip             : Factor w/ 68740 levels "9","10","19",..: 18073 3760 7775 9962 43910 4119 22306 24
##  $ app            : Factor w/ 332 levels "0","1","2","3",..: 4 4 4 15 4 4 4 4 4 61 ...
##  $ device         : Factor w/ 940 levels "0","1","2","4",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ os             : Factor w/ 292 levels "0","1","2","3",..: 14 20 14 14 14 17 24 20 14 23 ...
##  $ channel        : Factor w/ 170 levels "0","3","4","5",..: 116 116 116 158 116 116 116 116 116 149
##  $ click_time     : POSIXct, format: "2017-11-06 14:32:21" "2017-11-06 14:33:34" ...
##  $ attributed_time: POSIXct, format: NA NA ...
##  $ is_attributed  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
summary(train)
```

```
##        X                   ip               app              device
##  Min.   :1.0e+00   73516  :  51711   12     :1291185   1      :9381146
##  1st Qu.:2.5e+06   73487  :  51215   2      :1202534   2      : 456617
##  Median :5.0e+06   5314   :  35073   15     :1181585   3032   : 104393
##  Mean   :5.0e+06   5348   :  35004   3      :1170412   0      :  46476
##  3rd Qu.:7.5e+06   53454  :  25381   9      : 966839   59     :   1618
##  Max.   :1.0e+07   105560 :  23289   18     : 917820   40     :    462
##                    (Other):9778327   (Other):3269625   (Other):   9288
##        os              channel          click_time
##  19     :2410148   245    : 793105   Min.   :2017-11-06 14:32:21
##  13     :2199778   134    : 630888   1st Qu.:2017-11-06 17:06:58
##  17     : 531695   259    : 469845   Median :2017-11-06 20:27:57
##  18     : 483602   477    : 412559   Mean   :2017-11-06 20:15:02
##  22     : 365576   121    : 402226   3rd Qu.:2017-11-06 23:16:56
##  10     : 285907   107    : 388035   Max.   :2017-11-07 00:12:03
##  (Other):3723294   (Other):6903342
##  attributed_time               is_attributed
##  Min.   :2017-11-06 16:00:47   0:9981283
##  1st Qu.:2017-11-06 19:59:39   1:  18717
##  Median :2017-11-06 23:34:47
##  Mean   :2017-11-07 00:10:04
##  3rd Qu.:2017-11-07 02:40:27
##  Max.   :2017-11-07 15:59:53
##  NA's   :9981283
```

```r
# Teste dataset summary
summary(test)
```

```
##     click_id               ip                  app               device
##  0      :        1   5348   :  182522   9      :2872176   1      :17360269
##  1      :        1   5314   :  162935   12     :2306083   2      : 1041975
##  2      :        1   73516  :   69089   3      :2201000   0      :  258152
##  3      :        1   73487  :   68866   2      :2060903   3      :   76117
##  4      :        1   53454  :   61503   18     :1923024   5      :    8279
##  5      :        1   114276 :   52649   15     :1079113   59     :    2775
##  (Other):18790463   (Other):18192905   (Other):6348170   (Other):   42902
##        os              channel           click_time
```

```
##  19      :4334532   107     : 1214650   Min.   :2017-11-10 04:00:00
##  13      :3959515   265     :  778244   1st Qu.:2017-11-10 05:27:21
##  17      : 960531   232     :  684938   Median :2017-11-10 10:03:52
##  18      : 870068   477     :  683101   Mean   :2017-11-10 09:43:00
##  22      : 773184   178     :  582524   3rd Qu.:2017-11-10 13:34:07
##  8       : 520612   153     :  566046   Max.   :2017-11-10 15:00:00
##  (Other):7372027   (Other):14280966
```

Including Plots

Check unique values for categorical variables from train dataset sample.

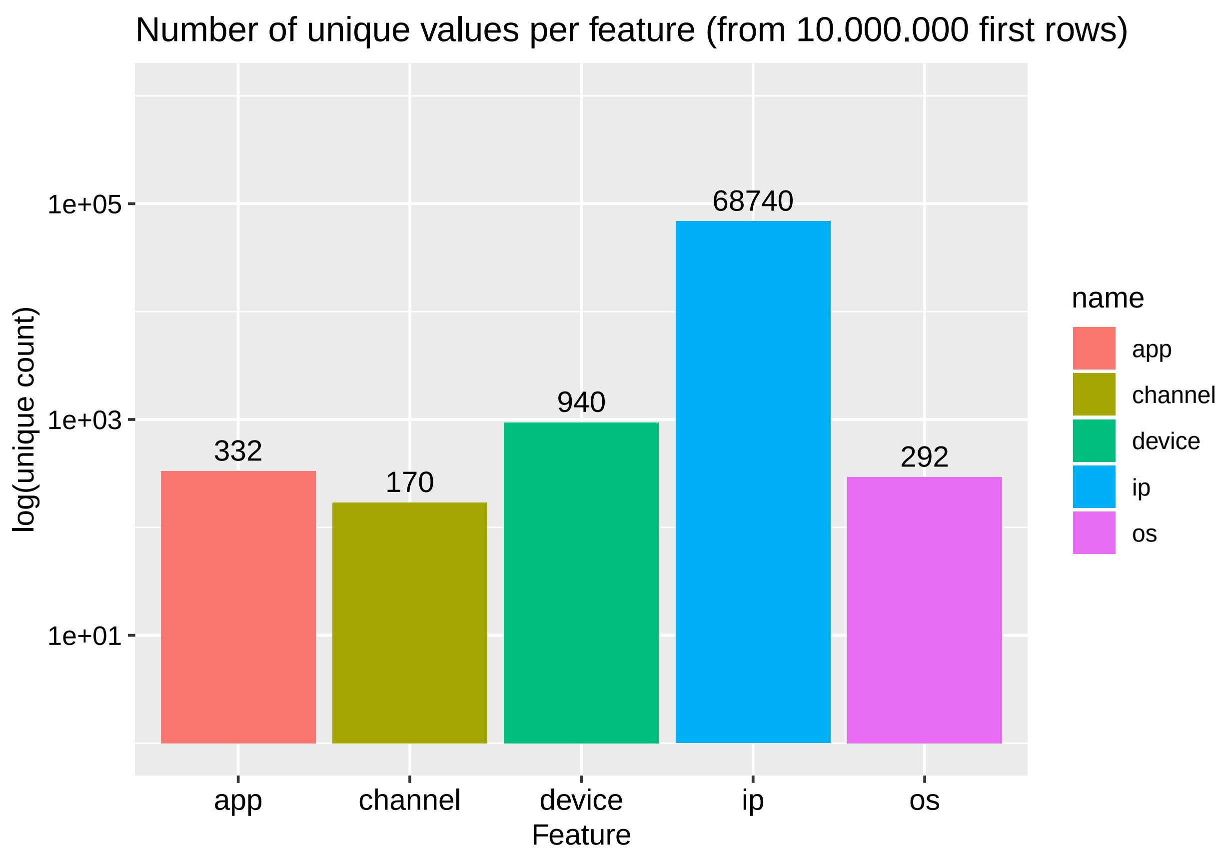
```r
# Loading required packages
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Get the number of unique values for categoriacal dependent variables
uniq <- sapply(train[,cnames], function(x) {return(length(unique(x)))})
uniq_data <- data.frame(value=uniq, name=names(uniq), row.names = NULL)
```

Plotting the number of unique values for categoriacal dependent variables

```r
# Bar plot of the number of single values
uniq_plot <- ggplot(uniq_data, aes(x=name, y=value, fill=name)) +
  geom_bar(stat = "identity") +
  ggtitle("Number of unique values per feature (from 10.000.000 first rows)") +
  xlab("Feature") +
  ylab("log(unique count)") +
  scale_y_log10(limits = c(1,1e6)) +
  geom_text(aes(label = sprintf("%d", value), y=value),  vjust = -0.5) +
  theme(axis.text.x = element_text(hjust = 0.5, size=11,color="black")) +
  theme(axis.text.y = element_text(hjust = 0.5, size=10,color="black"))

uniq_plot
```



As noted in the graph, the variables have many levels.

Summarizing attribute variables from downloaded application logs.

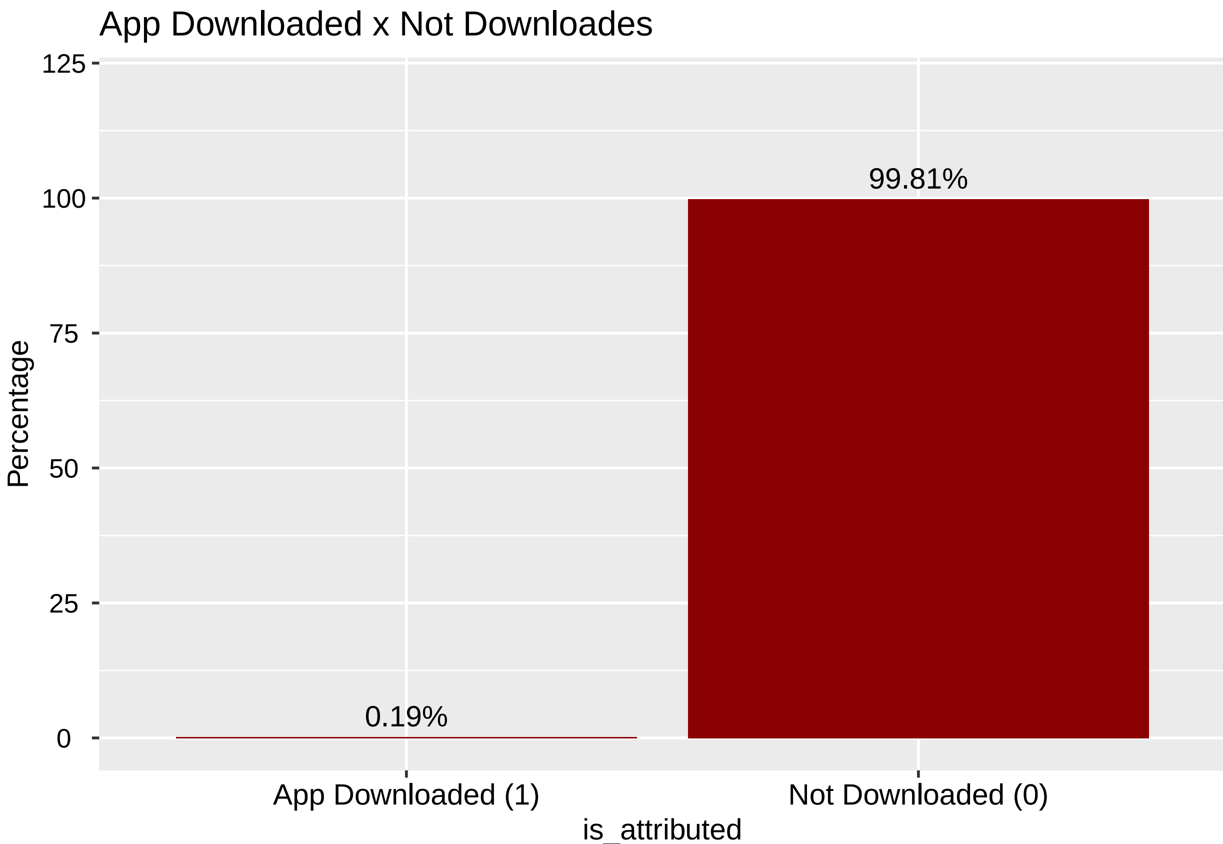
```
# Filtering and summarizing attribute variables from downloaded application logs
downloaded <- train[train$is_attributed == "1", c("attributed_time", "is_attributed")]
summary(downloaded)
```

```
##  attributed_time               is_attributed
##  Min.   :2017-11-06 16:00:47   0:    0
##  1st Qu.:2017-11-06 19:59:39   1:18717
##  Median :2017-11-06 23:34:47
##  Mean   :2017-11-07 00:10:04
##  3rd Qu.:2017-11-07 02:40:27
##  Max.   :2017-11-07 15:59:53
```

Plot proportion of apps download and not download

```
# Plotting proportion of apps download and not download
app_prop <- count(group_by(train[,c("app", "is_attributed")], is_attributed))
app_prop$n <- round(app_prop$n/sum(app_prop$n)*100, 2)
app_prop$is_attributed <- c("Not Downloaded (0)", "App Downloaded (1)")

appPropPlot <- ggplot(app_prop, aes(x=is_attributed, y=n)) +
  geom_bar(stat = "identity", fill="darkred") +
  ggtitle("App Downloaded x Not Downloades") +
  ylab("Percentage") +
  ylim(0,120) +
  geom_text(aes(label = sprintf("%.2f%%", n), y=n),  vjust = -0.5) +
  theme(axis.text.x = element_text(hjust = 0.5, size=11,color="black")) +
```

