Google_Analytics_Customer_Revenue_Prediction_20200518

June 22, 2020

1 GOOGLE ANALYTICS CUSTOMER REVENUE PREDICTION

1.1 Table of Contents

## 1. Introduction

Google Analytics Customer Revenue Prediction is one of the competition available in Kaggle. This project is aimed to analyze Google Merchandise Store customer dataset to predict the revenue gained from customer in the foreseable future. The dataset was downloaded from Kaggle which hosted a competition on November 2018. In the era of big data, extracting meaningful information from a dataset is essential to gain business insight and understanding the needs of each customers. Therefore, the available dataset can provide some information regarding factors that can contribute to the spending behaviour of a customer. The inference from the extracted information could possibly change and improve bussiness decision in strategize marketing budget and action plan to drive more revenues.

## 2. Data Wrangling

The data for this project can be obtained at Kaggle competition web page. The provided datasets in this competition are train_v2.csv and test_v2.csv, which are the training and testing data respectively. In the training set, it consists of the data from 1st August 2016 to 30th April 2018. On the other hand, the testing set covers the data range from 1st May 2018 to 15th October 2018. The requirement of the competition is to predict the expected log revenue of all of the customer in the training set during the period of 1st December 2018 to 31 January 2019.

There is a gap of 46 days between the test set data and the prediction period. This indicated that we have to train a model which can predict revenue that is possibly generated by a customer after 46 days.

Due to the long period in the training set, the data size contains 1.7 million records with 23.7 GB of file size. Therefore, before loading the whole dataset into the memory, it is wise to examine the dataset structure and conduct data preprocessing to prepare for exploratory data analysis.

[2]:

```
# Import library
import pandas as pd
import numpy as np
import json
import ast
import glob
import os
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime, time, date, timedelta
import pytz
from sklearn.preprocessing import MinMaxScaler
import geopandas as gpd #conda install -c conda-forge geopandas
import pycountry
from plotly.offline import init_notebook_mode, iplot, plot
import plotly.graph_objs as go
init_notebook_mode(connected=True) # For plotly
%matplotlib inline
from IPython.display import Image
```

1.1.1 2.1 Dataset Exploration

Since the dataset is huge (23.7 GB), it will take a long time to load the full dataset. Therefore, for the purpose of exploring the dataset, only part of the data is loaded to examine the features that contained in the dataset.

[2]:

```
df_partial = pd.read_csv("train_v2.csv", nrows = 10)
df_partial.head()
```

[2]:

```
  channelGrouping                               customDimensions      date  \
0  Organic Search             [{'index': '4', 'value': 'EMEA'}]  20171016
1        Referral    [{'index': '4', 'value': 'North America'}]  20171016
2          Direct    [{'index': '4', 'value': 'North America'}]  20171016
3  Organic Search             [{'index': '4', 'value': 'EMEA'}]  20171016
4  Organic Search  [{'index': '4', 'value': 'Central America'}]  20171016

                                              device        fullVisitorId  \
0  {"browser": "Firefox", "browserVersion": "not …  3162355547410993243
1  {"browser": "Chrome", "browserVersion": "not a…  8934116514970143966
2  {"browser": "Chrome", "browserVersion": "not a…  7992466427990357681
3  {"browser": "Chrome", "browserVersion": "not a…  9075655783635761930
4  {"browser": "Chrome", "browserVersion": "not a…  6960673291025684308

                                          geoNetwork  \
0  {"continent": "Europe", "subContinent": "Weste…
1  {"continent": "Americas", "subContinent": "Nor…
```

```
2  {"continent": "Americas", "subContinent": "Nor…
3  {"continent": "Asia", "subContinent": "Western…
4  {"continent": "Americas", "subContinent": "Cen…

                                                hits  socialEngagementType   \
0  [{'hitNumber': '1', 'time': '0', 'hour': '17',…  Not Socially Engaged
1  [{'hitNumber': '1', 'time': '0', 'hour': '10',…  Not Socially Engaged
2  [{'hitNumber': '1', 'time': '0', 'hour': '17',…  Not Socially Engaged
3  [{'hitNumber': '1', 'time': '0', 'hour': '9', …  Not Socially Engaged
4  [{'hitNumber': '1', 'time': '0', 'hour': '14',…  Not Socially Engaged

                                              totals  \
0  {"visits": "1", "hits": "1", "pageviews": "1",…
1  {"visits": "1", "hits": "2", "pageviews": "2",…
2  {"visits": "1", "hits": "2", "pageviews": "2",…
3  {"visits": "1", "hits": "2", "pageviews": "2",…
4  {"visits": "1", "hits": "2", "pageviews": "2",…

                                       trafficSource     visitId  visitNumber  \
0  {"campaign": "(not set)", "source": "google", …  1508198450            1
1  {"referralPath": "/a/google.com/transportation…  1508176307            6
2  {"campaign": "(not set)", "source": "(direct)"…  1508201613            1
3  {"campaign": "(not set)", "source": "google", …  1508169851            1
4  {"campaign": "(not set)", "source": "google", …  1508190552            1

   visitStartTime
0      1508198450
1      1508176307
2      1508201613
3      1508169851
4      1508190552
```

There are 12 features in the dataset. However, we found that 4 of the features are json columns, which are `device`, `geoNetwork`, `totals` and `trafficSource`. These json columns contain json format file in each row. To further explore these json columns, we have to normalize them using json_normalize function from pandas library. `customDimensions` and `hits` seem like a list and require further exploration. First of all, we will take a look at json columns.

```
[2]: df_partial = pd.read_csv("train_v2.csv", nrows = 10)
     df_partial.head()
```

```
[2]:   channelGrouping                              customDimensions      date  \
     0  Organic Search             [{'index': '4', 'value': 'EMEA'}]  20171016
     1        Referral    [{'index': '4', 'value': 'North America'}]  20171016
     2          Direct    [{'index': '4', 'value': 'North America'}]  20171016
     3  Organic Search             [{'index': '4', 'value': 'EMEA'}]  20171016
     4  Organic Search  [{'index': '4', 'value': 'Central America'}]  20171016
```

```
                                              device       fullVisitorId  \
0  {"browser": "Firefox", "browserVersion": "not …  3162355547410993243
1  {"browser": "Chrome", "browserVersion": "not a…  8934116514970143966
2  {"browser": "Chrome", "browserVersion": "not a…  7992466427990357681
3  {"browser": "Chrome", "browserVersion": "not a…  9075655783635761930
4  {"browser": "Chrome", "browserVersion": "not a…  6960673291025684308

                                          geoNetwork  \
0  {"continent": "Europe", "subContinent": "Weste…
1  {"continent": "Americas", "subContinent": "Nor…
2  {"continent": "Americas", "subContinent": "Nor…
3  {"continent": "Asia", "subContinent": "Western…
4  {"continent": "Americas", "subContinent": "Cen…

                                                hits  socialEngagementType  \
0  [{'hitNumber': '1', 'time': '0', 'hour': '17',…  Not Socially Engaged
1  [{'hitNumber': '1', 'time': '0', 'hour': '10',…  Not Socially Engaged
2  [{'hitNumber': '1', 'time': '0', 'hour': '17',…  Not Socially Engaged
3  [{'hitNumber': '1', 'time': '0', 'hour': '9', …  Not Socially Engaged
4  [{'hitNumber': '1', 'time': '0', 'hour': '14',…  Not Socially Engaged

                                              totals  \
0  {"visits": "1", "hits": "1", "pageviews": "1",…
1  {"visits": "1", "hits": "2", "pageviews": "2",…
2  {"visits": "1", "hits": "2", "pageviews": "2",…
3  {"visits": "1", "hits": "2", "pageviews": "2",…
4  {"visits": "1", "hits": "2", "pageviews": "2",…

                                       trafficSource     visitId  visitNumber  \
0  {"campaign": "(not set)", "source": "google", …  1508198450            1
1  {"referralPath": "/a/google.com/transportation…  1508176307            6
2  {"campaign": "(not set)", "source": "(direct)"…  1508201613            1
3  {"campaign": "(not set)", "source": "google", …  1508169851            1
4  {"campaign": "(not set)", "source": "google", …  1508190552            1

   visitStartTime
0      1508198450
1      1508176307
2      1508201613
3      1508169851
4      1508190552
```

We can extract 41 new features from the four json columns. However, we noticed that some of the feature has the value of **not available in demo dataset** and **(not set)**. We may need to further explore to determine how many of these kind of values are stored in the dataset. Next, we will explore column `customDimensions` and `hits`.

