DS5110-project

Zishen Li

2018/10/28

1.Introduction

fullvisitorId should be string to be unique in the data set

2.Enviorment

```
library(ggplot2)
library(tidyverse)
```

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

```
## <U+221A> tibble  1.4.2     <U+221A> purrr   0.2.4
## <U+221A> tidyr   0.8.1     <U+221A> dplyr   0.7.6
## <U+221A> readr   1.1.1     <U+221A> stringr 1.2.0
## <U+221A> tibble  1.4.2     <U+221A> forcats 0.2.0
```

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

```
library(dplyr)
library(stringr)
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
```

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

```
library(tidyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

3.Tidy data and data exploration

3.1 Import and tidy the data

Here we only import the train set since the test set of the competition does not provide the total revenue, which is our prediction target. Therefore, we choose to work on the training set and partition it into train, validation and test set.

```
train <- read.csv("/Users/mbp/Desktop/NEU/intro data managment/project/data/train.csv", na = NA)

dim(train)
```

```
## [1] 903653     12
```

```
sapply(train, class)
```

```
##      channelGrouping                 date               device
##             "factor"            "integer"             "factor"
##        fullVisitorId           geoNetwork            sessionId
##            "numeric"             "factor"             "factor"
## socialEngagementType               totals        trafficSource
##             "factor"             "factor"             "factor"
##              visitId          visitNumber       visitStartTime
##            "integer"            "integer"            "integer"
```

```
train[1,]
```

```
##   channelGrouping     date
## 1  Organic Search 20160902
##
## 1 {"browser": "Chrome", "browserVersion": "not available in demo dataset", "browserSize": "not availa
##   fullVisitorId
## 1   1.13166e+18
##
## 1 {"continent": "Asia", "subContinent": "Western Asia", "country": "Turkey", "region": "Izmir", "meti
##                         sessionId socialEngagementType
## 1 1131660440785968503_1472830385 Not Socially Engaged
##                                                                                 totals
## 1 {"visits": "1", "hits": "1", "pageviews": "1", "bounces": "1", "newVisits": "1"}
##
## 1 {"campaign": "(not set)", "source": "google", "medium": "organic", "keyword": "(not provided)", "ac
##      visitId visitNumber visitStartTime
## 1 1472830385           1     1472830385
```

The data set has 12 variables and 903,653 observations. There are 4 columns with JSON format. Moreover, as mentioned in the introduction, we change the data type of fullvisitorId to character as well.

```
library(jsonlite)

# Build a function to tidy the data set

tidy <- function(data,data_o){
  for (i in 1:4){
  first <- str_c(data[,i], collapse = ",")
  second <- str_c("[", first, "]")
  data_o <- cbind(data_o,fromJSON(second, flatten = TRUE))
  }
```

```
  data_o$fullVisitorId <- as.character(data_o$fullVisitorId)
  return(select(data_o, -device, -geoNetwork, -trafficSource, -totals))
}

train_tidy <- tidy(train[,c(3,5,8,9)], train)
dim(train_tidy)
```

```
## [1] 903653     55
```

```
sum(sapply(train_tidy, class) == "logical")
```

```
## [1] 3
```

```
sum(sapply(train_tidy, class) == "character")
```

```
## [1] 45
```

```
sum(sapply(train_tidy, class) == "factor")
```

```
## [1] 3
```

```
sum(sapply(train_tidy, class) == "integer")
```

```
## [1] 4
```

After parse the JSON format and tidy the data, we get a data set with 55 variales. 48 of them are character variables, 4 are numerical variables and 3 are boolean variables.

3.2 Data exploration

3.2.1 General introduction

```
# Distinct values of each variable
distinct_value <- sapply(train_tidy,n_distinct)
distinct_value <- as.data.frame(distinct_value)
names(distinct_value) <-"num_distinct_values"
distinct_value <- distinct_value %>%
  rownames_to_column("colnames") %>%
  mutate(colnames = reorder(colnames, -num_distinct_values))

constant <- filter(distinct_value,num_distinct_values == 1)
constant
```

```
##                              colnames num_distinct_values
## 1                socialEngagementType                   1
## 2                      browserVersion                   1
## 3                         browserSize                   1
## 4               operatingSystemVersion                   1
## 5                 mobileDeviceBranding                   1
## 6                    mobileDeviceModel                   1
## 7                  mobileInputSelector                   1
## 8                     mobileDeviceInfo                   1
## 9           mobileDeviceMarketingName                   1
## 10                        flashVersion                   1
## 11                            language                   1
## 12                        screenColors                   1
```