```
  theme(axis.text.y = element_text(hjust = 0.5, size=10,color="black"))

appPropPlot
```



As observed less than one percent of clicks were converted to downloads.

Analysing ips frequency

```
# Count 10 most clicked ips and their respective frequencies
temp <- as.data.frame(table(train$ip))
colnames(temp) <- c("ip", "count")
temp <- temp[order(temp$count, decreasing = TRUE),]
rownames(temp) <- 1:length(rownames(temp))
temp <- temp[1:10,]
temp
```

```
##        ip count
## 1   73516 51711
## 2   73487 51215
## 3    5314 35073
## 4    5348 35004
## 5   53454 25381
## 6  105560 23289
## 7  100275 23070
## 8  114276 22774
## 9  201182 22719
## 10 105475 22047
```

```r
# Count 10 most clicked downloaded ips and their respective frequencies
temp_downloaded <- as.data.frame(table(train[train$is_attributed == "1",]$ip))
colnames(temp_downloaded) <- c("downloaded_ip", "count_downloads")
temp_downloaded <- temp_downloaded[order(temp_downloaded$count, decreasing = TRUE),]
rownames(temp_downloaded) <- 1:length(rownames(temp_downloaded))
temp_downloaded <- temp_downloaded[1:10,]
temp_downloaded
```

```
##    downloaded_ip count_downloads
## 1          73487              56
## 2          73516              54
## 3           5314              26
## 4         201182              25
## 5           5348              24
## 6         100275              23
## 7         105475              22
## 8         105560              16
## 9          44744              15
## 10        123994              14
```

```r
# Checking coincide between 10 most clicked apss and the 10 most clicked downloaded apss
table(c(temp$ip,temp_downloaded$downloaded_ip))
```

```
##
##  1162  1173  9778 11715 16005 16012 21708 22868 22885 24864 26950 63053
##     2     2     1     1     2     2     2     2     2     1     1     2
```

Eight ips with the highest number of clicks were most downloaded.

Get summary of downloaded app dataset

```r
# Get statistic data from downloaded ip dataset
# Minimum, maximum, average, median, quartiles.
summary(as.integer(train[train$is_attributed == "1", "ip"]))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       5   16336   32828   33468   50388   68737
```

```r
# Count values
length(as.numeric(train[train$is_attributed == "1", "ip"]))
```

```
## [1] 18717
```

```r
# Count unique values
length(unique(as.integer(train[train$is_attributed == "1", "ip"])))
```

```
## [1] 16112
```

Analysing Most Popular IPs Getting the 300 most clicked ips, respective count of clickes and conversion rates (clicks converted to download).

Exibing table results:

```r
# Converting "is_attributed" to numeric temporarily to plotting
train$is_attributed <- as.numeric(train$is_attributed)-1

# Conversion Rates over Counts of 300 Most Popular IPs
# Exibing table results
count_rate <- train %>%
```

```
  group_by(ip) %>%
  summarise(click_count=n(), prop_downloaded = round(sum(is_attributed)/n(), 4)) %>%
  arrange(desc(click_count)) %>%
  slice(1:300) %>%
  data.frame()

count_rate
```

```
##          ip click_count prop_downloaded
## 1     73516       51711          0.0010
## 2     73487       51215          0.0011
## 3      5314       35073          0.0007
## 4      5348       35004          0.0007
## 5     53454       25381          0.0001
## 6    105560       23289          0.0007
## 7    100275       23070          0.0010
## 8    114276       22774          0.0001
## 9    201182       22719          0.0011
## 10   105475       22047          0.0010
## 11    95766       21966          0.0005
## 12    26995       19166          0.0005
## 13   209663       17605          0.0005
## 14    43793       15398          0.0008
## 15   137052       14840          0.0008
## 16    86767       14742          0.0004
## 17    17149       14673          0.0009
## 18   111025       14493          0.0001
## 19   138561       14119          0.0003
## 20   147957       14012          0.0010
## 21   114220       12818          0.0000
## 22    93054       12331          0.0009
## 23    92766       10904          0.0005
## 24    93021       10698          0.0010
## 25    92735       10534          0.0007
## 26   194308        9453          0.0006
## 27    93587        9450          0.0004
## 28    45745        9395          0.0011
## 29    44744        9232          0.0016
## 30    77048        9208          0.0014
## 31   114314        8831          0.0006
## 32    44725        8633          0.0012
## 33    84896        8594          0.0007
## 34   188387        8460          0.0011
## 35   114235        8431          0.0000
## 36    48240        8415          0.0002
## 37   194289        8239          0.0006
## 38   175837        8053          0.0004
## 39    48212        7995          0.0000
## 40   133522        7941          0.0013
## 41     3994        7884          0.0010
## 42   123994        7796          0.0018
## 43    48282        7774          0.0004
## 44    36150        7733          0.0004
## 45     4019        7719          0.0006
```

```
## 46   79881        7666          0.0005
## 47   43827        7652          0.0004
## 48   79857        7604          0.0005
## 49  178851        7425          0.0004
## 50   36183        7392          0.0004
## 51   44067        7383          0.0008
## 52   48170        7373          0.0000
## 53  147153        7319          0.0005
## 54   36213        7310          0.0010
## 55  108881        7265          0.0010
## 56  147046        7220          0.0008
## 57  125222        7184          0.0004
## 58  108913        7071          0.0006
## 59  147164        7034          0.0003
## 60    4052        6878          0.0009
## 61  147065        6849          0.0004
## 62  100393        6838          0.0010
## 63  105587        6701          0.0012
## 64  178873        6570          0.0002
## 65    5729        6557          0.0008
## 66   59125        6554          0.0002
## 67   79909        6541          0.0003
## 68  108341        6522          0.0005
## 69  108858        6456          0.0003
## 70    3964        6279          0.0008
## 71   79827        6207          0.0005
## 72   13634        6197          0.0008
## 73  119289        6177          0.0015
## 74  119369        5988          0.0008
## 75   90485        5981          0.0010
## 76  185670        5980          0.0005
## 77  108942        5975          0.0007
## 78  146001        5883          0.0003
## 79   52024        5814          0.0005
## 80   25071        5808          0.0005
## 81   52010        5704          0.0005
## 82    4989        5632          0.0018
## 83   37515        5411          0.0004
## 84   51992        5405          0.0009
## 85  109743        5375          0.0007
## 86   52043        5231          0.0008
## 87   84644        5195          0.0019
## 88   53715        5123          0.0004
## 89   84774        5104          0.0012
## 90   53964        5094          0.0004
## 91  119349        4984          0.0008
## 92   13597        4974          0.0004
## 93  100971        4959          0.0012
## 94   25097        4937          0.0014
## 95  197093        4871          0.0008
## 96   43855        4836          0.0010
## 97   90891        4742          0.0004
## 98   95820        4688          0.0006
## 99   90855        4678          0.0004
```

```
## 100 109723        4655          0.0002
## 101  85329        4613          0.0011
## 102  90509        4561          0.0009
## 103  25737        4451          0.0000
## 104  44673        4425          0.0014
## 105  44494        4383          0.0009
## 106  59395        4343          0.0005
## 107  92873        4331          0.0002
## 108  44458        4292          0.0009
## 109  97744        4206          0.0021
## 110 172483        4160          0.0012
## 111 133825        4157          0.0017
## 112  30587        4124          0.0010
## 113 114878        4090          0.0000
## 114 105910        4066          0.0000
## 115 172498        4066          0.0005
## 116 144604        4052          0.0000
## 117  85625        4035          0.0000
## 118 135992        4004          0.0000
## 119  91661        3944          0.0008
## 120  92673        3923          0.0003
## 121  91694        3910          0.0005
## 122 105433        3893          0.0005
## 123 105534        3890          0.0000
## 124  53960        3888          0.0003
## 125  87879        3869          0.0000
## 126  92852        3841          0.0005
## 127  70522        3822          0.0008
## 128  44555        3812          0.0010
## 129 151574        3805          0.0016
## 130 174548        3783          0.0019
## 131  75007        3776          0.0008
## 132 143418        3744          0.0000
## 133  25614        3732          0.0003
## 134  85644        3724          0.0008
## 135 105649        3704          0.0005
## 136 114678        3692          0.0011
## 137  97773        3690          0.0014
## 138  37948        3685          0.0000
## 139  25553        3682          0.0003
## 140  37919        3679          0.0014
## 141 105323        3677          0.0005
## 142  25679        3672          0.0003
## 143  87073        3659          0.0027
## 144  91712        3650          0.0003
## 145  97716        3637          0.0019
## 146 172522        3630          0.0006
## 147 105603        3629          0.0000
## 148  92712        3624          0.0011
## 149 105456        3616          0.0003
## 150 100182        3570          0.0006
## 151 192756        3551          0.0008
## 152 105569        3544          0.0011
## 153  91536        3521          0.0009
```

```
## 154  44527        3497          0.0014
## 155 172465        3493          0.0009
## 156  37972        3489          0.0009
## 157  25792        3468          0.0003
## 158  37892        3455          0.0009
## 159  37774        3453          0.0006
## 160  18703        3448          0.0000
## 161  91611        3446          0.0006
## 162  39756        3430          0.0009
## 163 105485        3416          0.0006
## 164  30564        3405          0.0003
## 165  25761        3391          0.0000
## 166  67197        3377          0.0006
## 167 151908        3364          0.0000
## 168   2095        3358          0.0015
## 169  53929        3355          0.0006
## 170  53479        3346          0.0003
## 171 114490        3340          0.0000
## 172 165085        3331          0.0003
## 173  76919        3307          0.0015
## 174 117867        3298          0.0012
## 175  25818        3291          0.0009
## 176 100212        3287          0.0003
## 177  97684        3284          0.0015
## 178  91574        3282          0.0006
## 179  67658        3259          0.0009
## 180 114461        3259          0.0009
## 181   2076        3256          0.0006
## 182  39782        3254          0.0003
## 183 105519        3242          0.0003
## 184  25705        3237          0.0003
## 185 109735        3233          0.0003
## 186 105292        3225          0.0000
## 187 202954        3185          0.0000
## 188  37813        3164          0.0003
## 189 114655        3137          0.0013
## 190  91885        3133          0.0013
## 191  85107        3125          0.0000
## 192 121472        3121          0.0019
## 193 205164        3121          0.0003
## 194 106460        3109          0.0003
## 195  54125        3089          0.0000
## 196 203048        3080          0.0000
## 197  25588        3073          0.0007
## 198  54157        3063          0.0003
## 199 176799        3059          0.0007
## 200  67628        3054          0.0003
## 201  25648        3032          0.0003
## 202  44536        2996          0.0003
## 203 118315        2987          0.0010
## 204 105861        2978          0.0007
## 205  24985        2967          0.0013
## 206 118339        2963          0.0007
## 207 176758        2961          0.0003
```

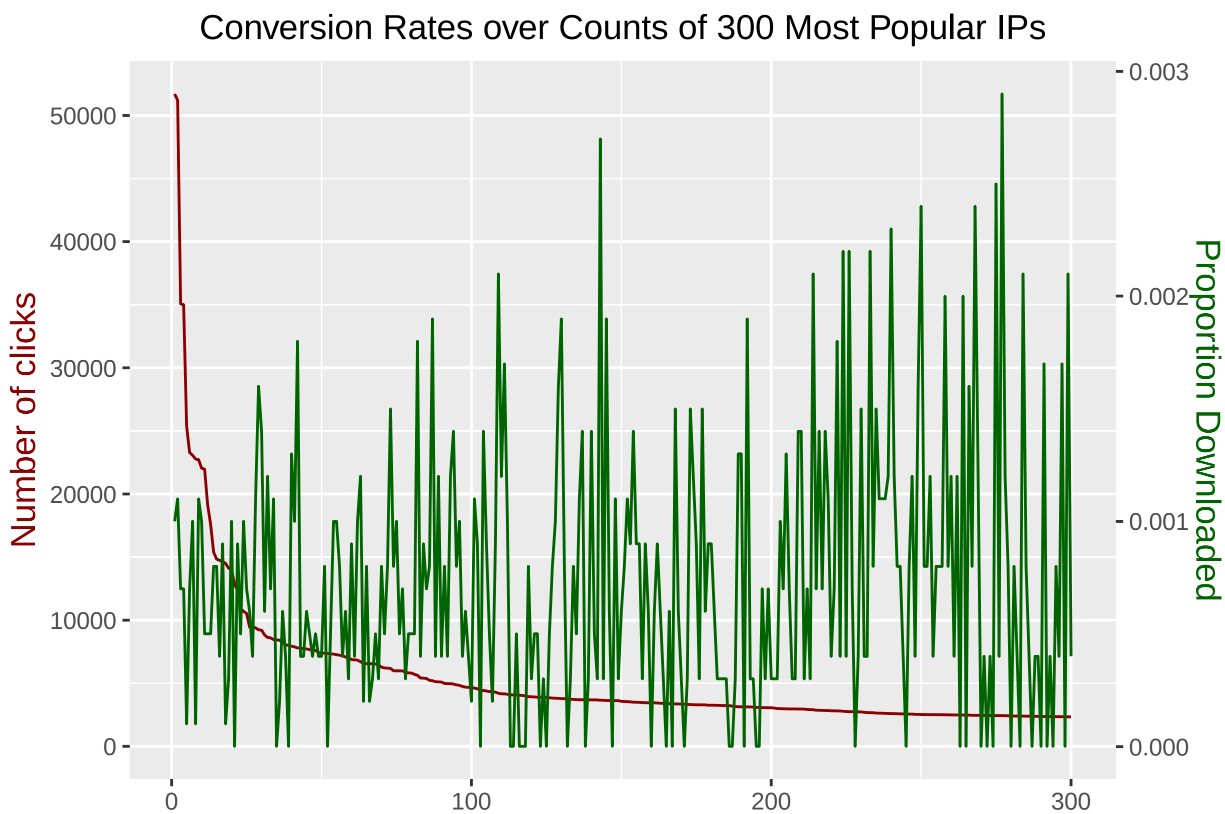
```
## 208 114904         2960           0.0003
## 209  91734         2955           0.0014
## 210 117898         2954           0.0014
## 211  12505         2950           0.0003
## 212 118229         2924           0.0007
## 213 118252         2914           0.0003
## 214  38219         2899           0.0021
## 215 118284         2863           0.0007
## 216  26814         2858           0.0014
## 217  38265         2850           0.0007
## 218  50169         2842           0.0014
## 219  62094         2837           0.0011
## 220  73671         2819           0.0004
## 221   4405         2809           0.0007
## 222  49383         2803           0.0018
## 223  48062         2800           0.0004
## 224  50512         2779           0.0022
## 225 118367         2766           0.0004
## 226  50482         2747           0.0022
## 227  76855         2746           0.0007
## 228 123788         2744           0.0000
## 229  50136         2733           0.0004
## 230  42139         2721           0.0015
## 231 100959         2697           0.0004
## 232  67439         2676           0.0004
## 233 111182         2672           0.0022
## 234  30614         2659           0.0008
## 235  38300         2635           0.0015
## 236  77085         2629           0.0011
## 237  99754         2626           0.0011
## 238  41232         2613           0.0011
## 239 106437         2607           0.0012
## 240  49431         2598           0.0023
## 241  99856         2595           0.0012
## 242 106598         2582           0.0008
## 243  49462         2573           0.0008
## 244  76885         2566           0.0004
## 245 109703         2555           0.0000
## 246 123703         2555           0.0008
## 247  99915         2552           0.0012
## 248  99769         2537           0.0004
## 249  99944         2533           0.0016
## 250 111153         2520           0.0024
## 251  37490         2518           0.0008
## 252  40289         2515           0.0008
## 253  44663         2509           0.0012
## 254 114816         2508           0.0004
## 255 100042         2507           0.0008
## 256 102467         2505           0.0008
## 257 100929         2504           0.0008
## 258 106200         2497           0.0020
## 259 145896         2493           0.0008
## 260 102235         2491           0.0012
## 261  12479         2485           0.0004
```