[4]:

```python
df_partial["customDimensions"]= df_partial["customDimensions"].apply(lambda x:␣
↪json.loads(x.strip("[]").replace("'", "\""))
                                    if "{" in x
                                    else {"index": np.NaN, "value": np.NaN})
df_temp = pd.json_normalize(df_partial["customDimensions"])
df_temp.columns = ["customDimensions.{}".format(sub) for sub in df_temp.columns]
df_partial = df_partial.drop("customDimensions", axis = 1).merge(df_temp,␣
↪right_index = True, left_index = True)
df_partial[df_temp.columns].head()
```

[4]:

```
  customDimensions.index customDimensions.value
0                      4                   EMEA
1                      4          North America
2                      4          North America
3                      4                   EMEA
4                      4        Central America
```

[5]:

```python
df_partial["hits"][0]
```

[5]:

```
"[{'hitNumber': '1', 'time': '0', 'hour': '17', 'minute': '0', 'isInteraction':
True, 'isEntrance': True, 'isExit': True, 'referer': 'https://www.google.co.uk/s
earch?q=water+bottle&ie=utf-8&num=100&oe=utf-8&hl=en&gl=GB&uule=w+CAIQIFISCamRx0
IRO1oCEXoliDJDoPjE&glp=1&gws_rd=cr&fg=1', 'page': {'pagePath':
'/google+redesign/bags/water+bottles+and+tumblers', 'hostname':
'shop.googlemerchandisestore.com', 'pageTitle': 'Water Bottles & Tumblers |
Drinkware | Google Merchandise Store', 'pagePathLevel1': '/google+redesign/',
'pagePathLevel2': '/bags/', 'pagePathLevel3': '/water+bottles+and+tumblers',
'pagePathLevel4': ''}, 'transaction': {'currencyCode': 'USD'}, 'item':
{'currencyCode': 'USD'}, 'appInfo': {'screenName': 'shop.googlemerchandisestore.
com/google+redesign/bags/water+bottles+and+tumblers', 'landingScreenName': 'shop
.googlemerchandisestore.com/google+redesign/bags/water+bottles+and+tumblers',
'exitScreenName': 'shop.googlemerchandisestore.com/google+redesign/bags/water+bo
ttles+and+tumblers', 'screenDepth': '0'}, 'exceptionInfo': {'isFatal': True},
'product': [{'productSKU': 'GGOEGDHC074099', 'v2ProductName': 'Google 17oz
Stainless Steel Sport Bottle', 'v2ProductCategory': 'Home/Drinkware/Water
Bottles and Tumblers/', 'productVariant': '(not set)', 'productBrand': '(not
set)', 'productPrice': '23990000', 'localProductPrice': '23990000',
'isImpression': True, 'customDimensions': [], 'customMetrics': [],
'productListName': 'Category', 'productListPosition': '1'}, {'productSKU':
'GGOEGDHQ015399', 'v2ProductName': '26 oz Double Wall Insulated Bottle',
'v2ProductCategory': 'Home/Drinkware/Water Bottles and Tumblers/',
'productVariant': '(not set)', 'productBrand': '(not set)', 'productPrice':
'24990000', 'localProductPrice': '24990000', 'isImpression': True,
'customDimensions': [], 'customMetrics': [], 'productListName': 'Category',
'productListPosition': '2'}, {'productSKU': 'GGOEYDHJ056099', 'v2ProductName':
'22 oz YouTube Bottle Infuser', 'v2ProductCategory': 'Home/Drinkware/Water
Bottles and Tumblers/', 'productVariant': '(not set)', 'productBrand': '(not
set)', 'productPrice': '4990000', 'localProductPrice': '4990000',
'isImpression': True, 'customDimensions': [], 'customMetrics': [],
'productListName': 'Category', 'productListPosition': '3'}, {'productSKU':
'GGOEGAAX0074', 'v2ProductName': 'Google 22 oz Water Bottle',
'v2ProductCategory': 'Home/Drinkware/Water Bottles and Tumblers/',
'productVariant': '(not set)', 'productBrand': '(not set)', 'productPrice':
'2990000', 'localProductPrice': '2990000', 'isImpression': True,
'customDimensions': [], 'customMetrics': [], 'productListName': 'Category',
'productListPosition': '4'}], 'promotion': [], 'eCommerceAction':
{'action_type': '0', 'step': '1'}, 'experiment': [], 'customVariables': [],
'customDimensions': [], 'customMetrics': [], 'type': 'PAGE', 'social':
{'socialNetwork': '(not set)', 'hasSocialSourceReferral': 'No',
'socialInteractionNetworkAction': ' : '}, 'contentGroup': {'contentGroup1':
'(not set)', 'contentGroup2': 'Bags', 'contentGroup3': '(not set)',
'contentGroup4': '(not set)', 'contentGroup5': '(not set)',
'previousContentGroup1': '(entrance)', 'previousContentGroup2': '(entrance)',
'previousContentGroup3': '(entrance)', 'previousContentGroup4': '(entrance)',
'previousContentGroup5': '(entrance)', 'contentGroupUniqueViews2': '1'},
'dataSource': 'web', 'publisher_infos': []}]"
```

We extracted additional two more features from the `customDimensions` column. Nonetheless, `hits` column contains complicated and unknown information. Therefore, `hits` will be removed from the dataset. In addition, `visitId` will not be used as value to identify distinct user, thus, this feature will be removed as well.

1.1.2 2.1 Loading the Train and Test dataset

```
[6]: def load_df(csv_path, chunksize = 100000):
         json_cols = ["device", "geoNetwork", "totals", "trafficSource"]
         df_reader = pd.read_csv(csv_path,
                                 converters={column: json.loads for column in␣
      ↪json_cols},
                                 dtype = {"fullVisitorId": str},
                                 chunksize = chunksize)
         res = pd.DataFrame()
         for idx , df in enumerate(df_reader):
             df.reset_index(drop = True, inplace = True)
             for col in json_cols:
                 df_temp = pd.json_normalize(df[col])
                 df_temp.columns = ["{}.{}".format(col, subcol) for subcol in␣
      ↪df_temp.columns]
                 df = df.drop(col, axis = 1).merge(df_temp, right_index = True,␣
      ↪left_index = True)
             df['customDimensions'] = df['customDimensions'].apply(ast.literal_eval)
             df['customDimensions'] = df['customDimensions'].str[0]
             df['customDimensions'] = df['customDimensions'].apply(lambda x:␣
      ↪{'index':np.NaN,'value':np.NaN} if pd.isnull(x) else x)
```

```
        column_as_df = pd.json_normalize(df['customDimensions'])
        column_as_df.columns = ["customDimensions.{}".format(subcol) for subcol␣
 ↪in column_as_df.columns]
        df = df.drop('customDimensions', axis=1).merge(column_as_df,␣
 ↪right_index=True, left_index=True)
        df.drop("hits", axis = 1, inplace = True)
        df.drop("visitId", axis = 1, inplace = True)
        res = pd.concat([res, df], axis = 0).reset_index(drop = True)
        del df
        gc.collect()
    return res
```

[7]:

```
%%time
df_train = load_df('train_v2.csv')
```

```
Wall time: 12min 43s
```

[8]:

```
%%time
df_test = load_df('test_v2.csv')
```

```
Wall time: 3min 10s
```

[9]:

```
print("The dataset consists of {} entries and {} features".format(df_train.
 ↪shape[0], df_train.shape[1]))
```

```
The dataset consists of 1708337 entries and 59 features
```

1.1.3 2.2 Dataset Preprocessing

Convert rafficSource.isTrueDirect and trafficSource.adwordsClickInfo.isVideoAd into boolean type. This step is taken first because it will cause error to the next data cleaning step while using string search.

