

# Projeto1 DataScienceAcademy

Igor Guerato Garbo

11/07/2021

## Projeto 1 Data Science Academy

Analise Preditiva de downloads por click de anuncio  
este projeto sera documentado conforme etapas abaixo  
##Etapa 1 - Carregando datasets Necessarios

```
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr    0.3.4
## v tibble   3.1.0     v dplyr    1.0.4
## v tidyr    1.1.3     v stringr  1.4.0
## v readr    1.4.0     vforcats  0.5.1

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

library(data.table)

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

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

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

library(lubridate)

## Attaching package: 'lubridate'
```

```

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

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

```

## Etapa 2 - Carregando Dataset Completo

```
inicial_Data<- fread("train.csv")
```

##Etapa 4 - Realizando Amostragem do Dataset

A amostragem foi realizada de forma a aproveitar todos os dados de download possíveis e equilibrar estes dados de forma a evitar resultados inviesados

```

numero.downloads <- nrow(inicial_Data[is_attributed == 1,])
indices.download <- inicial_Data[, .I[is_attributed == 1]]
indices.negativo <- sample(inicial_Data[is_attributed == 0,.I], size = numero.downloads)
WorkData<- inicial_Data[c(indices.download,indices.negativo)]
rm(inicial_Data)

```

##Etapa 5 - Semparando dados de treino e teste

```

Train_rows <- sample(1:nrow(WorkData),nrow(WorkData)*0.7)
Train_Data <- WorkData[Train_rows,]
Test_Data <- WorkData[-Train_rows,]

```

## Etapa 6 - Analise Exploratoria e transformacoes iniciais

```
str(Train_Data)
```

```

## Classes 'data.table' and 'data.frame':  639584 obs. of  8 variables:
## $ ip           : int  1675 27729 199680 93471 72986 52019 188348 102931 98432 112271 ...
## $ app          : int  35 2 12 12 15 2 3 18 19 15 ...
## $ device       : int  1 1 1 1 1 1 1 1 0 1 ...
## $ os           : int  19 15 19 19 18 16 7 17 50 13 ...
## $ channel      : int  21 18 259 178 379 205 137 121 347 245 ...
## $ click_time   : POSIXct, format: "2017-11-07 13:54:28" "2017-11-07 09:17:05" ...
## $ attributed_time: POSIXct, format: "2017-11-07 14:48:16" NA ...
## $ is_attributed : int  1 0 0 0 0 0 0 1 0 ...
## - attr(*, ".internal.selfref")=<externalptr>

max(WorkData$click_time)

## [1] "2017-11-09 15:59:33 UTC"

```

```

min(WorkData$click_time)

## [1] "2017-11-06 15:41:07 UTC"

lapply(Train_Data,function(dt){table(is.na(dt))})

## $ip
##
## FALSE
## 639584
##
## $app
##
## FALSE
## 639584
##
## $device
##
## FALSE
## 639584
##
## $os
##
## FALSE
## 639584
##
## $channel
##
## FALSE
## 639584
##
## $click_time
##
## FALSE
## 639584
##
## $attributed_time
##
## FALSE TRUE
## 320421 319163
##
## $is_attributed
##
## FALSE
## 639584

```

Verifica-se que composicao de datas nao eh suficiente para realizar uma variacao mensal ou semanal do dataset para tanto nao sera necessario criar novos dados para considerar mudancas de epoca no dataset, tambem pode-se verificar que apenas uma das variaveis possui dados NA e eh a variavel do horario do download uma variavel que podemos ignorar para o treinamento

```

Changing_types <- function(dt){
fatores <- c("app","device","os","channel","is_attributed")
Char <- c("ip")
dt[,fatores] <- dt[,lapply(.SD,as.factor), .SDcols = fatores]
dt[,Char] <- dt[,lapply(.SD,as.character), .SDcols = Char]
return(dt)
}

Train_Data<-Changing_types(Train_Data)
Test_Data<-Changing_types(Test_Data)

Adding_dates_columns <- function(df){df %>%
  mutate(Hour = hour(click_time),
        day = day(click_time))
}

Train_Data <-Adding_dates_columns(Train_Data)
Test_Data <-Adding_dates_columns(Test_Data)