```
## 262 102264        2483          0.0012
## 263  67467        2480          0.0000
## 264  40077        2478          0.0020
## 265 137397        2474          0.0000
## 266 193448        2474          0.0016
## 267  44488        2460          0.0008
## 268   8208        2459          0.0024
## 269  74013        2458          0.0012
## 270  99897        2457          0.0000
## 271 106524        2457          0.0004
## 272  44615        2446          0.0000
## 273  52094        2445          0.0004
## 274  67037        2443          0.0000
## 275  73954        2442          0.0025
## 276 100088        2439          0.0004
## 277  49407        2437          0.0029
## 278 114795        2432          0.0012
## 279  44645        2416          0.0008
## 280  67606        2411          0.0000
## 281 193434        2398          0.0008
## 282  12524        2397          0.0004
## 283 191846        2395          0.0000
## 284 106223        2393          0.0021
## 285   8259        2389          0.0008
## 286  40372        2388          0.0004
## 287  62129        2385          0.0000
## 288 116425        2378          0.0004
## 289 193406        2378          0.0004
## 290 145934        2370          0.0000
## 291   8391        2363          0.0017
## 292  32453        2362          0.0000
## 293  44590        2359          0.0004
## 294 109644        2359          0.0000
## 295  40056        2353          0.0008
## 296 123759        2353          0.0004
## 297 106308        2347          0.0017
## 298   2564        2344          0.0000
## 299  50197        2336          0.0021
## 300 123654        2335          0.0004
```

Exibing plot results:

```r
# Obtain the coefficient so that both y axes are on the same scale
count_rate$num <- seq(1:nrow(count_rate))
MAX <- max(count_rate$click_count)
mx <- max(count_rate$prop_downloaded)
coef <- mx/MAX

# Disabling scientific notation in R
options(scipen = 999)

# Plotting Conversion Rates over Counts of 300 Most Popular IPs
ggplot(count_rate, aes(x=num)) +
  geom_line(aes(x=num, y=click_count), color="darkred") +
  geom_line(aes(x=num, y=prop_downloaded/coef), color="darkgreen") +
```

```
  ggtitle("Conversion Rates over Counts of 300 Most Popular IPs") +
  scale_y_continuous(
    # Features of the first axis
    name = "Number of clicks",
    # Add a second axis and specify its features
    sec.axis = sec_axis(~.*coef, name="Proportion Downloaded")
  ) +
  theme(
    axis.title.y = element_text(color = "darkred", size=13),
    axis.title.y.right = element_text(color = "darkgreen", size=13),
    plot.title = element_text(hjust = 0.5)
  ) +
  xlab("")
```



According to the graph, the number of clicks and conversion rate for ips are not significantly correlated.

Repeating the previous analysis for applications, operating systems, devices and channels.

Getting the 100 most clicked apps, respective count of clickes and conversion rates (clicks converted to download) and exibing table and plot results:

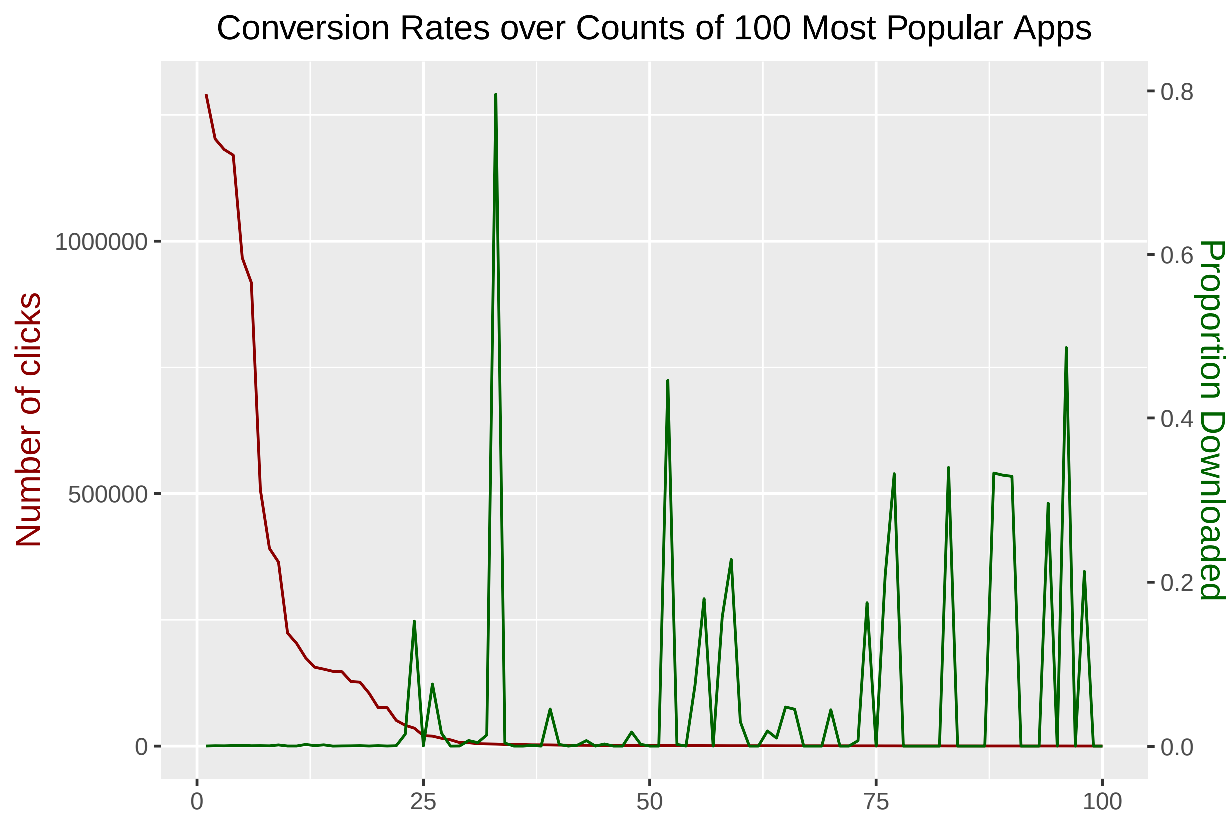
```
# Conversion Rates over Counts of 100 Most Popular Apps
count_rate <- train %>%
  group_by(app) %>%
  summarise(click_count=n(), prop_downloaded = round(sum(is_attributed)/n(), 4)) %>%
  arrange(desc(click_count)) %>%
  slice(1:100) %>%
  data.frame()

count_rate
```

```
##     app click_count prop_downloaded
## 1    12     1291185          0.0001
## 2     2     1202534          0.0004
## 3    15     1181585          0.0003
## 4     3     1170412          0.0006
## 5     9      966839          0.0009
## 6    18      917820          0.0004
## 7    14      507491          0.0005
## 8     1      391508          0.0003
## 9     8      364361          0.0014
## 10   21      223823          0.0001
## 11   13      203332          0.0001
## 12   20      174792          0.0020
## 13   24      156247          0.0006
## 14   11      152367          0.0015
## 15   23      148119          0.0000
## 16    6      147356          0.0002
## 17   64      127923          0.0003
## 18   26      126630          0.0005
## 19   25      104855          0.0001
## 20   27       76417          0.0005
## 21   28       76050          0.0001
## 22   17       50956          0.0004
## 23   10       41224          0.0146
## 24   19       35586          0.1526
## 25   22       20734          0.0003
## 26   29       19773          0.0758
## 27    5       15570          0.0158
## 28  151       12388          0.0000
## 29  160        6990          0.0001
## 30   36        6736          0.0068
## 31   32        4762          0.0042
## 32   82        4343          0.0136
## 33   35        3986          0.7958
## 34  150        3476          0.0037
## 35   80        3384          0.0000
## 36  183        2974          0.0000
## 37   58        2531          0.0008
## 38   88        2462          0.0000
## 39   45        2276          0.0453
## 40   46        1967          0.0020
## 41   33        1847          0.0000
## 42  208        1756          0.0011
## 43  103        1639          0.0067
## 44    4        1567          0.0000
## 45   74        1521          0.0026
## 46  109        1471          0.0000
## 47   38        1463          0.0000
## 48   55        1394          0.0172
## 49   65        1359          0.0022
## 50  215        1235          0.0000
## 51  536        1233          0.0000
## 52   72        1165          0.4464
## 53  110         940          0.0021
```

```
## 54   68          863          0.0000
## 55   83          860          0.0744
## 56   39          795          0.1799
## 57   56          788          0.0000
## 58  107          731          0.1573
## 59   66          676          0.2278
## 60   60          671          0.0298
## 61   94          667          0.0000
## 62   95          654          0.0000
## 63  122          650          0.0185
## 64  315          614          0.0098
## 65   52          566          0.0477
## 66  265          555          0.0450
## 67  134          549          0.0000
## 68  181          549          0.0000
## 69  419          538          0.0000
## 70  202          519          0.0443
## 71  231          518          0.0000
## 72  170          495          0.0000
## 73   86          455          0.0066
## 74   37          440          0.1750
## 75   53          439          0.0000
## 76  121          404          0.2079
## 77   84          379          0.3325
## 78  118          356          0.0000
## 79   91          352          0.0000
## 80  119          347          0.0000
## 81   93          344          0.0000
## 82   59          320          0.0000
## 83   50          294          0.3401
## 84   85          291          0.0000
## 85   49          283          0.0000
## 86  232          276          0.0000
## 87  100          261          0.0000
## 88   79          252          0.3333
## 89  145          251          0.3307
## 90  108          249          0.3293
## 91  185          243          0.0000
## 92  137          235          0.0000
## 93   47          228          0.0000
## 94   16          226          0.2965
## 95  266          223          0.0000
## 96  115          220          0.4864
## 97  363          200          0.0000
## 98   78          197          0.2132
## 99   81          191          0.0000
## 100  76          189          0.0000
```