[10]:

```
# Convert isTrueDirect and isVideoAd to boolean
#https://stackoverflow.com/questions/48350125/
 ↪pandas-conversion-from-object-to-boolean-always-returns-true-using-astype
df_train['trafficSource.isTrueDirect'] = df_train['trafficSource.isTrueDirect']␣
 ↪== True
df_train['trafficSource.adwordsClickInfo.isVideoAd'] = df_train['trafficSource.
 ↪adwordsClickInfo.isVideoAd'] == True

df_test['trafficSource.isTrueDirect'] = df_test['trafficSource.isTrueDirect']␣
 ↪== True
df_test['trafficSource.adwordsClickInfo.isVideoAd'] = df_test['trafficSource.
 ↪adwordsClickInfo.isVideoAd'] == True
```

From the previous exploration, we found that a lot of **not available in demo dataset** in some of the columns. This value is meaningless for model training. Therefore, if any one of the columns has more than 1 million of this value, this column will be discarded from the dataset.

[11]:

```
# Search for columns with more than 1 million "not available in demo dataset" entries
unavailable_cols = []
threshold = 1000000

for i in df_train.select_dtypes(include = [np.object]).columns:
    if(df_train[i].str.contains("not available in demo dataset").sum() > threshold):
        print(i)
        unavailable_cols.append(i)
```

```
device.browserVersion
device.browserSize
device.operatingSystemVersion
device.mobileDeviceBranding
device.mobileDeviceModel
device.mobileInputSelector
device.mobileDeviceInfo
device.mobileDeviceMarketingName
device.flashVersion
device.language
device.screenColors
device.screenResolution
geoNetwork.cityId
geoNetwork.latitude
geoNetwork.longitude
geoNetwork.networkLocation
trafficSource.adwordsClickInfo.criteriaParameters
```

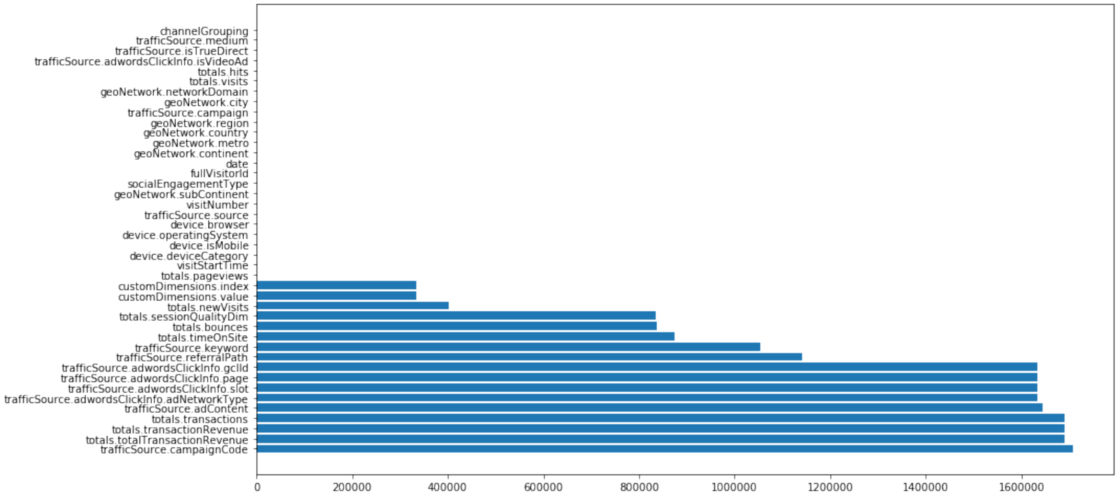
[12]:

```
# Remove the unavailable data columns
df_train.drop(columns = unavailable_cols, axis = 1, inplace = True)

df_test.drop(columns = unavailable_cols, axis = 1, inplace = True)
```

Next, we will visualize missing values of the dataset.

[13]:

```
# Visualize missing values
plt.figure(figsize=[14.70, 8.27])
plt.barh(df_train.isna().sum().sort_values(ascending = False).index,
         df_train.isna().sum().sort_values(ascending = False).values)
plt.show()
```



Remove `trafficSource.campaignCode` since there is only a non-missing value in this column and not available in test set.

```
[14]: # Remove columns
      df_train.drop(columns = 'trafficSource.campaignCode', axis = 1, inplace = True)
```

As mentioned previously, there are some missing values recorded in some of the entries, namely, **(not set)**, **not available in demo dataset**, **(not provided)**, **unknown.unknown**, **/** and **(none)**. Thus, we will replace these values with np.nan.

```
[15]: # Replace all empty fields
      def replace_empty(df):
          nulls = ['(not set)', 'not available in demo dataset', '(not provided)',
                   'unknown.unknown', '/', '(none)']

          for null in nulls:
              df.replace(null, np.nan, inplace = True)
```

```
[16]: replace_empty(df_train)

      replace_empty(df_test)
```

Next, we will explore each column one by one.

1. `channelGrouping` **STRING**

   The Default Channel Group associated with an end user's session for this View.

2. `date` **STRING**

   The date of the session in YYYYMMDD format.

3. `fullVisitorId` **STRING**

The unique visitor ID (also known as client ID).

4. `socialEngagementType` **STRING**

   Engagement type, either "Socially Engaged" or "Not Socially Engaged".

```
[17]: df_train["socialEngagementType"].unique()
```

```
[17]: array(['Not Socially Engaged'], dtype=object)
```

This column only contains one unique value which is 'Not Socially Engaged'. Therefore, it can be removed.

```
[18]: df_train.drop(columns = 'socialEngagementType', axis = 1, inplace = True)

      df_test.drop(columns = 'socialEngagementType', axis = 1, inplace = True)
```

5. `visitNumber` **INTEGER**

   The session number for this user. If this is the first session, then this is set to 1.

```
[19]: df_train['visitNumber'] = df_train['visitNumber'].astype(np.int32)

      df_test['visitNumber'] = df_test['visitNumber'].astype(np.int32)
```

6. `visitStartTime` **INTEGER**

   The timestamp (expressed as POSIX time).

Currently `date` and `visitStartTime` are string data type. To add more date features into the dataset, date related features are extracted from both `date` and `visitStartTime`.

```
[20]: # Define function to extract date features
      def date_feature(df):
          df["date"] = pd.to_datetime(df["date"], format="%Y%m%d") # seting the␣
       ↪column as pandas datetime
          df["visit_weekday"] = df['date'].dt.strftime('%A') #extracting week day
          df["visit_day"] = df['date'].dt.strftime("%d") # extracting day
          df["visit_month"] = df['date'].dt.strftime('%B') # extracting month
          df["visit_year"] = df['date'].dt.strftime("%Y") # extracting year
          df['visit_hour'] = df['visitStartTime'].apply(lambda x:
                                                     str(datetime.fromtimestamp(x,␣
       ↪pytz.timezone("UTC")).hour)) # extracting hour
```

```
[21]: # Apply function
      date_feature(df_train)
      date_feature(df_test)
```