```
## 13                    screenResolution                   1
## 14                              cityId                   1
## 15                            latitude                   1
## 16                           longitude                   1
## 17                      networkLocation                   1
## 18                              visits                   1
## 19 adwordsClickInfo.criteriaParameters                   1
```

There are 19 variables with constant value, which have no contribution to model we will build later. We remove those columns.

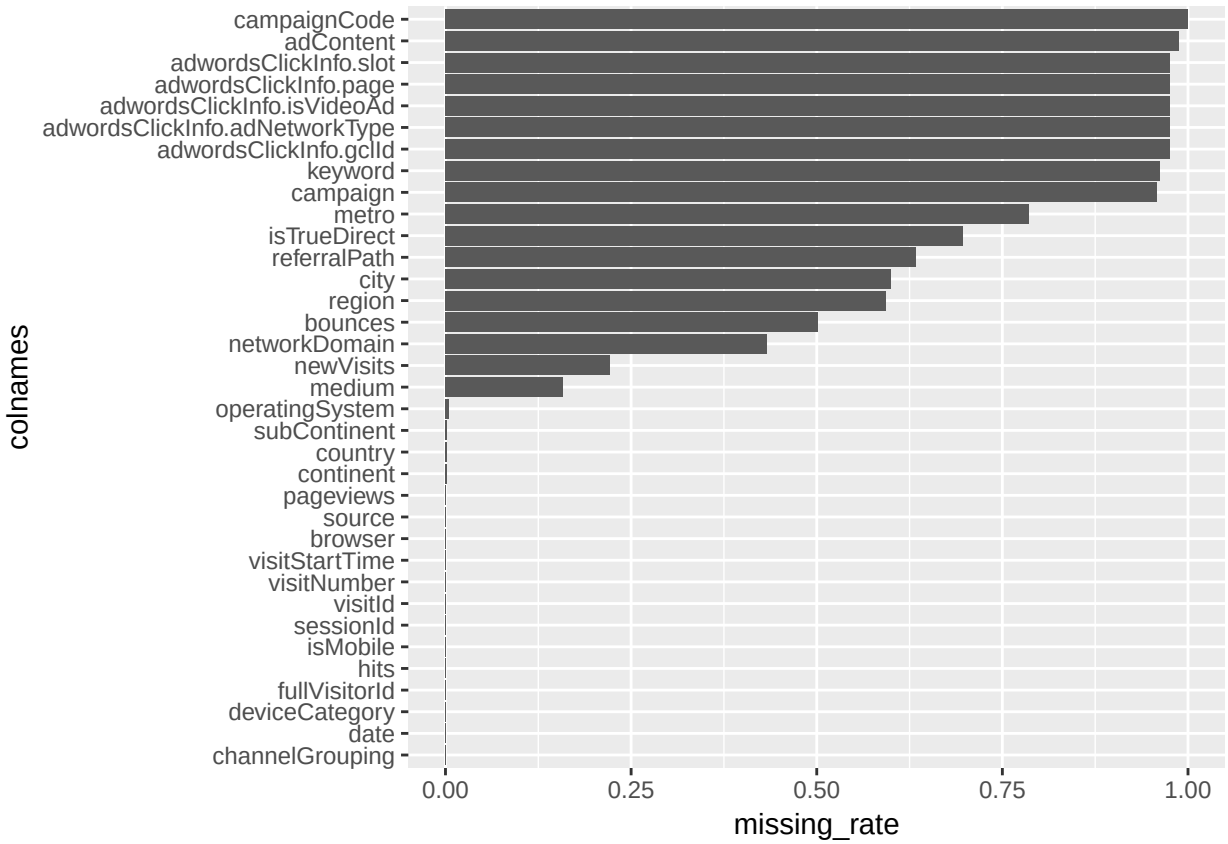
```
del <- constant$colnames
del <- as.character(del)
train_tidy_s <- select(train_tidy,-del)
```

3.2.2 Missing value

Explore the distribution of the missing value of each variable in the data set.

```
# Set the explanatory variable to x, response variable to y
x <- select(train_tidy_s, -transactionRevenue)
y <- train_tidy_s$transactionRevenue %>%
  as.numeric()
# Format transformation
x$channelGrouping <- as.character(x$channelGrouping)
x$sessionId <- as.character(x$sessionId)
# Change all kinds of missing values into NA.
x <- mutate_all(x,function(a) ifelse(a %in%
                                       c("not available in demo dataset",
                                         "(not provided)",
                                         "(not set)",
                                         "<NA>",
                                         "unknown.unknown",
                                         "(none)"), NA, a))

missing_rate <- sapply(x, function(a) mean(is.na(a)))
missing_rate%>%
  as.data.frame()%>%
  rename(missing_rate = '.')%>%
  rownames_to_column("colnames")%>%
  mutate(colnames = reorder(colnames,missing_rate))%>%
  ggplot()+
  geom_col(aes(colnames, missing_rate))+
  coord_flip()
```



We find there are 15 variables has more than 50% missing value rate. Further more we find the column "campaignCode" only have one none-NA value, which is useless in the later modeling. Thus we remove this column.