```
# Obtain the coefficient so that both y axes are on the same scale
count_rate$num <- seq(1:nrow(count_rate))
MAX <- max(count_rate$click_count)
mx <- max(count_rate$prop_downloaded)
coef <- mx/MAX
```

```r
# Plotting Conversion Rates over Counts of 100 Most Popular Apps
ggplot(count_rate, aes(x=num)) +
  geom_line(aes(x=num, y=click_count), color="darkred") +
  geom_line(aes(x=num, y=prop_downloaded/coef), color="darkgreen") +
  ggtitle("Conversion Rates over Counts of 100 Most Popular Apps") +
  scale_y_continuous(
    # Features of the first axis
    name = "Number of clicks",
    # Add a second axis and specify its features
    sec.axis = sec_axis(~.*coef, name="Proportion Downloaded")
  ) +
  theme(
    axis.title.y = element_text(color = "darkred", size=13),
    axis.title.y.right = element_text(color = "darkgreen", size=13),
    plot.title = element_text(hjust = 0.5)
  ) +
  xlab("")
```



According to the graph, the number of clicks and conversion rate for apps are not significantly correlated.

Getting the 100 most clicked Operational Systems, respective count of clickes and conversion rates (clicks converted to download) and exibing table and plot results:

```r
# Conversion Rates over Counts of Most Popular Operational Systems
count_rate <- train %>%
  group_by(os) %>%
  summarise(click_count=n(), prop_downloaded = round(sum(is_attributed)/n(), 4)) %>%
  arrange(desc(click_count)) %>%
  slice(1:100) %>%
  data.frame()
```

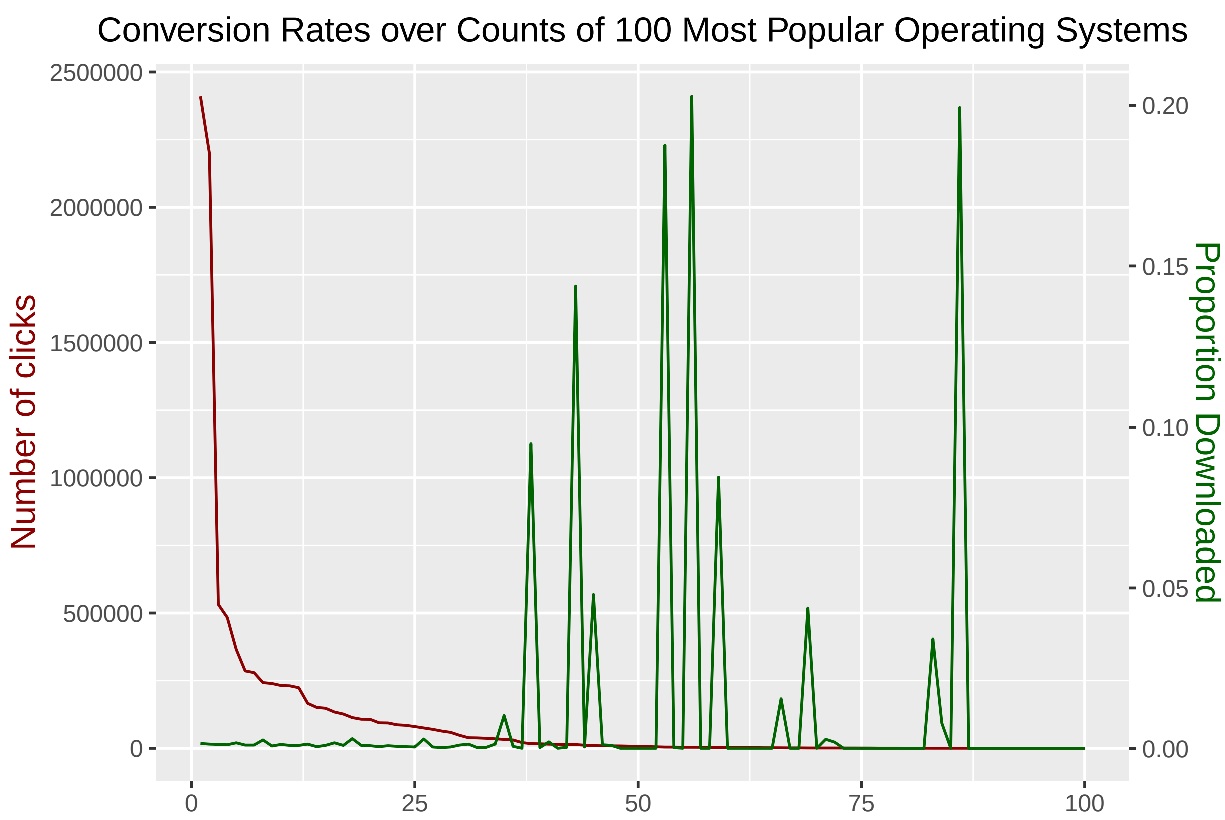
```
count_rate
```

```
##      os click_count prop_downloaded
## 1    19     2410148          0.0015
## 2    13     2199778          0.0013
## 3    17      531695          0.0012
## 4    18      483602          0.0011
## 5    22      365576          0.0017
## 6    10      285907          0.0010
## 7     8      279549          0.0010
## 8     6      242799          0.0026
## 9     9      239377          0.0007
## 10   25      232143          0.0012
## 11   15      230832          0.0009
## 12   20      223820          0.0009
## 13   16      166165          0.0013
## 14   37      151274          0.0005
## 15    3      147970          0.0009
## 16   14      134127          0.0017
## 17   41      126565          0.0009
## 18    1      113395          0.0030
## 19  607      107442          0.0009
## 20   12      107005          0.0008
## 21   27       94188          0.0005
## 22   35       93578          0.0008
## 23   23       86880          0.0006
## 24   32       84824          0.0005
## 25   53       80319          0.0004
## 26   28       75093          0.0029
## 27   11       70077          0.0004
## 28   47       63726          0.0002
## 29   30       58910          0.0004
## 30   26       47956          0.0010
## 31   31       38943          0.0013
## 32    2       38347          0.0002
## 33   36       36951          0.0003
## 34   49       35023          0.0013
## 35   40       32891          0.0102
## 36    4       30729          0.0006
## 37   42       21096          0.0000
## 38    0       17102          0.0947
## 39   43       16637          0.0002
## 40   34       15222          0.0020
## 41   58       14784          0.0000
## 42   46       14528          0.0003
## 43   24       13790          0.1437
## 44    7       11410          0.0004
## 45   38        9802          0.0478
## 46   48        9237          0.0011
## 47   44        8951          0.0009
## 48    5        8672          0.0000
## 49   65        7788          0.0000
## 50   56        7574          0.0000
```

```
## 51   70         6253           0.0000
## 52   79         5422           0.0000
## 53   21         4449           0.1875
## 54   66         4363           0.0002
## 55   64         3898           0.0000
## 56   29         3853           0.2027
## 57   52         3788           0.0000
## 58   55         3684           0.0000
## 59   50         3073           0.0843
## 60   77         3072           0.0000
## 61   39         3067           0.0000
## 62   73         3061           0.0000
## 63   97         2562           0.0000
## 64   62         1996           0.0000
## 65   63         1925           0.0000
## 66   76         1756           0.0154
## 67   98         1435           0.0000
## 68   90         1421           0.0000
## 69   59         1123           0.0436
## 70   57         1079           0.0000
## 71   96         1072           0.0028
## 72  109         1036           0.0019
## 73  100          913           0.0000
## 74   85          890           0.0000
## 75   60          865           0.0000
## 76   83          765           0.0000
## 77   74          475           0.0000
## 78  102          426           0.0000
## 79   69          407           0.0000
## 80  112          352           0.0000
## 81   71          339           0.0000
## 82   80          335           0.0000
## 83   84          294           0.0340
## 84   81          258           0.0078
## 85   87          241           0.0000
## 86   67          236           0.1992
## 87  106          236           0.0000
## 88   78          233           0.0000
## 89   92          225           0.0000
## 90  118          209           0.0000
## 91  111          192           0.0000
## 92   72          189           0.0000
## 93  107          171           0.0000
## 94   54          169           0.0000
## 95   68          165           0.0000
## 96  110          165           0.0000
## 97  155          154           0.0000
## 98  137          145           0.0000
## 99   89          142           0.0000
## 100 132          139           0.0000
```

```r
# Obtain the coefficient so that both y axes are on the same scale
count_rate$num <- seq(1:nrow(count_rate))
MAX <- max(count_rate$click_count)
```

```r
mx <- max(count_rate$prop_downloaded)
coef <- mx/MAX

# Plotting Conversion Rates over Counts of 100 Most Popular Operating Systems
ggplot(count_rate, aes(x=num)) +
  geom_line(aes(x=num, y=click_count), color="darkred") +
  geom_line(aes(x=num, y=prop_downloaded/coef), color="darkgreen") +
  ggtitle("Conversion Rates over Counts of 100 Most Popular Operating Systems") +
  scale_y_continuous(
    # Features of the first axis
    name = "Number of clicks",
    # Add a second axis and specify its features
    sec.axis = sec_axis(~.*coef, name="Proportion Downloaded")
  ) +
  theme(
    axis.title.y = element_text(color = "darkred", size=13),
    axis.title.y.right = element_text(color = "darkgreen", size=13),
    plot.title = element_text(hjust = 0.5)
  ) +
  xlab("")
```



According to the graph, the number of clicks and conversion rate for Operational Systems are not significantly correlated.

Getting the 100 most clicked devices, respective count of clickes and conversion rates (clicks converted to download) and exibing table and plot results:

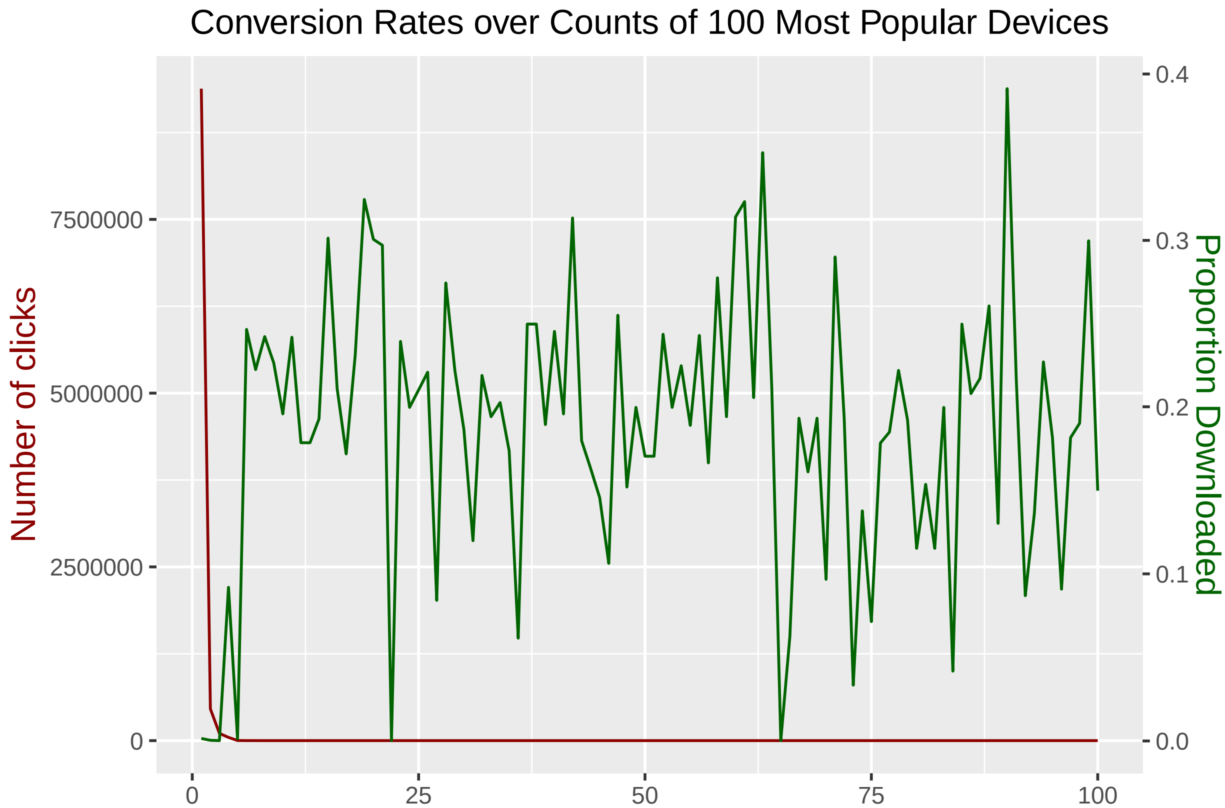
```r
# Conversion Rates and Counts of Most Popular by device
count_rate <- train %>%
  group_by(device) %>%
```