7. `device.browser` **STRING**

   The browser used (e.g., "Chrome" or "Firefox").

```
[22]: df_train["device.browser"].unique()
```

```
[22]: array(['Firefox', 'Chrome', 'Safari', 'UC Browser', 'Internet Explorer',
             'Edge', 'Samsung Internet', 'Android Webview', 'Safari (in-app)',
             'Opera Mini', 'Opera', 'YaBrowser', 'Amazon Silk',
             'Mozilla Compatible Agent', 'Puffin', 'Maxthon', 'BlackBerry',
             'ADM', 'Coc Coc', 'MRCHROME', 'Android Browser',
             'Playstation Vita Browser', 'Nintendo Browser', 'Nokia Browser',
             'SeaMonkey', 'Lunascape', 'IE with Chrome Frame', 'ThumbSniper',
             'LYF_LS_4002_12', 'DESKTOP', 'Mozilla', 'Browser',
             'osee2unifiedRelease', 'Seznam', nan,
             ';__CT_JOB_ID__:65da7e5f-0f05-4b5d-8d31-1f4d470a2b82;',
             'Apple-iPhone7C2',
             ';__CT_JOB_ID__:a80e8e16-6e98-455b-885a-a4dd40f3d344;',
             ';__CT_JOB_ID__:89e59554-ad41-4e94-957b-f12bd012530c;',
             'DDG-Android-3.1.1', 'NokiaE52-1', 'Iron',
             '[Use default User-agent string] LIVRENPOCHE', 'Konqueror',
             ';__CT_JOB_ID__:7e575295-571e-4e82-9254-7f2c8bbb9183;',
             'LYF_LS_4002_11', 'M5', 'Android Runtime', 'Changa 99695759', 'YE',
             'no-ua', '+Simple Browser', 'MQQBrowser', 'Nichrome',
             'tfowdqmibyshaklxuregpcnzvj', 'Autn-WKOOP', 'HTC802t_TD',
             ';__CT_JOB_ID__:a4f837b8-8d78-4c42-ba9a-d870cf1a4a7e;',
             ';__CT_JOB_ID__:58e2ecba-7666-4a10-b498-8216457ce472;',
             ';__CT_JOB_ID__:2547db0b-ec43-452a-a0d4-ff42b7dc7907;',
             ';__CT_JOB_ID__:dd6177aa-1baa-4007-9b38-b7cab4f7611c;',
             ';__CT_JOB_ID__:d14534ff-e2fc-4692-92aa-e34508f1c418;',
             ';__CT_JOB_ID__:4333777f-bb0c-4a18-935e-df5658dbce2d;', 'Netscape',
             ';__CT_JOB_ID__:6e9dcf2f-f58f-4938-91e3-77e00868177b;',
             'Amazon.com', 'DASH_JR_3G', 'DoCoMo', 'subjectAgent: NoticiasBoom',
             'vjebamzrktwcysxpdlonhiufqg', 'jdbknvrluyeaxoipgwczmthsqf',
             'flobzsdixhuwqakptjmcrveygn', 'epxmjusghnvircdfkwqlotzbay',
             'njroiedbwpmvykqlatxzuhcfgs', 'CSM Click',
             'SAMSUNG-SM-B355E Opera', 'flwadqukonrjegpbisyxztvhcm',
             'ejpxuidzlmagvthsfbqnkwyocr', 'lhkbrtuwomdeafnqygvxcspizj',
             'ighfsbrmpoctzjqxlywdenvuka', 'starmaker',
             'cnwmpegudakrqzljtvfxohbysi', 'wfpknuqxovyilmrdzbhgtecjas',
             'User Agent', '0',
             ';__CT_JOB_ID__:76fd1acb-e365-43c0-b967-908bcf5d5b59;',
             ';__CT_JOB_ID__:a24a8978-e5e8-4dc9-af66-c4ed89ea25d7;',
             ';__CT_JOB_ID__:85da5736-a78e-45a9-837e-f5a53e5cd725;',
             ';__CT_JOB_ID__:a7ed0808-e70c-4b19-b1a3-1018bbb7dc7f;',
             ';__CT_JOB_ID__:2e0eca60-83ab-482d-bb81-343d113254fb;',
             'ecgiwapzltrkujdhmqsbxfonvy', 'Hisense M20-M_LTE',
             'eosutpkiahjzvdgcwxlmyfqbrn', 'ujvrzsonxihlgaqdmkwtbfcpey',
             'NokiaC7-00',
             ';__CT_JOB_ID__:0a075729-93a5-43d0-9638-4cbd41d5f5a5;',
             'bsfnwveckhgpdoyjxmizruqtla', 'efkaxnbyohqtspzlvcwrjmigdu',
             'wvsmagudcqeytijorlhxnfzkbp', 'rpfanjzoxyemsgbtichqkudwlv',
             'cajrnbtvqwfkolzyxushpdgime', 'ohfgqlpiuyknvmbctszjarxdwe',
             'jscatcher', 'Dillo', 'Reddit', 'ecwozghsufybtdkjrlvxpamiqn',
             'uhdypcxbgzajmeqwlofnrsitkv', 'hbijxvdyrgnatwzmlcpkfusqoe',
             'lpmqaxwbzyteokrfusnjhvdigc', 'wncrmxukofqljsgvzahiybpdet',
             'ajsqixbltuvwpmdcokfyzhgren', 'dohyinzpvbsktjeguxmrqcwafl',
             'uybjlgntzwpacihremkqsxdovf', 'rbydojcflwzvnuaepmsgxhiktq',
             'afjurnqyolshpibxczdwktmvge', 'wdhtapevfnqzskcroxgjmiybul',
             'mhwxofpevcagujznbsiqlrkytd',
             ';__CT_JOB_ID__:fe02e46f-b6ae-41f1-8563-3b40bbb623a9;',
             ';__CT_JOB_ID__:0b39e7ca-1431-42e3-ba1f-9d8951a65840;',
             'KINGSUN-F4', 'lxjwoyfivgdbkqtuzsrmhencpa',
             'zurcqesbhljxmpwdgnvkoyafit', 'TCL P500M',
             'kqebrzuwmiycxdvtoljnhsfpga', 'dkagwlhmfqxercuozpnbvtsiyj',
             'ohukwejvqmdtibfrzpycgxanls',
             ';__CT_JOB_ID__:97909e28-4228-4b55-8ad5-cc791f2b583c;',
             'ymzsbiduaejrchvxlwkfnqgtop', 'fspmihbxzowgnuctrqykjlvade'],
            dtype=object)
```

There are some unknown browser type under the `device.browser` column. For the ease of analysis, these unknown browsers are categorized as bot.

[23]:
```python
# Define function to categorize device_browser
def categorize_browser(x):

    if 'Chrome' == x:
        return 'Chrome'
    elif 'Safari' == x:
        return 'Safari'
    elif 'Firefox' == x:
        return 'Firefox'
    elif 'Internet Explorer' == x:
        return 'Internet Explorer'
    elif 'Android Webview' == x:
        return 'Android Webview'
    elif 'Edge' == x:
        return 'Edge'
    elif 'Samsung Internet' == x:
        return 'Samsung Internet'
    elif 'Opera Mini' == x:
        return 'Opera Mini'
    elif 'Safari (in-app)' == x:
        return 'Safari (in-app)'
    elif 'Opera' == x:
        return 'Opera'
    elif 'UC Browser' == x:
        return 'UC Browser'
    elif 'YaBrowser' == x:
        return 'YaBrowser'
    elif 'Amazon Silk' == x:
        return 'Amazon Silk'
    elif 'Coc Coc' == x:
        return 'Coc Coc'
    elif 'Android Browser' == x:
        return 'Android Browser'
    elif 'Maxthon' == x:
        return 'Maxthon'
    elif 'Puffin' == x:
        return 'Puffin'
    elif 'BlackBerry' == x:
        return 'BlackBerry'
    elif 'Nintendo Browser' == x:
        return 'Nintendo Browser'
    elif 'Nokia Browser' == x:
        return 'Nokia Browser'
    elif 'Iron' == x:
        return 'Iron'
    elif 'SeaMonkey' == x:
        return 'SeaMonkey'
    elif 'Mozilla' == x:
        return 'Mozilla'
    elif 'Seznamr' == x:
        return 'Seznam'
    elif 'Playstation Vita Browser' == x:
        return 'Playstation Vita Browser'
    elif 'Lunascape' == x:
        return 'Lunascape'
    elif '+Simple Browser' == x:
        return '+Simple Browser'
    elif 'Konqueror' == x:
        return 'Konqueror'
    elif 'Android Runtime' == x:
        return 'Android Runtime'
    else:
        return 'Bot'
```

[24]:

```
df_train['device.browser'] = df_train['device.browser'].apply(lambda x:␣
 ↪categorize_browser(str(x)))

df_test['device.browser'] = df_test['device.browser'].apply(lambda x:␣
 ↪categorize_browser(str(x)))
```

8. `device_operatingSystem` **STRING**

   The operating system of the device (e.g., “Macintosh” or “Windows”).

```
[25]: df_train["device.operatingSystem"].unique()
```

```
[25]: array(['Windows', 'Chrome OS', 'Android', 'Macintosh', 'iOS', 'Linux',
              nan, 'Windows Phone', 'Samsung', 'Tizen', 'BlackBerry', 'OS/2',
              'Playstation Vita', 'Xbox', 'Nintendo Wii', 'Firefox OS',
              'Nintendo 3DS', 'Nintendo WiiU', 'SymbianOS', 'FreeBSD', 'Nokia',
              'OpenBSD', 'SunOS', 'NTT DoCoMo'], dtype=object)
```

9. `device.isMobile` **BOOLEAN**

   If the user is on a mobile device, this value is true, otherwise false.