```

##Etapa 7 - Analise exploratoria das variaveis resultantes

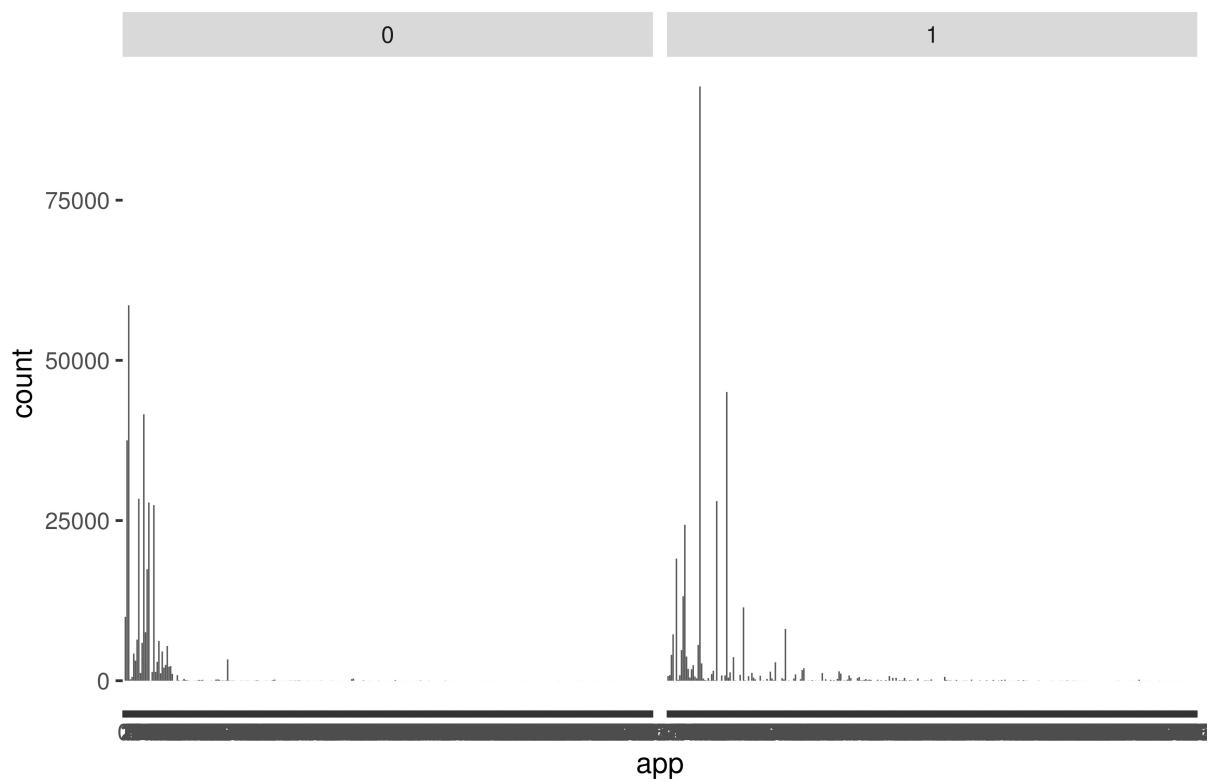
```

lapply(colnames(Train_Data),function(x){
  if(unlist(Train_Data[,lapply(.SD,is.factor),.SDcols = c(x)])){
    ggplot(Train_Data, aes_string(x)) +
      geom_bar()+
      facet_grid(. ~ is_attributed)+
      ggtitle(paste("Relacao de Downloads por ",x))
  }
})

## [[1]]
## NULL
##
## [[2]]

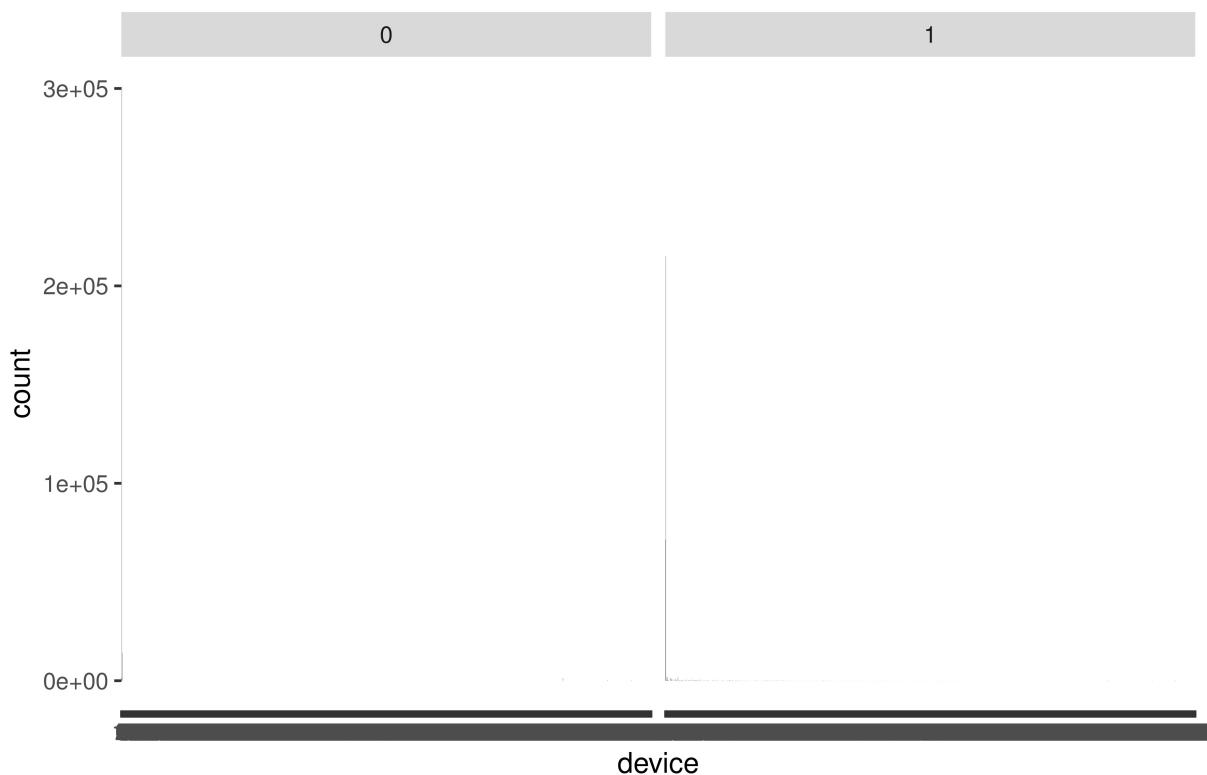
```