```r
  summarise(click_count=n(), prop_downloaded = round(sum(is_attributed)/n(), 4)) %>%
  arrange(desc(click_count)) %>%
  slice(1:100) %>%
  data.frame()

count_rate
```

```
##     device click_count prop_downloaded
## 1        1     9381146          0.0013
## 2        2      456617          0.0002
## 3     3032      104393          0.0000
## 4        0       46476          0.0920
## 5       59        1618          0.0012
## 6       40         462          0.2468
## 7        6         458          0.2227
## 8       16         334          0.2425
## 9       18         247          0.2267
## 10      33         204          0.1961
## 11      21         190          0.2421
## 12     154         151          0.1788
## 13    3033         151          0.1788
## 14      37         145          0.1931
## 15      30         126          0.3016
## 16      46         123          0.2114
## 17     114         122          0.1721
## 18       7         121          0.2314
## 19      88         117          0.3248
## 20     109         113          0.3009
## 21      67         111          0.2973
## 22     748         103          0.0000
## 23     136          96          0.2396
## 24      78          95          0.2000
## 25      82          95          0.2105
## 26      97          95          0.2211
## 27     374          95          0.0842
## 28      50          91          0.2747
## 29     211          81          0.2222
## 30      60          75          0.1867
## 31     203          75          0.1200
## 32      56          73          0.2192
## 33      96          72          0.1944
## 34     220          69          0.2029
## 35     343          69          0.1739
## 36      20          65          0.0615
## 37       4          60          0.2500
## 38     101          60          0.2500
## 39     214          58          0.1897
## 40      89          57          0.2456
## 41     231          51          0.1961
## 42     299          51          0.3137
## 43      73          50          0.1800
## 44     234          49          0.1633
## 45     168          48          0.1458
## 46     102          47          0.1064
```

```
## 47     103           47           0.2553
## 48     276           46           0.1522
## 49     137           45           0.2000
## 50      25           41           0.1707
## 51     127           41           0.1707
## 52     210           41           0.2439
## 53      11           40           0.2000
## 54      76           40           0.2250
## 55      36           37           0.1892
## 56     558           37           0.2432
## 57      42           36           0.1667
## 58     100           36           0.2778
## 59     189           36           0.1944
## 60     124           35           0.3143
## 61      95           34           0.3235
## 62     263           34           0.2059
## 63     381           34           0.3529
## 64     251           33           0.2121
## 65     736           33           0.0000
## 66     395           32           0.0625
## 67       9           31           0.1935
## 68      14           31           0.1613
## 69      75           31           0.1935
## 70     229           31           0.0968
## 71     350           31           0.2903
## 72     379           31           0.1935
## 73     479           30           0.0333
## 74     362           29           0.1379
## 75      52           28           0.0714
## 76      53           28           0.1786
## 77      54           27           0.1852
## 78     190           27           0.2222
## 79      51           26           0.1923
## 80      61           26           0.1154
## 81     208           26           0.1538
## 82     334           26           0.1154
## 83      68           25           0.2000
## 84      26           24           0.0417
## 85     396           24           0.2500
## 86     581           24           0.2083
## 87     118           23           0.2174
## 88     129           23           0.2609
## 89     160           23           0.1304
## 90     230           23           0.3913
## 91     338           23           0.2174
## 92     422           23           0.0870
## 93      19           22           0.1364
## 94     106           22           0.2273
## 95     169           22           0.1818
## 96     447           22           0.0909
## 97     486           22           0.1818
## 98     417           21           0.1905
## 99     132           20           0.3000
## 100    240           20           0.1500
```

```r
# Obtain the coefficient so that both y axes are on the same scale
count_rate$num <- seq(1:nrow(count_rate))
MAX <- max(count_rate$click_count)
mx <- max(count_rate$prop_downloaded)
coef <- mx/MAX

# Plotting
ggplot(count_rate, aes(x=num)) +
  geom_line(aes(x=num, y=click_count), color="darkred") +
  geom_line(aes(x=num, y=prop_downloaded/coef), color="darkgreen") +
  ggtitle("Conversion Rates over Counts of 100 Most Popular Devices") +
  scale_y_continuous(
    # Features of the first axis
    name = "Number of clicks",
    # Add a second axis and specify its features
    sec.axis = sec_axis(~.*coef, name="Proportion Downloaded")
  ) +
  theme(
    axis.title.y = element_text(color = "darkred", size=13),
    axis.title.y.right = element_text(color = "darkgreen", size=13),
    plot.title = element_text(hjust = 0.5)
  ) +
  xlab("")
```



According to the graph, the number of clicks and conversion rate for devices are not significantly correlated.

Getting the 100 most clicked channels, respective count of clickes and conversion rates (clicks converted to download) and exibing table and plot results:

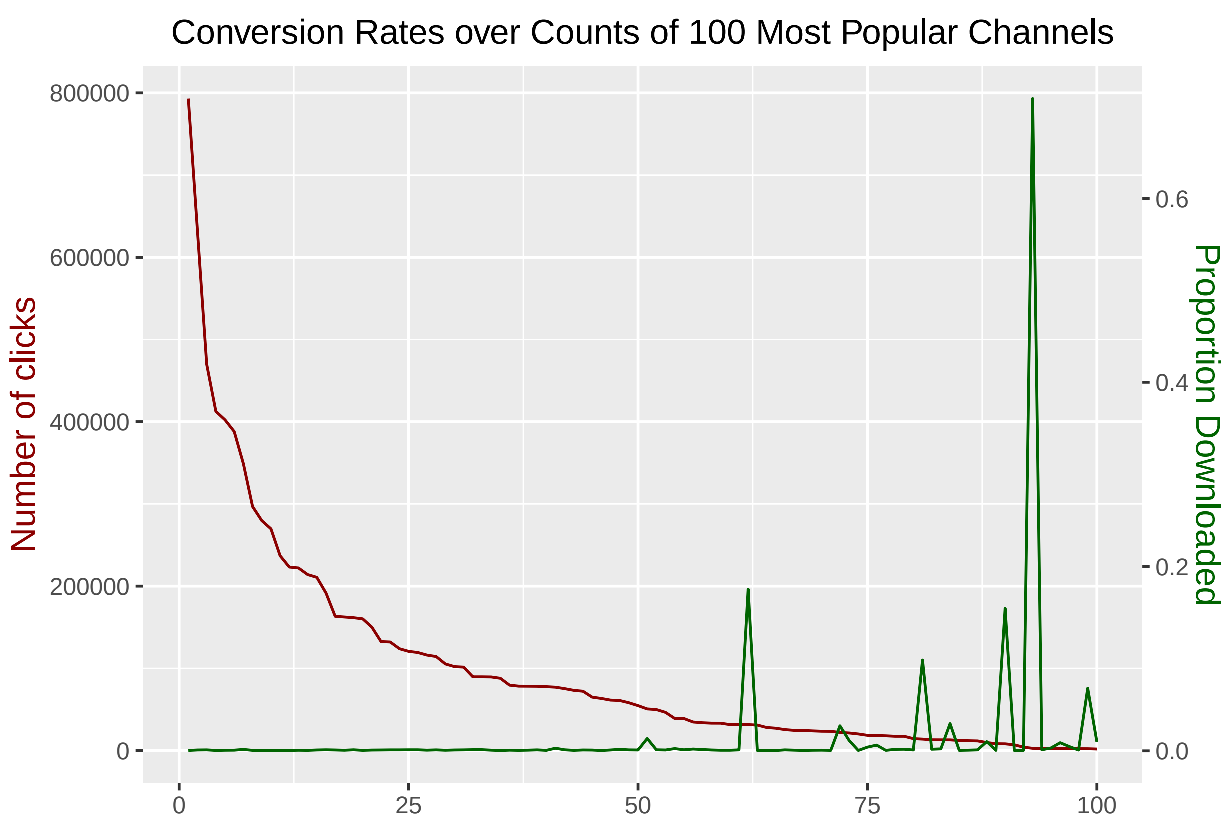
```r
# Conversion Rates over Counts of Most Popular Channels
count_rate <- train %>%
  group_by(channel) %>%
  summarise(click_count=n(), prop_downloaded = round(sum(is_attributed)/n(), 4)) %>%
  arrange(desc(click_count)) %>%
  slice(1:100) %>%
  data.frame()

count_rate
```

```
##    channel click_count prop_downloaded
## 1      245      793105          0.0001
## 2      134      630888          0.0006
## 3      259      469845          0.0007
## 4      477      412559          0.0001
## 5      121      402226          0.0003
## 6      107      388035          0.0004
## 7      145      348862          0.0012
## 8      153      296832          0.0002
## 9      205      279720          0.0002
## 10     178      269720          0.0001
## 11     265      236949          0.0002
## 12     128      223205          0.0001
## 13     140      222096          0.0003
## 14     459      214060          0.0002
## 15     442      210687          0.0006
## 16     215      191618          0.0008
## 17     122      163312          0.0006
## 18     280      162425          0.0003
## 19     379      161608          0.0008
## 20     135      160215          0.0002
## 21     439      150074          0.0005
## 22     105      132516          0.0006
## 23     480      132030          0.0007
## 24     469      123869          0.0007
## 25     219      120673          0.0008
## 26     489      119261          0.0008
## 27     137      116107          0.0004
## 28     435      114372          0.0007
## 29     328      105392          0.0003
## 30     452      102074          0.0006
## 31     409      101462          0.0007
## 32     334       89689          0.0009
## 33     424       89682          0.0009
## 34     115       89512          0.0004
## 35     125       87895          0.0000
## 36     130       79525          0.0004
## 37     237       78336          0.0002
## 38     258       78247          0.0004
## 39     401       78163          0.0007
## 40       3       77703          0.0002
## 41     377       77054          0.0025
## 42     173       75319          0.0008
## 43     212       73234          0.0002
```

```
## 44        19       72148          0.0006
## 45       315       64963          0.0005
## 46       364       63361          0.0000
## 47       463       61366          0.0006
## 48       232       60869          0.0013
## 49       234       58134          0.0007
## 50       349       54608          0.0006
## 51       347       50608          0.0131
## 52       266       49840          0.0008
## 53       386       46399          0.0006
## 54       244       39003          0.0021
## 55       481       38936          0.0008
## 56       319       34687          0.0016
## 57       466       33805          0.0011
## 58       412       33307          0.0006
## 59       278       33270          0.0003
## 60       111       31583          0.0003
## 61       430       31557          0.0007
## 62       213       31500          0.1754
## 63       326       31000          0.0001
## 64       123       28072          0.0002
## 65       417       27184          0.0000
## 66       236       25486          0.0007
## 67       497       24556          0.0004
## 68       400       24432          0.0001
## 69       124       23929          0.0003
## 70       371       23466          0.0004
## 71       211       23380          0.0002
## 72       113       22058          0.0269
## 73       243       21376          0.0110
## 74       116       20162          0.0001
## 75       110       18514          0.0037
## 76       478       18278          0.0059
## 77       118       17917          0.0002
## 78       325       17344          0.0013
## 79       487       17344          0.0014
## 80       402       14470          0.0006
## 81        21       13911          0.0984
## 82       445       13104          0.0014
## 83       376       13037          0.0018
## 84       343       12989          0.0293
## 85       242       12197          0.0002
## 86        17       11958          0.0004
## 87       150       11685          0.0007
## 88       317        9752          0.0097
## 89       467        8287          0.0002
## 90       101        8143          0.1545
## 91       457        6987          0.0001
## 92       446        4077          0.0002
## 93       274        2703          0.7088
## 94       406        2615          0.0008
## 95       373        2514          0.0028
## 96       330        2482          0.0085
## 97       449        2352          0.0043
```

```
## 98      262        2208          0.0005
## 99      210        2200          0.0677
## 100     411        1829          0.0093
```

```r
# Obtain the coefficient so that both y axes are on the same scale
count_rate$num <- seq(1:nrow(count_rate))
MAX <- max(count_rate$click_count)
mx <- max(count_rate$prop_downloaded)
coef <- mx/MAX

# Plotting
ggplot(count_rate, aes(x=num)) +
  geom_line(aes(x=num, y=click_count), color="darkred") +
  geom_line(aes(x=num, y=prop_downloaded/coef), color="darkgreen") +
  ggtitle("Conversion Rates over Counts of 100 Most Popular Channels") +
  scale_y_continuous(
    # Features of the first axis
    name = "Number of clicks",
    # Add a second axis and specify its features
    sec.axis = sec_axis(~.*coef, name="Proportion Downloaded")
  ) +
  theme(
    axis.title.y = element_text(color = "darkred", size=13),
    axis.title.y.right = element_text(color = "darkgreen", size=13),
    plot.title = element_text(hjust = 0.5)
  ) +
  xlab("")
```



According to the graph, the number of clicks and conversion rate for channels are not significantly correlated.

Time Patterns

The analysis of temporal patterns will be made with the sample provided by the kaggle. The first lines of the dataset are organized by time and therefore are not random. Thus, they are inappropriate for detecting temporal patterns.

Getting and treating the random sample train dataset: converting datetime variables to POSIXct format and rounding "click_time" variable's hours.