```
sum(missing_rate>0.5)
```

```
## [1] 15
```

```
missing_rate["campaignCode"]
```

```
## campaignCode
##    0.9999989
```

```
sum(!is.na(x$campaignCode))
```

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

```
unique(x$campaignCode)
```

```
## [1] NA              "11251kjhkvahf"
```

```
x <- select(x,-campaignCode)
```

For the response variable, it has relatively high missing value rate, 98%.

```
summary(y)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
## 1.000e+04 2.493e+07 4.945e+07 1.337e+08 1.077e+08 2.313e+10    892138
```

```
mean(is.na(y))
```

```
## [1] 0.9872573
```

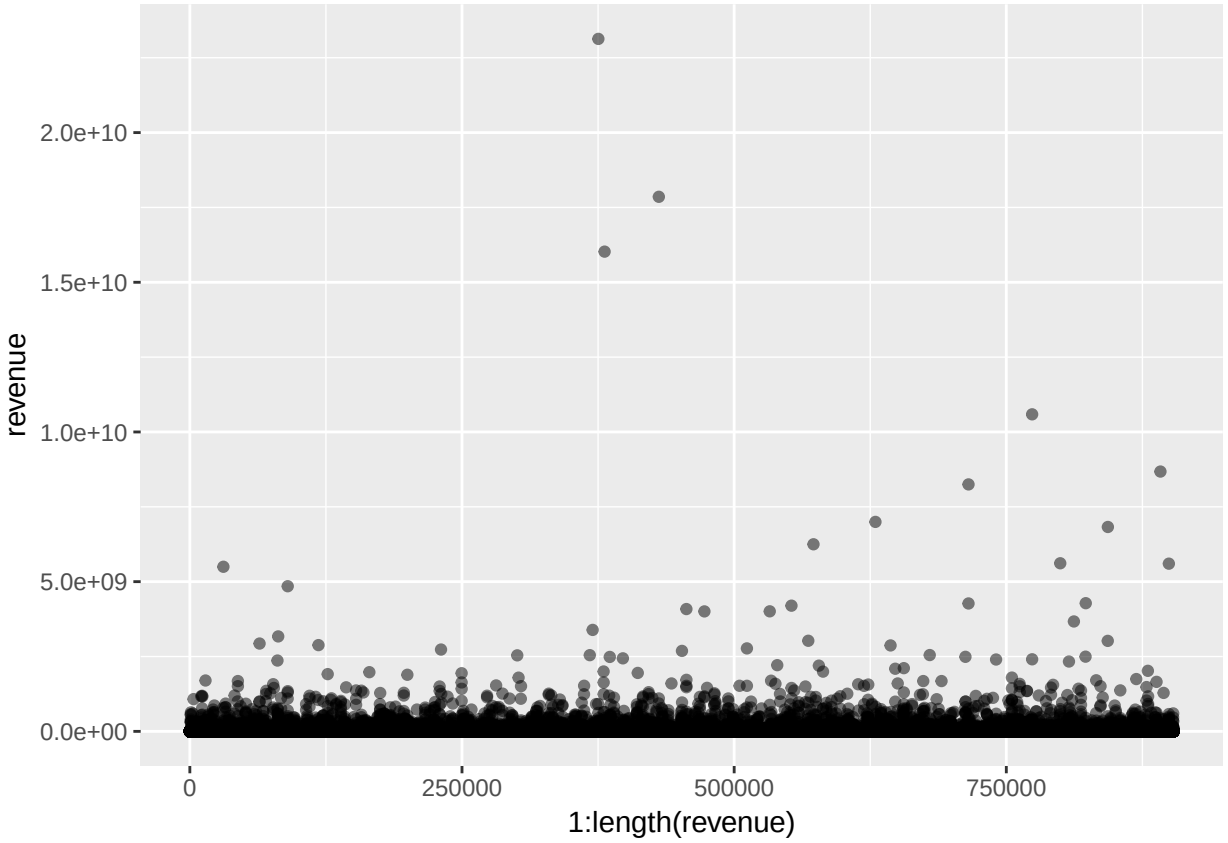
Intuitively, such high missing value rate may due to high proportion of customers who usually need multiple visits to complete the online purchase. Based on that assumption, we could repalce thoes NAs with 0.