## Relacao de Downloads por app



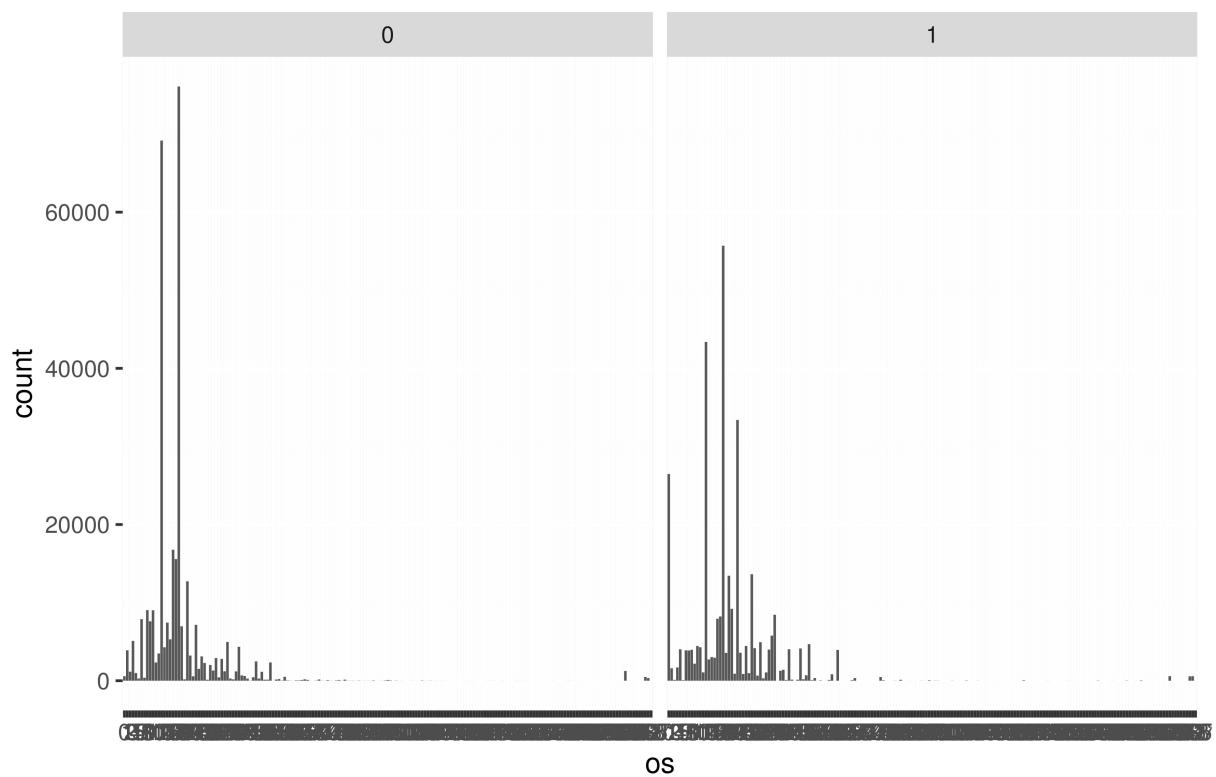
```
##  
## [[3]]
```

Relacao de Downloads por device



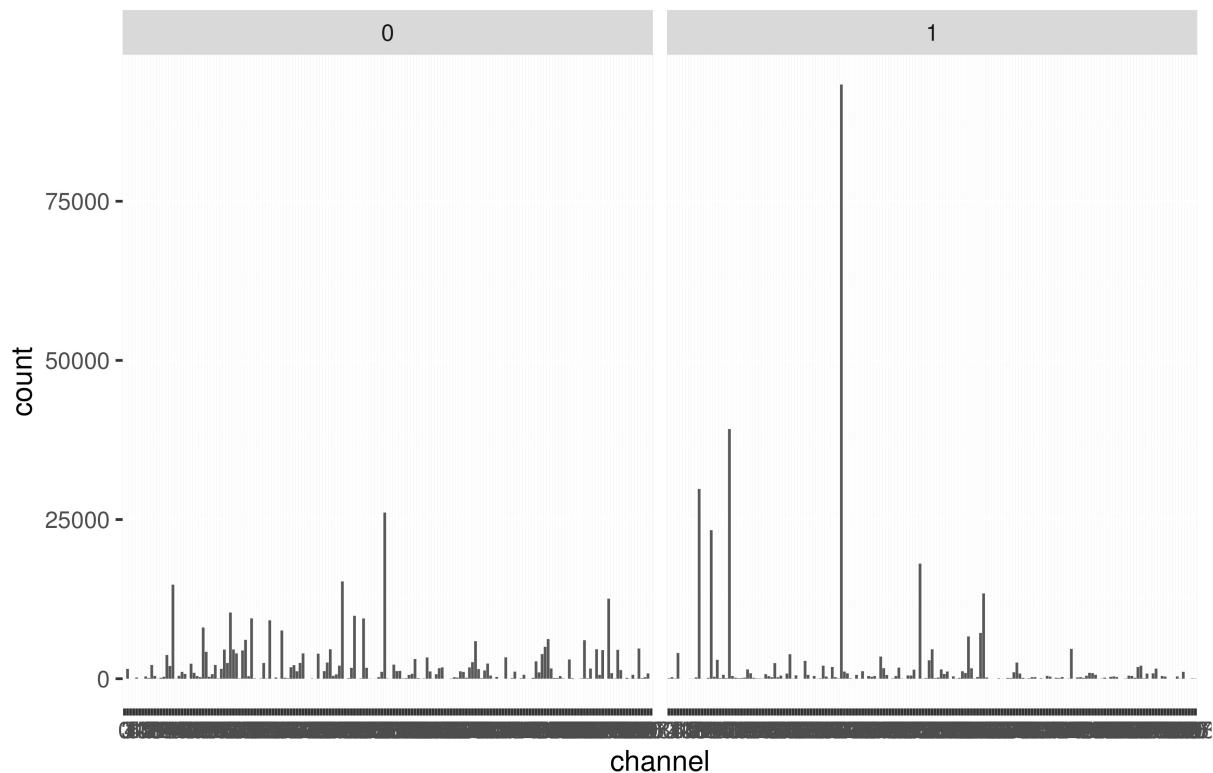
```
##  
## [4]
```

## Relacao de Downloads por os



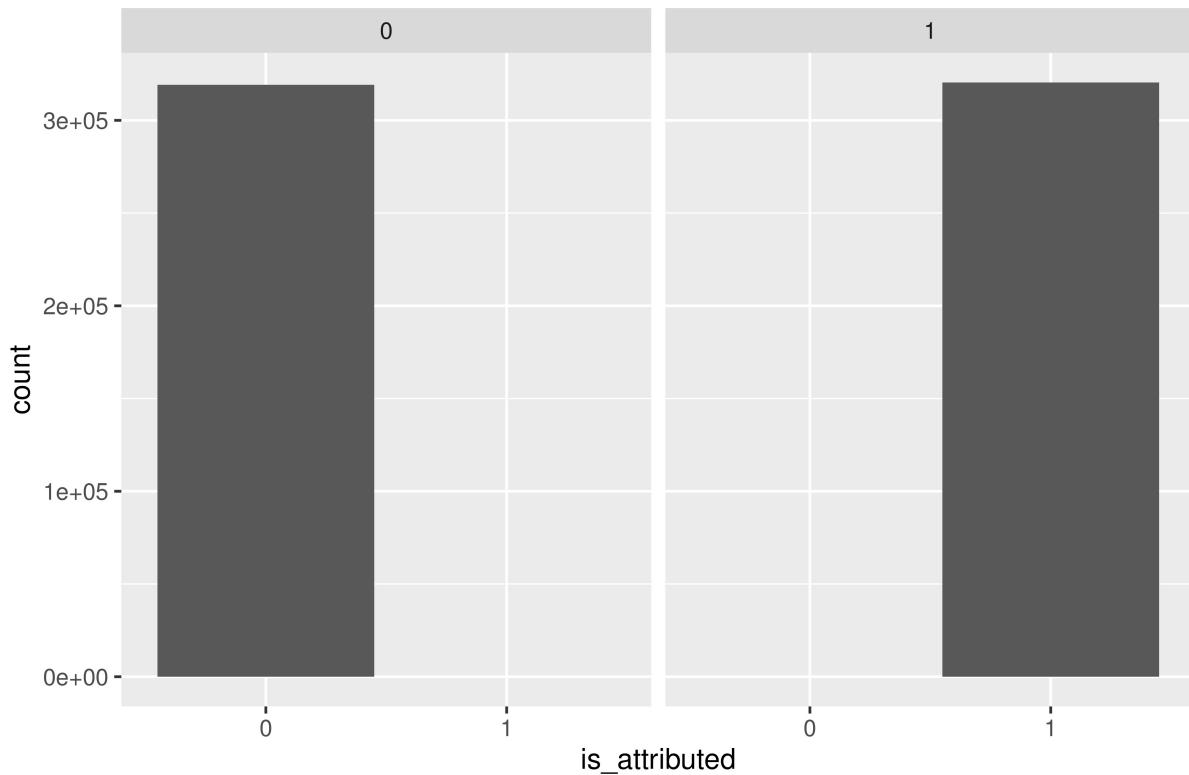
```
##  
## [[5]]
```

Relacao de Downloads por channel



```
##  
## [[6]]  
## NULL  
##  
## [[7]]  
## NULL  
##  
## [[8]]
```

## Relacao de Downloads por is\_attributed



```
##  
## [[9]]  
## NULL  
##  
## [[10]]  
## NULL
```

Vemos que pela grande quantidade de fatores fica dificil retirar informacoes que facilitem a analise mas consegue-se perceber a consistencia de que aparelhos, os e apps que possuem mais cliques tambem possuem maior numero de downloads.