```
# Importing a random training dataset sample for time pattern analysis
sampleTrain <- read.csv("train_sample.csv")

head(sampleTrain)
```

```
##       ip app device os channel          click_time attributed_time
## 1  87540  12      1 13     497 2017-11-07 09:30:38
## 2 105560  25      1 17     259 2017-11-07 13:40:27
## 3 101424  12      1 19     212 2017-11-07 18:05:24
## 4  94584  13      1 13     477 2017-11-07 04:58:08
## 5  68413  12      1  1     178 2017-11-09 09:00:09
## 6  93663   3      1 17     115 2017-11-09 01:22:13
##   is_attributed
## 1             0
## 2             0
## 3             0
## 4             0
## 5             0
## 6             0
```

```
#convert click_time and attributed_time to time series
sampleTrain$attributed_time <- dplyr::na_if(sampleTrain$attributed_time, "")
sampleTrain$attributed_time <- as.POSIXct(sampleTrain$attributed_time, tz = Sys.timezone())
sampleTrain$click_time <- as.POSIXct(sampleTrain$click_time, tz = Sys.timezone())

# Convert "is_attributed to numeric
sampleTrain$is_attributed <- as.numeric(sampleTrain$is_attributed)

#round the time to nearest hour
sampleTrain$click_round <- lubridate::round_date(sampleTrain$click_time, "hour")
```

Temporal anaylis: checking for hourly patterns by tables and plotting graphs.

```
# Checking for hourly patterns
count_rate <- sampleTrain[,c("click_round", "is_attributed")] %>%
  group_by(click_round) %>%
  summarise(click_count=n(), conversion_rate = round(sum(is_attributed)/n(), 4)) %>%
  data.frame()

head(count_rate, 100)
```

```
##            click_round click_count conversion_rate
## 1  2017-11-06 16:00:00         684          0.0000
## 2  2017-11-06 17:00:00         921          0.0022
## 3  2017-11-06 18:00:00         523          0.0000
## 4  2017-11-06 19:00:00         367          0.0000
## 5  2017-11-06 20:00:00         251          0.0000
## 6  2017-11-06 21:00:00         235          0.0085
```

```
## 7  2017-11-06 22:00:00          396           0.0000
## 8  2017-11-06 23:00:00          955           0.0021
## 9  2017-11-07 00:00:00         1649           0.0018
## 10 2017-11-07 01:00:00         1928           0.0031
## 11 2017-11-07 02:00:00         1715           0.0035
## 12 2017-11-07 03:00:00         1692           0.0053
## 13 2017-11-07 04:00:00         1861           0.0005
## 14 2017-11-07 05:00:00         1843           0.0016
## 15 2017-11-07 06:00:00         1700           0.0024
## 16 2017-11-07 07:00:00         1540           0.0019
## 17 2017-11-07 08:00:00         1609           0.0050
## 18 2017-11-07 09:00:00         1480           0.0014
## 19 2017-11-07 10:00:00         1645           0.0018
## 20 2017-11-07 11:00:00         1894           0.0042
## 21 2017-11-07 12:00:00         1631           0.0018
## 22 2017-11-07 13:00:00         1714           0.0023
## 23 2017-11-07 14:00:00         1763           0.0011
## 24 2017-11-07 15:00:00         1572           0.0019
## 25 2017-11-07 16:00:00         1462           0.0014
## 26 2017-11-07 17:00:00          939           0.0000
## 27 2017-11-07 18:00:00          514           0.0000
## 28 2017-11-07 19:00:00          297           0.0000
## 29 2017-11-07 20:00:00          242           0.0041
## 30 2017-11-07 21:00:00          218           0.0000
## 31 2017-11-07 22:00:00          417           0.0072
## 32 2017-11-07 23:00:00         1028           0.0049
## 33 2017-11-08 00:00:00         1692           0.0000
## 34 2017-11-08 01:00:00         1800           0.0044
## 35 2017-11-08 02:00:00         1688           0.0024
## 36 2017-11-08 03:00:00         1858           0.0032
## 37 2017-11-08 04:00:00         1788           0.0011
## 38 2017-11-08 05:00:00         1871           0.0043
## 39 2017-11-08 06:00:00         1607           0.0044
## 40 2017-11-08 07:00:00         1660           0.0024
## 41 2017-11-08 08:00:00         1667           0.0018
## 42 2017-11-08 09:00:00         1518           0.0007
## 43 2017-11-08 10:00:00         1884           0.0016
## 44 2017-11-08 11:00:00         1781           0.0034
## 45 2017-11-08 12:00:00         1937           0.0021
## 46 2017-11-08 13:00:00         1877           0.0016
## 47 2017-11-08 14:00:00         2030           0.0059
## 48 2017-11-08 15:00:00         1965           0.0005
## 49 2017-11-08 16:00:00         1564           0.0019
## 50 2017-11-08 17:00:00         1008           0.0000
## 51 2017-11-08 18:00:00          516           0.0019
## 52 2017-11-08 19:00:00          352           0.0028
## 53 2017-11-08 20:00:00          269           0.0037
## 54 2017-11-08 21:00:00          257           0.0039
## 55 2017-11-08 22:00:00          459           0.0000
## 56 2017-11-08 23:00:00          997           0.0000
## 57 2017-11-09 00:00:00         1599           0.0044
## 58 2017-11-09 01:00:00         1700           0.0018
## 59 2017-11-09 02:00:00         1609           0.0012
## 60 2017-11-09 03:00:00         1669           0.0000
```

```
## 61 2017-11-09 04:00:00        1963          0.0020
## 62 2017-11-09 05:00:00        2157          0.0023
## 63 2017-11-09 06:00:00        1962          0.0015
## 64 2017-11-09 07:00:00        1849          0.0022
## 65 2017-11-09 08:00:00        1571          0.0019
## 66 2017-11-09 09:00:00        1592          0.0057
## 67 2017-11-09 10:00:00        1684          0.0018
## 68 2017-11-09 11:00:00        1813          0.0022
## 69 2017-11-09 12:00:00        1748          0.0023
## 70 2017-11-09 13:00:00        1845          0.0027
## 71 2017-11-09 14:00:00        1943          0.0010
## 72 2017-11-09 15:00:00        1829          0.0027
## 73 2017-11-09 16:00:00         737          0.0000
```

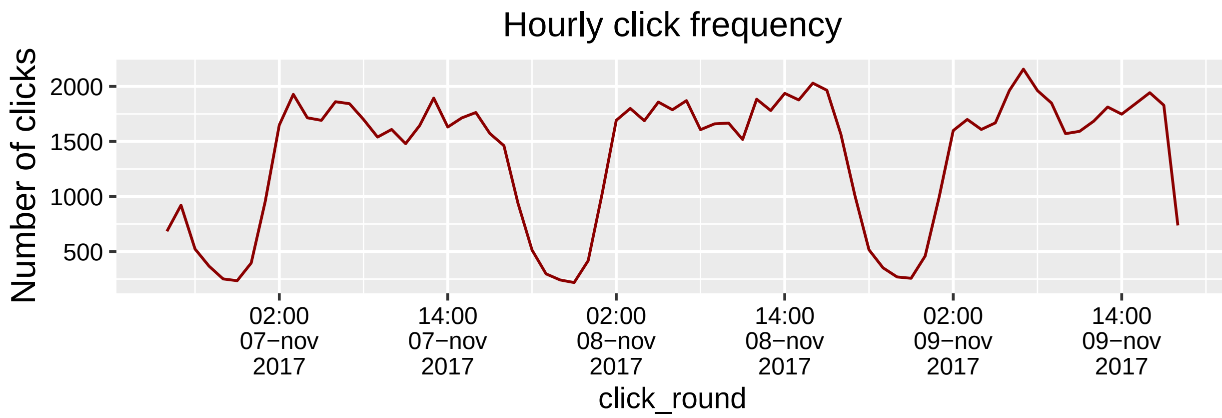
```r
# Plotting
require(gridExtra)
```

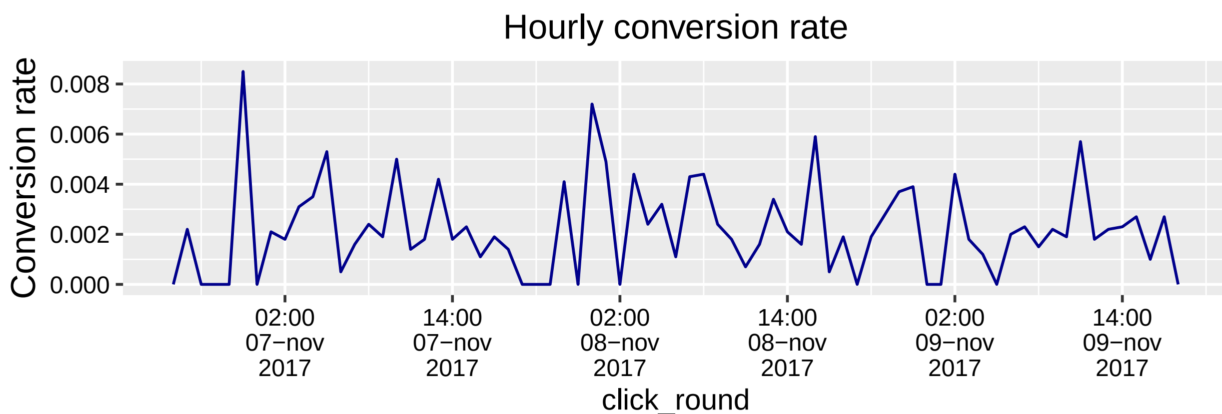
```
## Loading required package: gridExtra
```

```
##
## Attaching package: 'gridExtra'
```

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

```r
# Plotting hourly clicks
plot1 <- ggplot(count_rate, aes(x=click_round)) +
  geom_line(aes(x=click_round, y=click_count), color="darkred") +
  ggtitle("Hourly click frequency") +
  scale_y_continuous(name = "Number of clicks") +
  scale_x_datetime(labels = scales::date_format("%H:%M\n%d-%b\n%Y"), date_breaks = "12 hours") +
  theme(
    axis.title.y = element_text(color = "black", size=13),
    axis.text.x= element_text(color = "black"),
    axis.text.y= element_text(color = "black"),
    plot.title = element_text(hjust = 0.5)
  )