10. `device.deviceCategory` **STRING**

    The type of device (Mobile, Tablet, Desktop).

```
[26]: df_train["device.deviceCategory"].unique()
```

```
[26]: array(['desktop', 'mobile', 'tablet'], dtype=object)
```

11. `geoNetwork.continent` **STRING**

    The continent from which sessions originated, based on IP address.

```
[27]: df_train["geoNetwork.continent"].unique()
```

```
[27]: array(['Europe', 'Americas', 'Asia', 'Oceania', nan, 'Africa'],
            dtype=object)
```

12. `geoNetwork.subContinent` **STRING**

    The sub-continent from which sessions originated, based on IP address of the visitor.

```
[28]: df_train["geoNetwork.subContinent"].unique()
```

```
[28]: array(['Western Europe', 'Northern America', 'Western Asia',
              'Central America', 'Northern Europe', 'Southern Asia',
              'Southeast Asia', 'Eastern Europe', 'South America',
              'Eastern Asia', 'Southern Europe', 'Australasia', 'Central Asia',
              nan, 'Northern Africa', 'Eastern Africa', 'Southern Africa',
              'Western Africa', 'Caribbean', 'Middle Africa', 'Melanesia',
              'Micronesian Region', 'Polynesia'], dtype=object)
```

13. `geoNetwork.country` **STRING**

    The country from which sessions originated, based on IP address.

```
[29]: df_train["geoNetwork.country"].unique()
```

```
[29]: array(['Germany', 'United States', 'Turkey', 'Mexico', 'United Kingdom',
              'Denmark', 'Netherlands', 'Sweden', 'Canada', 'India', 'Belgium',
              'Philippines', 'Slovakia', 'Brazil', 'Japan', 'Taiwan', 'Peru',
              'Ireland', 'Norway', 'Romania', 'Russia', 'Italy', 'New Zealand',
              'Czechia', 'Serbia', 'Argentina', 'Australia', 'Hong Kong',
              'Indonesia', 'Singapore', 'Kazakhstan', 'Thailand', 'Ecuador',
              'Switzerland', 'Spain', 'France', 'Malaysia', 'Poland', 'Bulgaria',
              'Jordan', 'China', 'Pakistan', nan, 'Israel', 'Vietnam',
              'Bangladesh', 'Greece', 'Algeria', 'Georgia', 'Ukraine',
              'South Korea', 'Austria', 'Ethiopia', 'Colombia', 'Sudan', 'Egypt',
              'United Arab Emirates', 'Panama', 'Portugal', 'Latvia', 'Chile',
              'Belarus', 'South Africa', 'El Salvador', 'Nigeria', 'Venezuela',
              'Sri Lanka', 'Estonia', 'Croatia', 'Myanmar (Burma)', 'Lithuania',
              'Armenia', 'Puerto Rico', 'Saudi Arabia', 'Dominican Republic',
              'Finland', 'Hungary', 'Cambodia', 'Qatar', 'Tunisia', 'Morocco',
              'Mongolia', 'Rwanda', 'Afghanistan', 'Trinidad & Tobago',
              'Bolivia', 'Zambia', 'Iraq', 'Guatemala', 'Honduras', 'Yemen',
              'Tanzania', 'Oman', 'Greenland', 'Kuwait', 'French Guiana',
              'Réunion', 'Kosovo', 'Curaçao', 'Malta', 'Montenegro', 'Slovenia',
              'Kenya', 'Moldova', 'Costa Rica', 'Bosnia & Herzegovina',
              'Paraguay', 'Botswana', 'Uruguay', 'Jamaica', 'Gambia',
              'Madagascar', 'Togo', 'Lebanon', 'Libya', 'Uzbekistan',
              'Mauritius', 'Cyprus', 'Macedonia (FYROM)', 'Albania', 'Bahrain',
              'Turks & Caicos Islands', 'Zimbabwe', 'Ghana', 'Cape Verde',
              'Senegal', 'Côte d’Ivoire', 'Laos', 'Azerbaijan', 'Barbados',
              'Uganda', 'Nepal', 'Mali', 'Mauritania', 'Nicaragua', 'Iceland',
              'Palestine', 'Haiti', 'St. Kitts & Nevis', 'Somalia', 'Cameroon',
              'Namibia', 'Congo - Kinshasa', 'New Caledonia', 'Kyrgyzstan',
              'Luxembourg', 'Benin', 'Guinea', 'Guam', 'San Marino', 'Liberia',
              'Malawi', 'Angola', 'Guyana', 'Brunei', 'Guadeloupe', 'Belize',
              'Maldives', 'Guinea-Bissau', 'Mozambique', 'Gabon', 'Macau',
              'Burkina Faso', 'Tajikistan', 'Martinique', 'Congo - Brazzaville',
              'French Polynesia', 'Fiji', 'St. Lucia', 'Iran', 'Monaco',
              'Swaziland', 'Bahamas', 'Burundi', 'Turkmenistan',
              'Papua New Guinea', 'Liechtenstein', 'Bermuda', 'Guernsey',
              'Northern Mariana Islands', 'Antigua & Barbuda', 'Sint Maarten',
              'Niger', 'South Sudan', 'Jersey', 'Andorra',
              'St. Vincent & Grenadines', 'Bhutan', 'Cayman Islands',
              'Faroe Islands', 'Chad', 'Suriname', 'Djibouti', 'Syria',
              'Gibraltar', 'Lesotho', 'U.S. Virgin Islands', 'Mayotte', 'Aruba',
              'Equatorial Guinea', 'Grenada', 'Norfolk Island', 'Isle of Man',
              'Caribbean Netherlands', 'Vanuatu', 'Sierra Leone',
              'Åland Islands', 'St. Pierre & Miquelon', 'British Virgin Islands',
              'Samoa', 'Timor-Leste', 'Comoros', 'Solomon Islands', 'St. Martin',
              'Montserrat', 'Cook Islands', 'St. Helena', 'American Samoa',
              'Dominica', 'Seychelles', 'Anguilla', 'Tonga', 'Marshall Islands',
              'Central African Republic', 'Micronesia', 'São Tomé & Príncipe',
              'St. Barthélemy', 'Eritrea'], dtype=object)
```