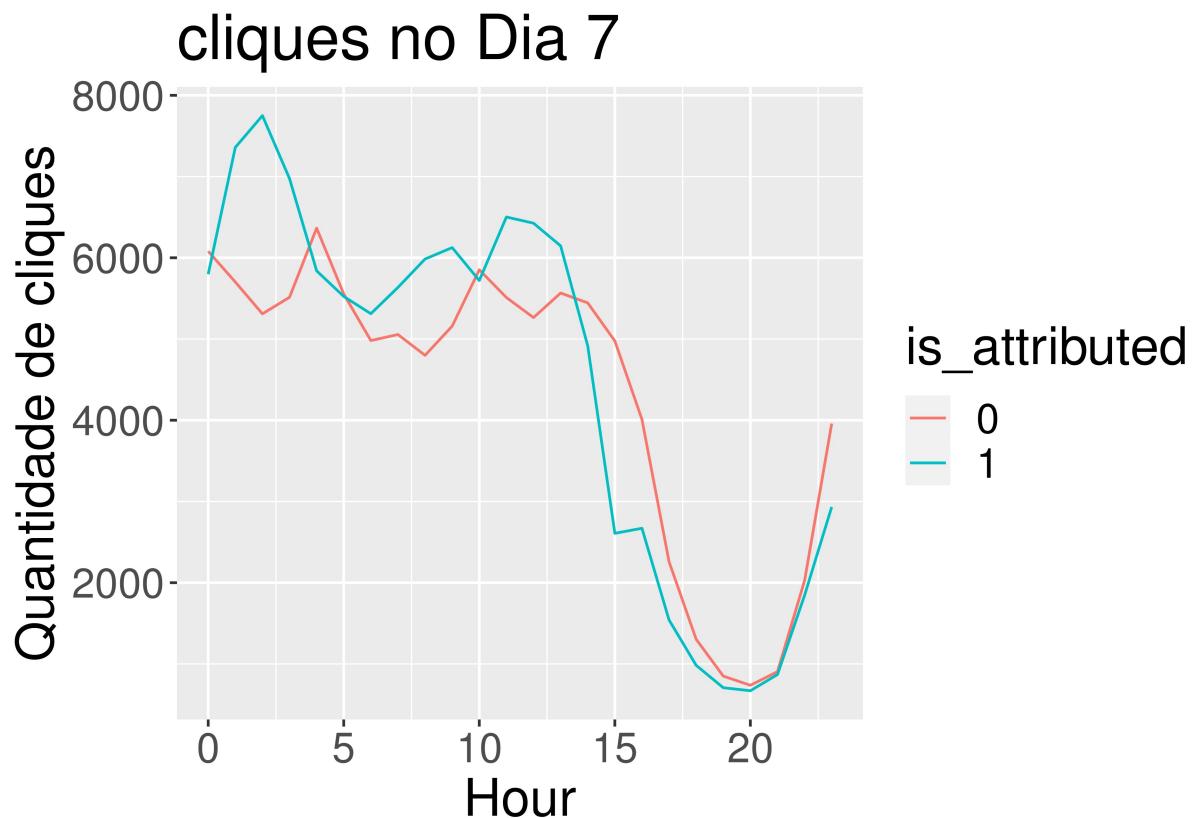
```
Resumo <- Train_Data %>%
  count(Hour, day, is_attributed)

Ddays <- unique(Resumo$day)

tms.plot <- function(Ddays){
  ggplot(Resumo[Resumo$day == Ddays,], aes(x = Hour, y = n, group = is_attributed, color = is_attributed))
  + geom_line() +
  ylab("Quantidade de cliques")+
  labs(title = paste("cliques no Dia ", as.character(Ddays), sep = ""))+
  theme(text = element_text(size = 20))
}

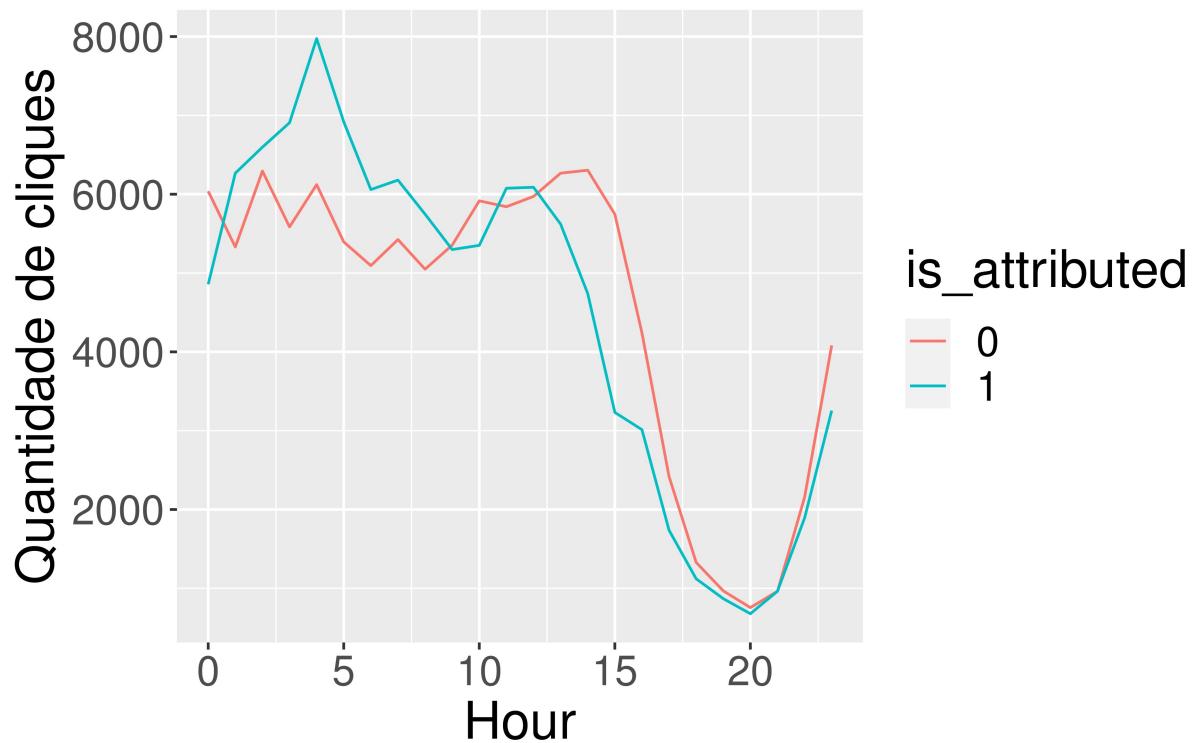
lapply(Ddays, tms.plot)
```