```
y[is.na(y)] <- 0
summary(y)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.000e+00 0.000e+00 0.000e+00 1.704e+06 0.000e+00 2.313e+10
```

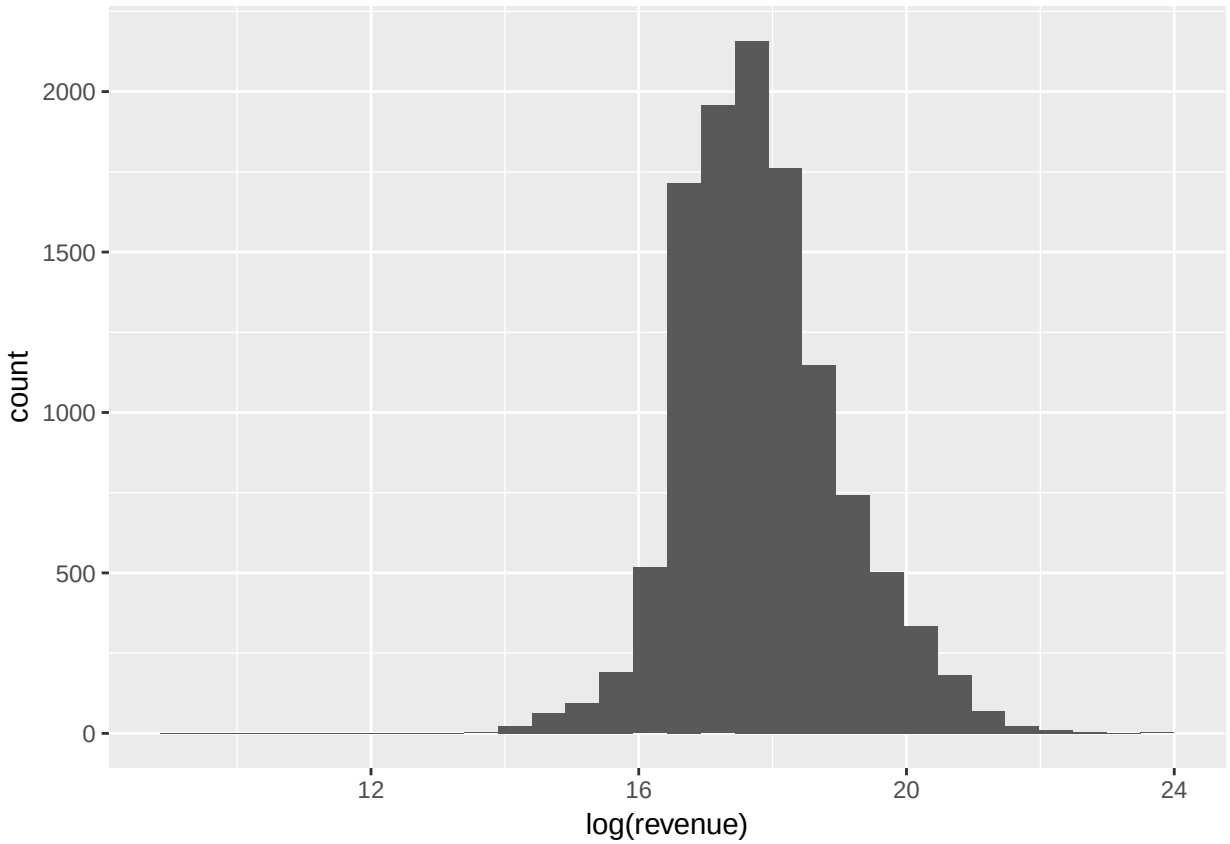
3.2.3 distribution of response variable

```
# The distribution of revenue
y %>% as.data.frame()%>%
  rename(revenue = '.')%>%
  ggplot(aes(1:length(revenue), revenue))+
    geom_point(alpha = 0.5)
```




```
# The distribution of log revenue
y %>% as.data.frame()%>%
  rename(revenue = '.')%>%
  filter(revenue>0)%>%
  ggplot()+
    geom_histogram(aes(log(revenue)))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



As mentioned above, a large part of the revenue is 0. Since our target is to predict log of revenue, we also plot the distribution of log(revenue) which is a little bit right skew.

3.2.4 distribution of explanatory variables(ZHANG DUO)

3.2.5 Correlations among explanatory variables(ZHANG DUO)

3.2.6 Correlations between explanatory variables and response variables(DORIS)

3.2.X Preparation for modeling

```
# Transformation of some columns
x <- x%>%
  mutate(date = ymd(date), visitId = as.character(visitId),
         hits = as.integer(hits),pageviews = as.integer(pageviews),
         bounces = as.integer(bounces),
         newVisits = as.integer(newVisits))
```