14. `geoNetwork.region` **STRING**

    The region from which sessions originate, derived from IP addresses. In the U.S., a region is a state, such as New York.

```
[30]: df_train["geoNetwork.region"].unique()
```

```
[30]: array([nan, 'California', 'England', 'Mexico City', 'Nevada', 'Brussels',
              'Tokyo', 'County Dublin', 'Maharashtra', 'Istanbul', 'Ontario',
              'Telangana', 'Pennsylvania', 'Michigan', 'Massachusetts',
              'British Columbia', 'Madhya Pradesh', 'Quebec', 'New South Wales',
              'Jakarta', 'New York', 'State of Sao Paulo', 'Washington',
              'District of Columbia', 'Chiayi County', 'Delhi', 'Karnataka',
              'Bangkok', 'Aragon', 'Zurich', 'Masovian Voivodeship', 'Texas',
              'Georgia', 'Illinois', 'Tamil Nadu', 'Sindh', 'Lombardy',
              'Federal Territory of Kuala Lumpur', 'Saint Petersburg',
              'Tennessee', 'Hanoi', 'Taipei City', 'Madrid', 'Berlin',
              'Ho Chi Minh City', 'Victoria', 'Seoul', 'Ile-de-France', 'Lisbon',
              'Bogota', 'New Taipei City', 'Stockholm County',
              'Western Province', 'Lagos', 'Riyadh Province', 'Dubai',
              'Colorado', 'Buenos Aires', 'Lima Region',
              'Santiago Metropolitan Region', 'Dublin City', 'North Holland',
              'Virginia', 'Community of Madrid', 'West Bengal', 'Catalonia',
              'State of Rio de Janeiro', 'Queensland', 'Moscow', 'Izmir',
              'Lazio', 'Tel Aviv District', 'Ho Chi Minh', 'Nuevo Leon',
              'Ankara', 'Metro Manila', 'Taichung City', 'Wisconsin',
              'Dhaka Division', 'Tainan City', 'Nouvelle-Aquitaine',
              'Taoyuan County', 'North Carolina', 'Cusco', 'Budapest', 'Montana',
              'Osaka Prefecture', 'Capital Region of Denmark', 'Bavaria',
              'Uttar Pradesh', 'Auckland', 'Auvergne-Rhone-Alpes',
              'Porto District', 'County Cork', 'Walloon Region',
              'Kanagawa Prefecture', 'Oregon', 'Kyiv city', 'Hesse', 'Prague',
              'Moravian-Silesian Region', 'Hamburg', 'Usti nad Labem Region',
              'Hradec Kralove Region', 'Beijing', 'Arizona', 'Zhejiang',
              'Rajasthan', 'Ohio', 'Selangor', 'City of Zagreb', 'Malacca',
              'Gauteng', 'Attica', 'Minnesota', 'Vastra Gotaland County',
              'Lower Silesian Voivodeship', 'Gujarat', 'Western Cape', 'Assam',
              'Vienna', 'Makkah Province', 'Alberta', 'Haryana', 'Veneto',
              'Mures County', 'Greater Poland Voivodeship', 'Bucharest',
              'Timis County', 'West Java', 'Central Visayas', 'Hauts-de-France',
              'Florida', 'Andalusia', 'Western Australia', 'New Jersey',
              'Chandigarh', 'East Java', 'State of Minas Gerais', 'Quang Ngai',
              'Nakhon Pathom', 'Khon Kaen', 'Ba Ria - Vung Tau', 'Da Nang',
              'Grand Casablanca', 'Haiphong', 'Prachuap Khiri Khan',
              'Nakhon Sawan', 'Chon Buri', 'Federal District', 'Thai Nguyen',
              'Dong Nai', 'Bursa', 'Rayong', 'Chiang Mai', 'State of Parana',
              'Nakhon Ratchasima', 'Phra Nakhon Si Ayutthaya',
              'Khanh Hoa Province', 'Lam Djong', 'Bihor County',
'Castile-La Mancha', 'Cairo Governorate',
'Lesser Poland Voivodeship', 'Grand Est', 'Shanghai', 'Queretaro',
'Flanders', 'Adana', 'Surat Thani', 'State of Bahia',
'State of Rio Grande do Sul', 'Cluj County', 'Djak Lak Province',
'Antioquia', 'Udon Thani', 'Hai Duong', 'Songkhla',
'Binh Dinh Province', 'Tbilisi', 'Bac Giang', 'Bac Ninh Province',
'Tien Giang', 'Can Tho', 'Eskisehir Province', 'Bihar',
'Valencian Community', 'Basque Country', 'Davao Region',
'Dnipropetrovsk Oblast', 'Mersin Province', 'Konya', 'Saraburi',
'Antalya', "Provence-Alpes-Cote d'Azur", 'Indiana',
'Leiria District', 'South Australia', 'Waikato',
'Bratislava Region', 'Perak', 'Kaohsiung City', 'Riga',
'Hsinchu County', 'Giza Governorate', 'KwaZulu-Natal',
'Sohag Governorate', 'Alexandria Governorate', 'Pays de la Loire',
'Emilia-Romagna', 'Punjab', 'South Holland', 'Idaho', 'Iowa',
'Vojvodina', 'Utah', 'Odisha', 'North Rhine-Westphalia',
'Amman Governorate', 'Piedmont', 'Jalisco', 'Andhra Pradesh',
'Oklahoma', 'North Brabant', 'North Sumatra', 'Baja California',
'Chihuahua', 'Tamaulipas', 'Geneva', 'Harju County', 'Nebraska',
'South Sulawesi', 'Scotland', 'Vilnius County', 'Maryland',
'South Carolina', 'Brittany', 'Changhua County',
'Central Macedonia', 'Abu Dhabi', 'Murcia', 'Lower Saxony',
'Gelderland', 'Akershus', 'Kentucky', 'Missouri', 'Zulia',
'Utrecht', 'Oslo', 'Galicia', 'Minsk Region', 'Center District',
'Canary Islands', 'County Carlow', 'Kansas', 'Gia Lai Province',
'Quang Nam Province', 'Thai Binh', 'Nam Dinh', 'Setubal',
'Guatemala Department', 'Kerala', 'Sverdlovsk Oblast',
'Thua Thien Hue', 'Djong Thap Province', 'Primorsky Krai',
'Almaty Province', 'Pathum Thani', 'Aydin Province', 'Panama',
'Valle del Cauca', 'Capital District', 'Gaziantep',
'Baghdad Governorate', 'Red Sea Governorate', 'Diyarbakir',
'Ubon Ratchathani', 'Sofia City Province', 'Louisiana',
'Balearic Islands', 'Kharkiv Oblast', 'Odessa Oblast',
'Pingtung County', 'Salzburg', 'County Wicklow',
'Baden-Wurttemberg', 'Aguascalientes', 'State of Mato Grosso',
'Ouest Department', 'Distrito Nacional', 'Managua Department',
'State of Goias', 'San Jose Province', 'Manitoba',
'Bamako Capital District', 'Pichincha', 'State of Pernambuco',
'Santiago Province', 'State of Rio Grande do Norte',
'San Salvador Department', 'Cordoba', 'Santo Domingo Province',
'State of Amazonas', 'Caldas', 'State of Para', 'Puebla',
'Sinaloa', 'Santa Cruz Department', 'State of Ceara',
'State of Mexico', 'San Juan', 'Dakar Region', 'Guanajuato',
'Yucatan', 'Atlantico', 'Francisco Morazan Department',
'State of Maranhao', 'La Libertad', 'State of Alagoas',
'St. Andrew Parish', 'State of Sergipe', 'La Paz Department',
'Alabama', 'Santa Fe Province', 'Mendoza Province', 'Iasi County',
'Wellington', 'Central Denmark Region', 'Hawaii',
'Eastern Province', 'Centre-Val de Loire', 'Region Zealand',
'Vladimir Oblast', 'Doha', 'Hung Yen Province', 'Kayseri Province',
'Binh Phuoc', 'Miyazaki Prefecture', 'Bremen', 'Nairobi County',
'Chiang Rai', 'Trabzon', 'Piura', 'Samsun', 'Phitsanulok',
'North Denmark Region', 'Vinh Phuc Province', 'Nghe An',
'Phu Tho Province', 'Arequipa', 'Overijssel', 'Southern Province',
'Newfoundland and Labrador', 'Phuket', 'Tatarstan', 'Sibiu',
'Oran Province', 'Tay Ninh Province', 'Gangwon-do',
'Lower Austria', 'Erzurum', 'Sakon Nakhon', 'Algiers Province',
'Al Madinah Province', 'Johor', 'Aquitaine',
'Silesian Voivodeship', 'Fukui Prefecture', 'Kien Giang',
'State of Espirito Santo', 'South Moravian Region',
'Kosice Region', 'Aveiro District', 'Corsica', 'Ljubljana',
'Region of Southern Denmark', 'Asturias', 'Saxony',
'Buenos Aires Province', 'Hunedoara County', 'Thanh Hoa',
'Guangdong', 'New Brunswick', 'Islamabad Capital Territory',
'Cantabria', 'Thuringia', 'Nagano Prefecture', 'Vaud',
'County Louth', 'Sonora', 'Tuscany', 'Binh Thuan Province',
'Upper Austria', 'Skane County', 'Aust-Agder', 'Occitanie',
'Region Syddanmark', 'State of Santa Catarina', 'Orebro County',
'Quintana Roo', 'Olomouc Region', 'Campania', 'Lviv Oblast',
'Busan', 'Phnom Penh', 'Greater Accra Region', 'Yangon Region',
'Federation of Bosnia and Herzegovina', 'Vientiane Prefecture',
'Aichi Prefecture', 'Hokkaido', 'Fukuoka Prefecture',
'Hyogo Prefecture', 'Miyagi Prefecture', 'Saitama Prefecture',
'Gifu Prefecture', 'Republic of Bashkortostan',
'Nizhny Novgorod Oblast', 'Sicily', 'Krasnodar Krai',
'Irbid Governorate', 'Daegu', 'Nord-Pas-de-Calais', 'Sharjah',
'Split-Dalmatia County', 'Beirut Governorate', 'Lampang',
'Kyoto Prefecture', 'Santander Department', 'Alaska',
'Gyeonggi-do', 'Brest Region', 'Alba County', 'Lopburi',
'Nakhon Si Thammarat', 'Dakahlia Governorate',
'Viana do Castelo District', 'Northern Ireland',
'Okinawa Prefecture', 'Menofia Governorate', 'Jonkoping County',
'Tyrol', 'Connecticut', 'Podkarpackie Voivodeship',
'Miaoli County', 'Kumamoto Prefecture',
'West Pomeranian Voivodeship', 'Wales', 'Prince Edward Island',
'Maha Sarakham', 'Special Region of Yogyakarta', 'Abruzzo',
'Australian Capital Territory', 'Meghalaya', 'Basra Governorate',
'Castile and Leon', 'Central Java', 'Ha Tinh Province',
'Pomeranian Voivodeship', 'Apulia', 'Braga', 'Sardinia',
'Pazardzik', 'Osijek-Baranja County', 'Lucerne', 'West Virginia',
'Groningen', 'Faro District', 'Medjimurje County', 'Delaware',
'Schleswig-Holstein', 'Hordaland', 'Ibaraki Prefecture',
'Beni Suef Governorate', 'Cundinamarca', 'Valparaiso Region',
'Magdalena', 'Rostov Oblast', 'Perm Krai', 'Gharbia Governorate',
       'New Hampshire', 'Rhone-Alpes'], dtype=object)
```