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



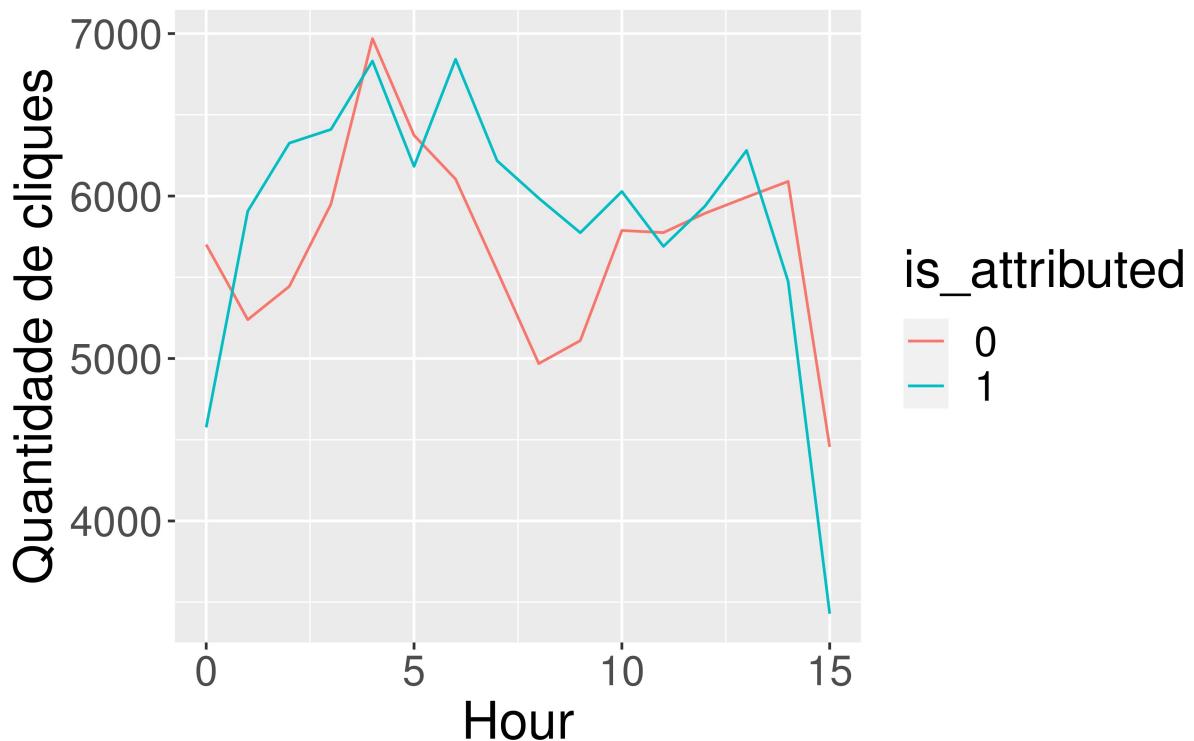
```
##  
## [[2]]
```

## cliques no Dia 8



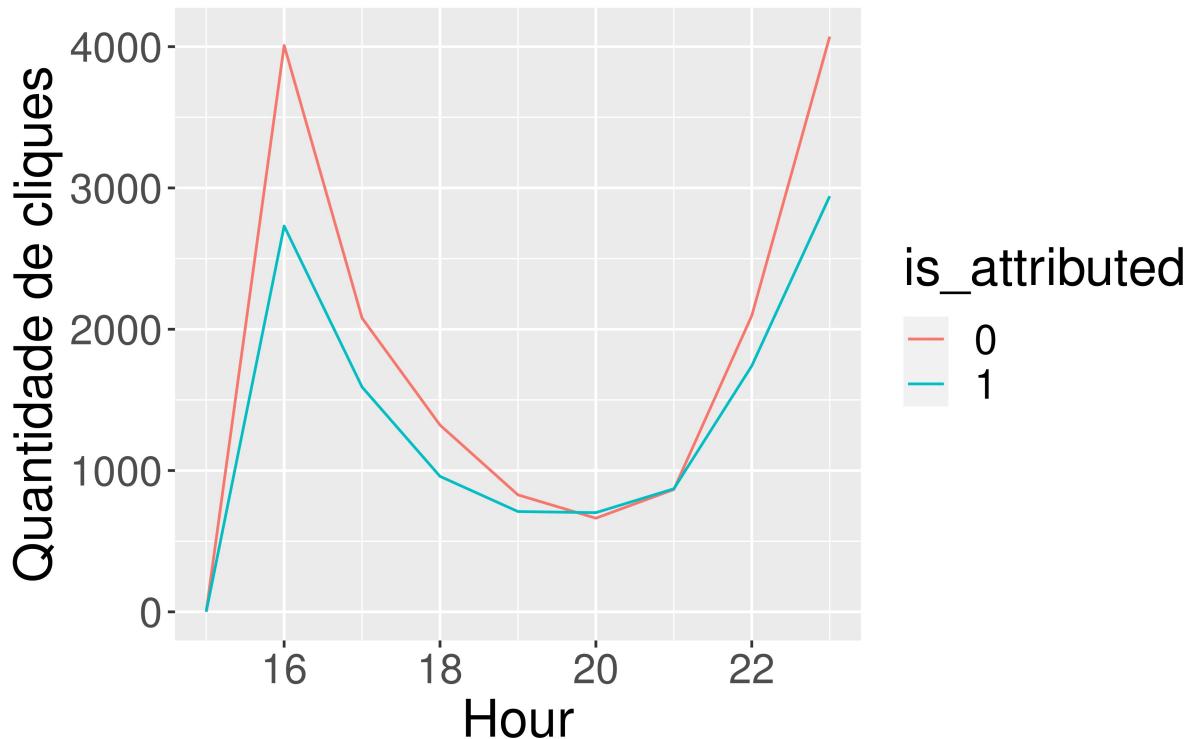
```
##  
## [[3]]
```

## cliques no Dia 9



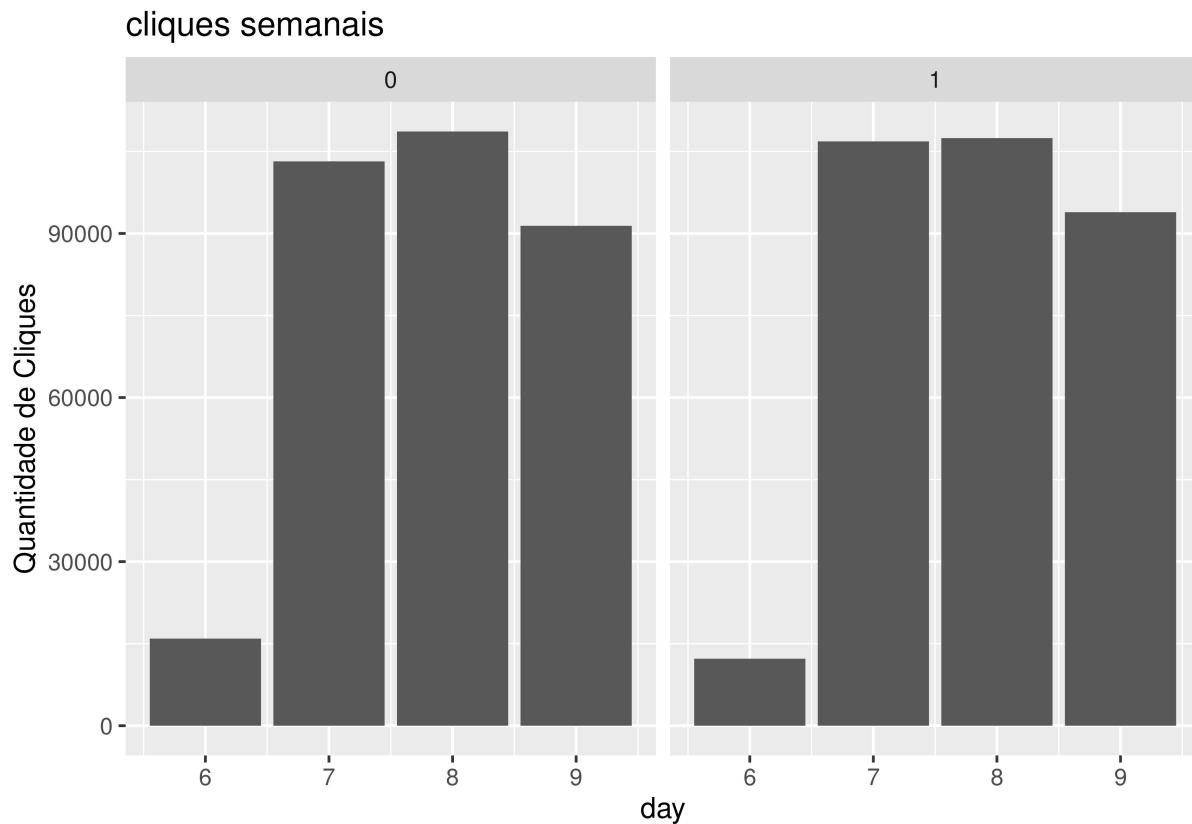
```
##  
## [[4]]
```

## cliques no Dia 6



Vemos que em alguns dias e horarios possuimos maior quantidade de downloads por cliques o que representa uma importancia do horario para realizar a analise preditiva, sendo que existe uma boa correlacao entre quantidade de cliques e downloads

```
ggplot(Train_Data, aes(x = day))+  
  geom_bar() +  
  ylab("Quantidade de Cliques") +  
  facet_grid(. ~ is_attributed) +  
  labs(title = paste("cliques semanais"))
```



verificase que existe pouca relacao entre o dia e quantidade de downloads de forma isolada, exeto que temos dias com maior quantidade de cliques e portanto maior quantidade de downloads.

##Etapa 8 - Treinamento do modelo

inicialmente foram considerados tres tipos de modelo para realizar treinamento foram o naive bayes, knn e SVM

```

library(e1071)
library(kernlab)

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':