# Plotting hourly conversion rate
plot2 <- ggplot(count_rate, aes(x=click_round)) +
  geom_line(aes(x=click_round, y=conversion_rate), color="darkblue") +
  ggtitle("Hourly conversion rate") +
  scale_y_continuous(name = "Conversion rate") +
  scale_x_datetime(labels = scales::date_format("%H:%M\n%d-%b\n%Y"), date_breaks = "12 hours",
                   limits = ) +
  theme(
    axis.title.y = element_text(color = "black", size=13),
    axis.text.x= element_text(color = "black"),
    axis.text.y= element_text(color = "black"),
    plot.title = element_text(hjust = 0.5)
  )

grid.arrange(plot1, plot2, nrow = 2)
```



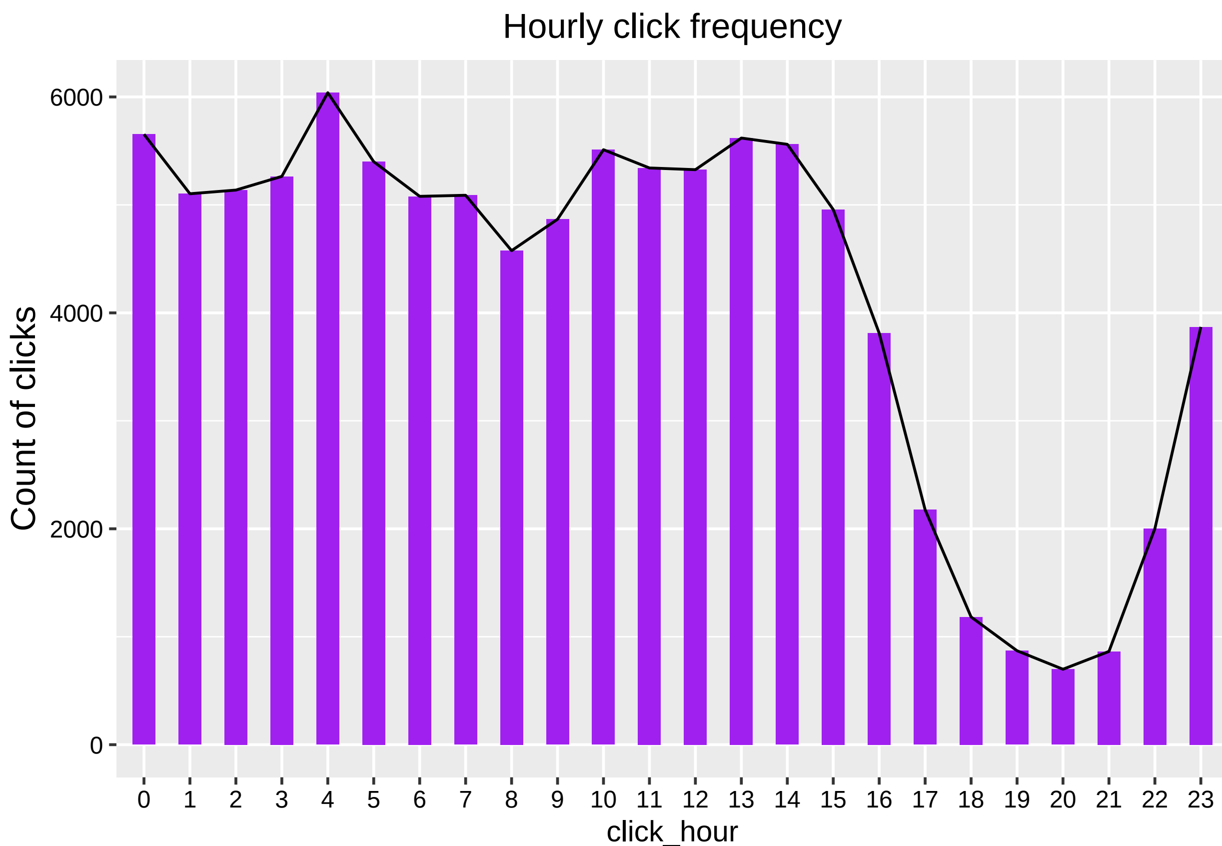


Looking at the graph, I noticed that the click count and conversion rate for downloads do not appear to be significantly correlated.

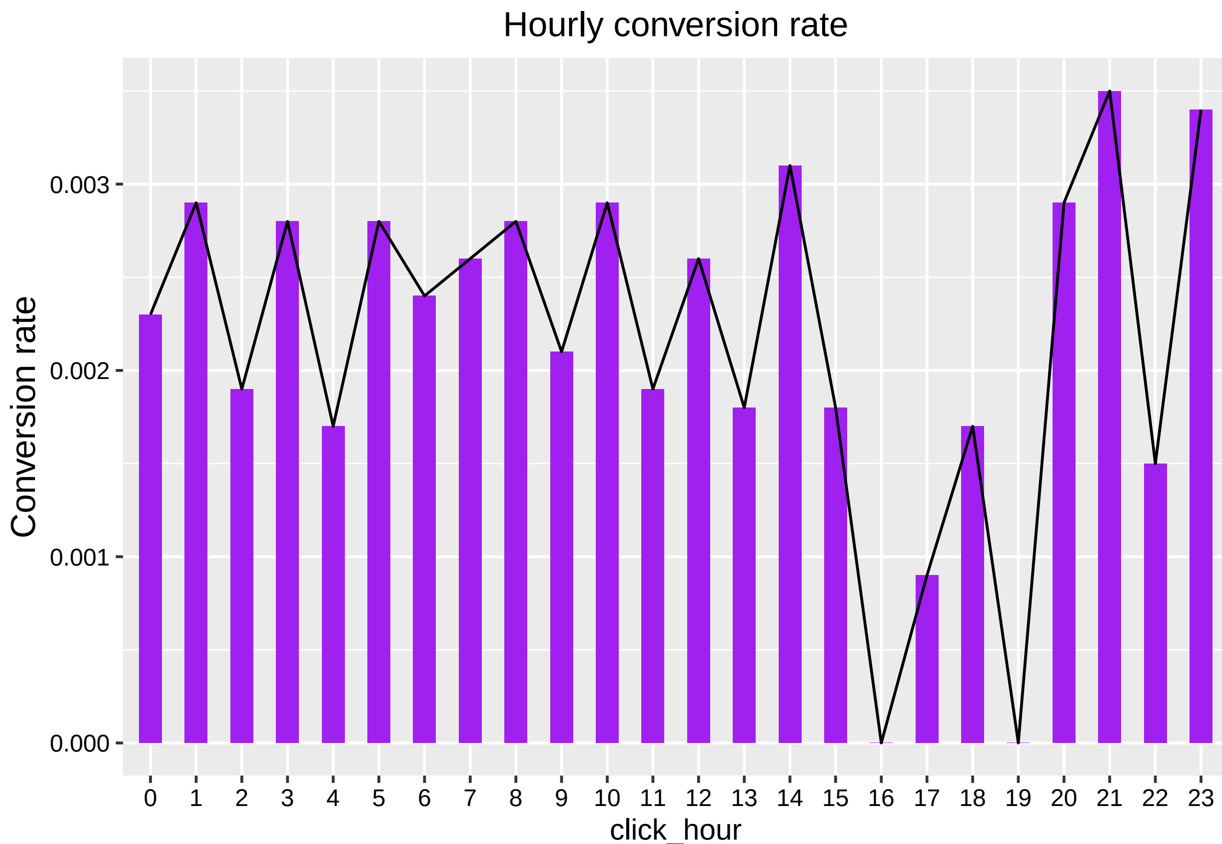
Extracting hour from "click_time" variable and getting the number of clicks and conversion rate by hour. Exibing table and graphs:

```
# Extract hour from "click_time" and add to sample train dataset
sampleTrain$click_hour <- as.factor(lubridate::hour(sampleTrain$click_time))

# Getting number of clicks by hour
count_rate <- sampleTrain[,c("click_hour", "is_attributed")] %>%
  group_by(click_hour) %>%
  summarise(click_count=n(), conversion_rate = round(mean(is_attributed), 4)) %>%
  data.frame()

# Visualizing data.frame
View(count_rate)

# Plotting number of clicks by hour
ggplot(count_rate, aes(x=click_hour, y=click_count)) +
  geom_bar(stat = "identity", fill="purple", width = 0.5) +
  geom_line( aes(x=as.numeric(click_hour), y=click_count)) +
  ggtitle("Hourly click frequency") +
  scale_y_continuous(name = "Count of clicks") +
  theme(
    axis.title.y = element_text(color = "black", size=13),
    axis.text.x= element_text(color = "black"),
    axis.text.y= element_text(color = "black"),
```

```
    plot.title = element_text(hjust = 0.5)
  )
```



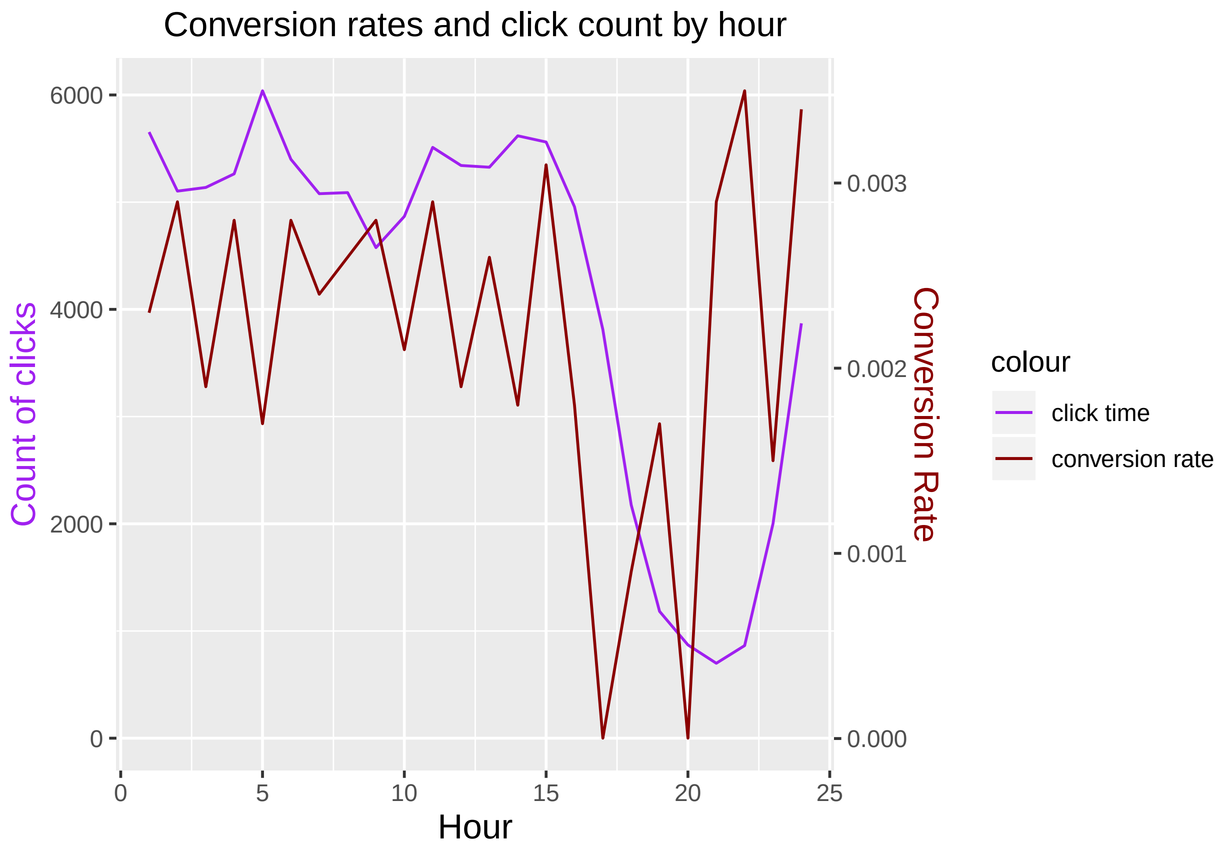

```
# Plotting hourly conversion rate
ggplot(count_rate, aes(x=click_hour, y=conversion_rate)) +
  geom_bar(stat = "identity", fill="purple", width = 0.5) +
  geom_line( aes(x=as.numeric(click_hour), y=conversion_rate)) +
  ggtitle("Hourly conversion rate") +
  scale_y_continuous(name = "Conversion rate") +
  theme(
    axis.title.y = element_text(color = "black", size=13),
    axis.text.x= element_text(color = "black"),
    axis.text.y= element_text(color = "black"),
    plot.title = element_text(hjust = 0.5)
  )
```



Apparently, fewer clicks occur between 27 and 22 hours and fewer downloads at 16, 17, 19 and 22 hours.

Plotting conversion rate and count of clicks to check if both variables correlate:

```
# Plotting conversion rate and count of clicks to check if the variables correlate
# Obtain the coefficient so that both y axes are on the same scale
MAX <- max(count_rate$click_count)
mx <- max(count_rate$conversion_rate)
coef <- mx/MAX

# Plotting
ggplot(count_rate, aes(x=num)) +
  geom_line(aes(x=as.numeric(click_hour), y=click_count, color="click time")) +
  geom_line(aes(x=as.numeric(click_hour), y=conversion_rate/coef, color="conversion rate")) +
  ggtitle("Conversion rates and click count by hour") +
  scale_y_continuous(
    # Features of the first axis
    name = "Count of clicks",
    # Add a second axis and specify its features
    sec.axis = sec_axis(~.*coef, name="Conversion Rate")
  ) + xlab("Hour") +
  theme(
    axis.title.y = element_text(color = "purple", size=13),
    axis.title.x = element_text(color = "black", size=13),
    axis.title.y.right = element_text(color = "darkred", size=13),
    plot.title = element_text(hjust = 0.5)
  ) +
  scale_color_manual(values = c("purple", "darkred"))
```



According to the graph, the variables appear to have a weak correlation.

Here, I check the time difference between click_time and it's conversion to download (attributed_time) on sample train dataset: visualizing and summarizing