15. geoNetwork.metro **STRING**

    The Designated Market Area (DMA) from which sessions originate.

```
[31]: df_train["geoNetwork.metro"].unique()
```

```
[31]: array([nan, 'San Francisco-Oakland-San Jose CA', 'London', 'JP_KANTO',
       'Los Angeles CA', 'Pittsburgh PA', 'Detroit MI',
       'Boston MA-Manchester NH', 'New York NY', 'Seattle-Tacoma WA',
       'Washington DC (Hagerstown MD)', 'San Antonio TX', 'Atlanta GA',
       'Chicago IL', 'Dallas-Ft. Worth TX', 'Philadelphia PA',
       'San Diego CA', 'Austin TX', 'Nashville TN', 'Houston TX',
       'Yorkshire', 'Denver CO', 'Roanoke-Lynchburg VA',
       'La Crosse-Eau Claire WI', 'Charlotte NC', 'Butte-Bozeman MT',
       'JP_KINKI', 'Portland OR', 'Phoenix AZ', 'Columbus OH',
       'Minneapolis-St. Paul MN', 'North West', 'Jacksonville FL',
       'Meridian (exc. Channel Islands)',
       'Orlando-Daytona Beach-Melbourne FL', 'Las Vegas NV', 'Midlands',
       'Springfield-Holyoke MA', 'Green Bay-Appleton WI',
       'Harlingen-Weslaco-Brownsville-McAllen TX', 'Indianapolis IN',
       'Chico-Redding CA', 'Norfolk-Portsmouth-Newport News VA',
       'East Of England', 'Lansing MI', 'Idaho Falls-Pocatello ID',
       'Omaha NE', 'Salt Lake City UT', 'Miami-Ft. Lauderdale FL',
       'Oklahoma City OK', 'Raleigh-Durham (Fayetteville) NC',
       'Tampa-St. Petersburg (Sarasota) FL', 'Memphis TN',
       'Sacramento-Stockton-Modesto CA', 'Central Scotland',
       'Charleston SC', 'Boise ID', 'Louisville KY', 'St. Louis MO',
       'Cleveland-Akron (Canton) OH',
       'Paducah KY-Cape Girardeau MO-Harrisburg-Mount Vernon IL',
       'Milwaukee WI', 'Tulsa OK',
       'Greenville-Spartanburg-Asheville-Anderson',
       'Albany-Schenectady-Troy NY', 'El Paso TX', 'Kansas City MO',
       'Fresno-Visalia CA', 'New Orleans LA', 'North East',
       'Springfield MO', 'Baltimore MD', 'Madison WI',
       'Greenville-New Bern-Washington NC', 'Dayton OH', 'Ft. Wayne IN',
       'Cincinnati OH', 'Birmingham (Ann and Tusc) AL',
       'Des Moines-Ames IA', 'Lexington KY',
       'Grand Rapids-Kalamazoo-Battle Creek MI', 'Honolulu HI',
       'North Scotland', 'Sioux City IA', 'Buffalo NY', 'Mankato MN',
       'Tri-Cities TN-VA', 'Columbus GA', 'Spokane WA',
       'Tucson (Sierra Vista) AZ', 'Wilkes Barre-Scranton PA',
       'Chattanooga TN', 'West Palm Beach-Ft. Pierce FL', 'JP_OTHER',
       'Monterey-Salinas CA', 'Wichita-Hutchinson KS',
       'Lincoln & Hastings-Kearney NE', 'Tallahassee FL-Thomasville GA',
       'JP_CHUKYO', 'Richmond-Petersburg VA', 'Reno NV', 'Anchorage AK',
       'Toledo OH', 'Providence-New Bedford,MA',
       'Champaign & Springfield-Decatur IL', 'Panama City FL', 'Ulster',
       'Lubbock TX', 'Hartford & New Haven CT', 'HTV West', 'HTV Wales',
       'Colorado Springs-Pueblo CO', 'Syracuse NY',
       'Rochester-Mason City-Austin,IA', 'Utica NY',
       'Flint-Saginaw-Bay City MI', 'Charlottesville VA', 'Augusta GA',
       'Wheeling WV-Steubenville OH', 'Abilene-Sweetwater TX',
       'Rochester NY', 'Erie PA'], dtype=object)
```

16. `geoNetwork.city` **STRING**

    Users' city, derived from their IP addresses or Geographical IDs.

17. `geoNetwork.networkDomain` **STRING**

    The domain name of user's ISP, derived from the domain name registered to the ISP's IP address.

18. `totals.visits` **INTEGER**

    The number of sessions (for convenience). This value is 1 for sessions with interaction events. The value is null if there are no interaction events in the session.

[32]:
```
df_train["totals.visits"].unique()
```

[32]:
```
array(['1'], dtype=object)
```

This column can be ignored because it has only 1 unique value.

[33]:
```
df_train.drop(columns = 'totals.visits', axis = 1, inplace = True)

df_test.drop(columns = 'totals.visits', axis = 1, inplace = True)
```

19. `totals.hits` **INTEGER**

    Total number of hits within the session.

Both `totals.hits` and `totals.pageviews` are similar data. So, `totals.hits` is dropped.

[34]:
```
df_train['totals.hits'] = df_train['totals.hits'].astype(np.int32)

df_test['totals.hits'] = df_test['totals.hits'].astype(np.int32)
```

20. `totals.pageviews` **INTEGER**

    Total number of pageviews within the session.

[35]:
```
# Convert totals_pageviews into a integer data type and fill na value with 0
def clean_pageviews(df):
    df['totals.pageviews'].fillna(0, inplace = True)
    df['totals.pageviews'] = df['totals.pageviews'].astype(np.int32)

# Apply function
```

```
clean_pageviews(df_train)

clean_pageviews(df_test)
```

21. `totals.bounces` **INTEGER**

    Total bounces (for convenience). For a bounced session, the value is 1, otherwise it is null.

[36]:

```
# convert totals_bounces into a integer data type and fill na value with 0
def clean_bounces(df):
    df['totals.bounces'].fillna(0, inplace = True)
    df['totals.bounces'] = df['totals.bounces'].astype(np.int32)

# Apply function
clean_bounces(df_train)

clean_bounces(df_test)
```

22. `totals.newVisits` **INTEGER**

    Total number of new users in session (for convenience). If this is the first visit, this value is 1, otherwise it is null.