##   cross

## The following object is masked from 'package:ggplot2':
##   alpha

library(kknn)

#treinamento do modelo naive bayes
modelo1_naive <- naiveBayes(is_attributed ~ . -attributed_time, data = Train_Data)

```

Inicialmente foi realizado apenas o treinamento do modelo naive bayes, os demais modelo ficaram com vetores extremamente grandes com tamanhos maiores que 200 G o que impediua utilizacao

##Etapa 9 - Teste do Modelo

```
predicao1 <- predict(modelo1_naive, Train_Data[,c(1:6,9:10)])  
  
Tabela1 <- table(predicao1,Train_Data$is_attributed)  
  
Tabela1[1,2]  
  
## [1] 84944  
  
#teste inicial para verificar se treinamento teve resultados satisfatorios  
Calc_Perf_measures <-function(table1,measure){  
  tp <- table1[1,1]  
  tf <- table1[2,2]  
  fp <- table1[2,1]  
  fn <- table1[1,2]  
  if (measure == "accuracy") {  
    accuracy <- (tp+tf)/(tp+tf+fp+fn)  
    return(accuracy)  
  } else {  
    if (measure == "recall") {  
      recall <- tp/(tp+fn)  
      return(recall)  
    } else {  
      if (measure == "precision") {  
        precision <- tp/(tp+fp)  
        return(precision)  
      } else {  