```
# Checking the time difference between clicking add and download it
sampleTrain$timeDiff <- hms::as_hms(sampleTrain$attributed_time - sampleTrain$click_time)

# Checking first rows and the time passaed between click and download
head(sampleTrain[sampleTrain$is_attributed == 1,], 15)
```

```
##           ip app device os channel          click_time     attributed_time
## 285  224120  19      0 29     213 2017-11-08 02:22:13 2017-11-08 02:22:38
## 482  272894  10      1  7     113 2017-11-08 06:10:05 2017-11-08 06:10:37
## 1209  79001  19      0  0     213 2017-11-07 09:54:22 2017-11-07 11:59:05
## 1342 131029  19      0  0     343 2017-11-09 10:58:46 2017-11-09 11:52:01
## 1413  40352  19      0  0     213 2017-11-07 22:19:03 2017-11-08 01:55:02
## 1667  48733  35      1 18     274 2017-11-07 12:25:50 2017-11-07 13:10:30
## 1772 330861  35      1 22      21 2017-11-08 18:54:44 2017-11-08 22:39:52
## 1918 309576   5      1 32     113 2017-11-09 08:47:51 2017-11-09 08:47:55
## 3915 220571  71      1 25       3 2017-11-08 04:35:21 2017-11-08 04:37:46
## 3993 240051  35      1 19      21 2017-11-08 08:07:13 2017-11-08 09:46:42
## 4301 110652  19     16  0     213 2017-11-09 08:15:34 2017-11-09 09:30:19
## 4425 252612   5      1 31     113 2017-11-07 20:21:11 2017-11-07 20:21:42
## 4565  48072  19     21 24     213 2017-11-07 05:17:29 2017-11-07 06:49:01
## 4604  12506  62      1 19      21 2017-11-08 05:56:57 2017-11-08 08:56:58
## 4608 184467  35      1 30     274 2017-11-07 22:29:06 2017-11-08 00:16:14
##      is_attributed         click_round click_hour timeDiff
## 285              1 2017-11-08 02:00:00          2 00:00:25
```

```
## 482              1 2017-11-08 06:00:00         6 00:00:32
## 1209             1 2017-11-07 10:00:00         9 02:04:43
## 1342             1 2017-11-09 11:00:00        10 00:53:15
## 1413             1 2017-11-07 22:00:00        22 03:35:59
## 1667             1 2017-11-07 12:00:00        12 00:44:40
## 1772             1 2017-11-08 19:00:00        18 03:45:08
## 1918             1 2017-11-09 09:00:00         8 00:00:04
## 3915             1 2017-11-08 05:00:00         4 00:02:25
## 3993             1 2017-11-08 08:00:00         8 01:39:29
## 4301             1 2017-11-09 08:00:00         8 01:14:45
## 4425             1 2017-11-07 20:00:00        20 00:00:31
## 4565             1 2017-11-07 05:00:00         5 01:31:32
## 4604             1 2017-11-08 06:00:00         5 03:00:01
## 4608             1 2017-11-07 22:00:00        22 01:47:08
```

```
# Getting time passed summary
as.data.frame(lapply(summary(as.numeric(sampleTrain[sampleTrain$is_attributed == 1,
                                                    "timeDiff"])), hms::as_hms))
```

```
##       Min.   X1st.Qu.   Median            Mean   X3rd.Qu.     Max.
## 1 00:00:02 00:00:52.5 00:03:18 01:14:59.572687 01:21:27.5 12:52:21
```

Here I check the time difference between click_time and it's conversion to download (attributed_time) on first 10.000.000 rows of train dataset:

```
# Checking the time difference between clicking add and download it
# on the first 10.000.000 rows of train datset
train$timeDiff <- hms::as_hms(train$attributed_time - train$click_time)

# Summary
as.data.frame(lapply(summary(as.numeric(train[train$is_attributed == 1,
                                              "timeDiff"])), hms::as_hms))
```

```
##       Min. X1st.Qu.   Median            Mean X3rd.Qu.     Max.
## 1 00:00:00 00:01:26 00:25:03 03:48:02.893733 06:34:14 23:52:38
```

The difference varias from 0 to almost 24 hours (one day) on first rows of train dataset.

Feature Selection

In this script the selection of characteristics is performed for the creation of the model. I used randomForest algorithm to measure variables' importance. So, I converted categorical dependent variables to integer. Due to my machine's processing and memory limitations, I worked with random samples from the training and test datasets to build and evaluate the model.

```
# Set seed
set.seed(123)

# Feature selection using randomForest package
# load the library
library(randomForest)
```

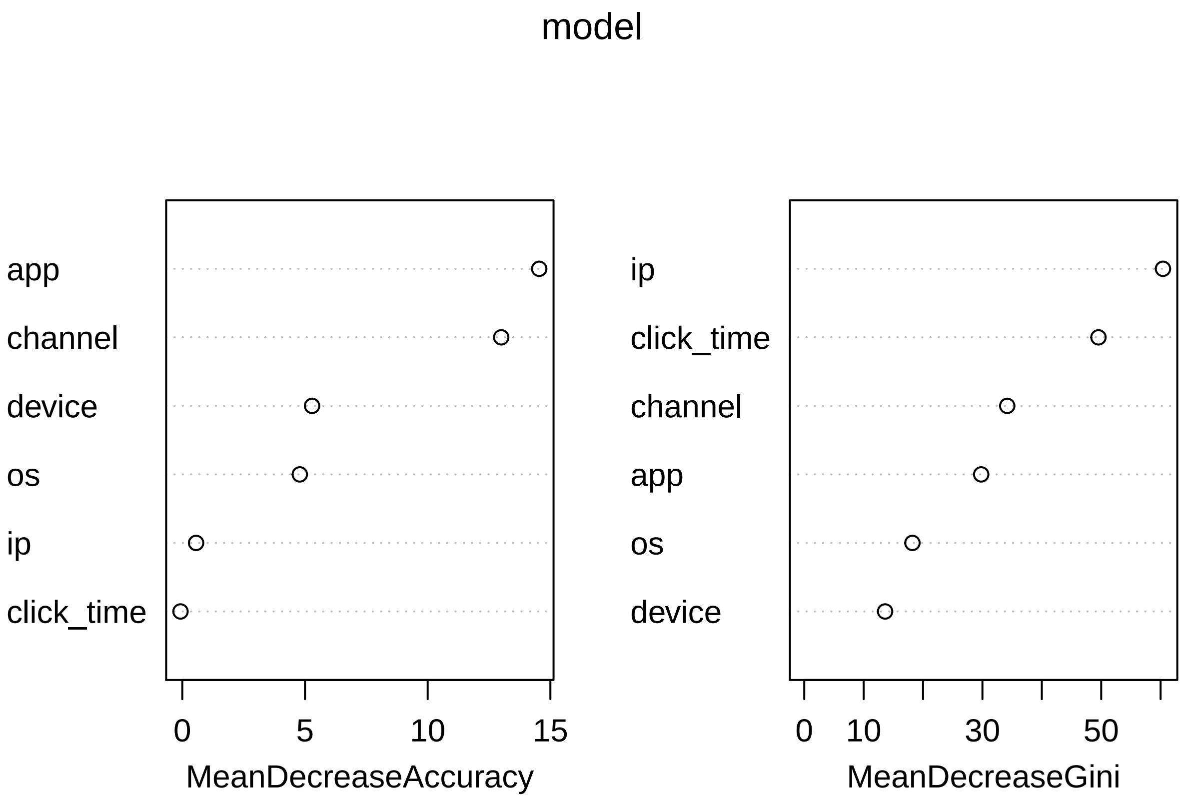
```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
```

```
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':
##
##     combine

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

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
# Converting target variable to factor
train$is_attributed <- as.factor(train$is_attributed)

# Converting dependents variables to integer to run random forest algorithm
cnames <- c('ip', 'app', 'device', 'os', 'channel')
for (var in cnames) {
  train[,var] <- as.integer(train[,var])
  test[,var] <- as.integer(test[,var])
}

# Due to my machine's processing and memory limitations, I will work with random samples
# from the training and test datasets to build and evaluate the model
index <- sample(1:nrow(train), 100000)
train_data <- train[index,]

# Creating random forest model measuring importance
model <- randomForest( is_attributed ~ ip + app + device + os + channel + click_time,
                       data = train_data,
                       ntree = 100,
                       nodesize = 10,
                       importance = TRUE)

# Plotting estimate variable importance
varImpPlot(model)
```



The dataset has few variables so I will make the first version of the model using all but (attribute_time).

Model construction and evaluation

In this script, the prediction model is created. Then, the prediction for the test data set is made and the model is evaluated. Due to my machine's processing and memory limitations, I will work with random samples from the training and test datasets on proportion 70% training to 30% test to build and evaluate the model. I used random forest algorithm again to build the model. Then I assessed its efficiency with a matrix of confusion, and accuracy. "sample_submission.csv" file contains correct classification for test dataset.

Getting samples of train and test datasets and and treating it.

```
# Getting the correct classification for test dataset
test_result <- read.csv("sample_submission.csv")

# Getting smaller samples of trein and test datasets
# on proportion 70% training to 30% test
index <- sample(1:nrow(train), 100000)
train_data <- train[index, ]

str(train_data)
```

```
## 'data.frame':    100000 obs. of  10 variables:
##  $ X              : int  950526 4629928 1497650 4446263 4027508 9918896 7434971 5723399 492647 46509
##  $ ip             : int  1920 24864 3438 16667 9159 8045 43030 55274 15605 15727 ...
##  $ app            : int  13 14 9 3 4 7 19 13 13 3 ...
##  $ device         : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ os             : int  19 7 14 7 7 18 20 14 14 21 ...
##  $ channel        : int  48 157 41 144 138 30 16 76 72 136 ...
```

```
##  $ click_time     : POSIXct, format: "2017-11-06 16:20:44" "2017-11-06 19:35:50" ...
##  $ attributed_time: POSIXct, format: NA NA ...
##  $ is_attributed  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ timeDiff       : 'hms' num  NA NA NA NA ...
##   ..- attr(*, "units")= chr "secs"
```

```
index <- sample(1:nrow(test), 42857)
test_data <- test[index,]
test_data <- merge(test_data, test_result[index,], by.x="click_id", by.y="click_id")
test_data$click_id <- NULL

# Converting dependent variables to integer before creating model
# using random forest algorithm
cnames <- c('ip', 'app', 'device', 'os', 'channel')
for (var in cnames) {
  train_data[,var] <- as.integer(train_data[,var])
  test_data[,var] <- as.integer(test_data[,var])
}

# Emphasize the levels of the target variable
# since I used a random sample to build the model
levels(train_data$is_attributed) <- c("0", "1")
```

Creating and printing model:

```
# Creating model
model <- randomForest( is_attributed ~ ip + app + device + os + channel + click_time,
                       data = train_data,
                       ntree = 100,
                       nodesize = 10)

# Print model
print(model)
```

```
##
## Call:
##  randomForest(formula = is_attributed ~ ip + app + device + os +      channel + click_time, data = t
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.16%
## Confusion matrix:
##       0  1   class.error
## 0 99803  9 0.00009016952
## 1   154 34 0.81914893617
```

Predicting classification using sample test dataset and building a data.frame containing columns: expected results and correct results.

```
# Generating predictions in test data
pred <- data.frame(observed = test_data$is_attributed,
                        predicted = predict(model, newdata = test_data[,1:6]))

pred$observed <- as.factor(pred$observed)
levels(pred$observed) = c("0", "1")
```

```r
levels(pred$predicted) = c("0", "1")

# Visualizing the results
View(pred)
```

Evaluating model

```r
# Evaluating the model
# Generating a confusion matrix
caret::confusionMatrix(pred$observed, pred$predicted)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 42835    22
##          1     0     0
##
##                Accuracy : 0.9995
##                  95% CI : (0.9992, 0.9997)
##     No Information Rate : 0.9995
##     P-Value [Acc > NIR] : 0.5564
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.000007562
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.9995
##          Neg Pred Value :    NaN
##              Prevalence : 0.9995
##          Detection Rate : 0.9995
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

The model created has the model created has an accuracy of 0.9996. Besides, it's a efficient model. However as the tests were done with a small sample I cannot say that it is an efficient model. The sample may not contemplate positive oficial results (downloaded apps).