        if (measure == "fscore") {  
          fscore <- 2*tp/(2*tp+fp+fn)  
          return(fscore)  
        } else {  
          if (measure == "all") {  
            fscore <- 2*tp/(2*tp+fp+fn)  
            precision <- tp/(tp+fp)  
            recall <- tp/(tp+fn)  
            accuracy <- (tp+tf)/(tp+tf+fp+fn)  
            calc_perf <- c(accuracy, recall, precision, fscore)  
            names(calc_perf)<- c("accuracy", "recall", "precision", "fscore")  
            return(calc_perf)  
          }  
        }  
      }  
    }  
  }  
}  
  
Calc_Perf_measures(Tabela1,"all")
```

```

## accuracy    recall precision    fscore
## 0.8046777 0.7667181 0.8747317 0.8171711

#teste para verificar como algoritmo se sai com novos datasets
predicao_test <- predict(modelo1_naive, Test_Data[,c(1:6,9:10)])

tabela_test1 <- table(predicao_test,Test_Data$is_attributed)

Calc_Perf_measures(tabela_test1,"all")

```

```

## accuracy    recall precision    fscore
## 0.8671728 0.8426341 0.9018647 0.8712439

```

Vemos que o modelo naive bayes teve boa accuracia e bom f-score o que permite que o mesmo possa ser usado em producao mas que ainda pode ser melhorado.