[37]:

```
# convert totals_newVisits into a integer data type and fill na value with 0
def clean_newVisits(df):
    df['totals.newVisits'].fillna(0, inplace = True)
    df['totals.newVisits'] = df['totals.newVisits'].astype(np.int32)

# Apply function
clean_newVisits(df_train)

clean_newVisits(df_test)
```

23. `totals.sessionQualityDim` **INTEGER**

    An estimate of how close a particular session was to transacting, ranging from 1 to 100, calculated for each session. A value closer to 1 indicates a low session quality, or far from transacting, while a value closer to 100 indicates a high session quality, or very close to transacting. A value of 0 indicates that Session Quality is not calculated for the selected time range.

[38]:

```
# convert sessionQualityDim into a integer data type and fill na value with 0
def clean_sessionQualityDim(df):
    df['totals.sessionQualityDim'].fillna(0, inplace = True)
    df['totals.sessionQualityDim'] = df['totals.sessionQualityDim'].astype(np.
 ↪int32)

# Apply function
```

```
clean_sessionQualityDim(df_train)

clean_sessionQualityDim(df_test)
```

24. `totals.timeOnSite` **INTEGER**

    Total time of the session expressed in seconds.

[39]:
```
# convert totals_timeOnSite into a integer data type and fill na value with 0
def clean_timeOnSite(df):
    df['totals.timeOnSite'].fillna(0, inplace = True)
    df['totals.timeOnSite'] = df['totals.timeOnSite'].astype(np.int32)

# Apply function
clean_timeOnSite(df_train)

clean_timeOnSite(df_test)
```

25. `totals.transactions` **INTEGER**

    Total number of ecommerce transactions within the session.

[40]:
```
df_train['totals.transactions'].fillna(0, inplace = True)
df_train['totals.transactions'] = df_train['totals.transactions'].astype(np.
 ↪int32)

df_test['totals.transactions'].fillna(0, inplace = True)
df_test['totals.transactions'] = df_test['totals.transactions'].astype(np.int32)
```

26. `totals.transactionRevenue` **FLOAT**

    This field is deprecated. Use “totals.totalTransactionRevenue” instead

[41]:
```
df_train['totals.transactionRevenue'] = df_train['totals.transactionRevenue'].
 ↪astype('float')

df_test['totals.transactionRevenue'] = df_test['totals.transactionRevenue'].
 ↪astype('float')
```

27. `totals.totalTransactionRevenue` **FLOAT**

    Total transaction revenue, expressed as the value passed to Analytics multiplied by 10^6 (e.g., 2.40 would be given as 2400000).

[42]:
```
df_train['totals.totalTransactionRevenue'] = df_train['totals.
 ↪totalTransactionRevenue'].astype('float')

df_test['totals.totalTransactionRevenue'] = df_test['totals.
 ↪totalTransactionRevenue'].astype('float')
```

28. `trafficSource.campaign` **STRING**

    The campaign value. Usually set by the utm_campaign URL parameter.

29. `trafficSource.source` **STRING**

    The campaign value. Usually set by the utm_campaign URL parameter.

30. `trafficSource.medium` **STRING**

    The medium of the traffic source. Could be "organic", "cpc", "referral", or the value of the utm_medium URL parameter.

31. `trafficSource.keyword` **STRING**

    If this was a search results page, this is the keyword entered.

[43]:

```
# https://shop.googlemerchandisestore.com/
# 6qEhsCssdK0z36ri = YouTube Small Sticker Sheet
# https://shop.googlemerchandisestore.com/Google+Redesign/Accessories/
 ↪YouTube+Small+Sticker+Sheet
# 1hZbAqLCbjwfgOH7 = Google
# https://shop.googlemerchandisestore.com/asearch.html?
 ↪vid=20160512512&key=1hZbAqLCbjwfgOH7&keyword=1hZbAqLCbjwfgOH7
import re

googleKeywords = ['google', 'goo', 'gle',('1hZbAqLCbjwfgOH7').lower()]
youtubeKeywords = ['youtube', 'yt', 'yotube', 'yutube', ('6qEhsCssdK0z36ri').
 ↪lower()]
androidKeywords = ['android']
autoMatchingKeywords = ['automatic matching']
userTargetingKeywords = ['user vertical targeting']
remarketingKeywords = ['remarketing/content targeting']
doubleClickAdExchangeKeywords = ['doubleclick']

# Define function to categorize trafficSource_keyword
def categorize_trafficSource_keyword(x):

    if pd.isna(x):
        return
    else:
        x = str(x).lower()

        isGoogle = re.findall(r"(?=("+'|'.join(googleKeywords)+r"))",x)
        isYoutube = re.findall(r"(?=("+'|'.join(youtubeKeywords)+r"))",x)
        isAndroid = re.findall(r"(?=("+'|'.join(androidKeywords)+r"))",x)
        isAutoMatching = re.findall(r"(?=("+'|'.
 ↪join(autoMatchingKeywords)+r"))",x)
        isUserTargeting = re.findall(r"(?=("+'|'.
 ↪join(userTargetingKeywords)+r"))",x)
```

```
        isRemarketing = re.findall(r"(?=("+'|'.
↪join(remarketingKeywords)+r"))",x)
        isDoubleClickAdExchange = re.findall(r"(?=("+'|'.
↪join(doubleClickAdExchangeKeywords)+r"))",x)

        if isGoogle:
            keyword = 'Google'
        elif isYoutube:
            keyword = 'Youtube'
        elif isAndroid:
            keyword = 'Android'
        elif isAutoMatching:
            keyword = '(automatic matching)'
        elif isUserTargeting:
            keyword = '(User vertical targeting)'
        elif isRemarketing:
            keyword = '(Remarketing/Content targeting)'
        elif isDoubleClickAdExchange:
            keyword = 'DoubleClick Ad Exchange'
        else:
            keyword = 'Other'

        return keyword
```

[44]:

```
# Apply Function
df_train['trafficSource.keyword'] = df_train['trafficSource.keyword'].
↪apply(categorize_trafficSource_keyword)

df_test['trafficSource.keyword'] = df_test['trafficSource.keyword'].
↪apply(categorize_trafficSource_keyword)
```

32. `trafficSource_referralPath` **STRING**

    If trafficSource.medium is “referral”, then this is set to the path of the referrer. (The host name of the referrer is in trafficSource.source.)

33. `trafficSource_isTrueDirect` **BOOLEAN**

    True if the source of the session was Direct (meaning the user typed the name of your website URL into the browser or came to your site via a bookmark), This field will also be true if 2 successive but distinct sessions have exactly the same campaign details. Otherwise NULL.

34. `trafficSource_adwordsClickInfo.gclId` **STRING**

    The Google Click ID.

For Google Ads info, create new column `googleAds` and if `adwordsClickInfo.gclId` is not null, fill it with **1** otherwise **0**.

[45]:

```
# Define function to categorize Google Ads info
def categorize_adwordsClickInfo(x):

    if pd.isna(x):
        return '1'
    else:
        return '0'
```

[46]:

```
df_train['googleAds'] = df_train['trafficSource.adwordsClickInfo.gclId'].
 ↪apply(categorize_adwordsClickInfo)

df_test['googleAds'] = df_test['trafficSource.adwordsClickInfo.gclId'].
 ↪apply(categorize_adwordsClickInfo)
```

Remove unused columns for Google Ads info.

[47]:

```
# Drop unused/nan columns
unused_cols = ['trafficSource.adwordsClickInfo.page', 'trafficSource.
 ↪adwordsClickInfo.slot',
               'trafficSource.adwordsClickInfo.gclId', 'trafficSource.
 ↪adwordsClickInfo.adNetworkType',
               'trafficSource.adwordsClickInfo.isVideoAd', 'trafficSource.
 ↪adContent']

df_train.drop(unused_cols, axis = 1, inplace = True)

df_test.drop(unused_cols, axis = 1, inplace = True)
```

35. `customDimensions.index` **INTEGER**

    The index of the custom dimension.

36. `customDimensions.value` **STRING**

    The value of the custom dimension.

[48]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1708337 entries, 0 to 1708336
Data columns (total 39 columns):
 #   Column                             Dtype
---  ------                             -----
 0   channelGrouping                    object
 1   date                               datetime64[ns]
 2   fullVisitorId                      object
 3   visitNumber                        int32
 4   visitStartTime                     int64
 5   device.browser                     object
 6   device.operatingSystem          object
 7   device.isMobile                 bool
 8   device.deviceCategory           object
 9   geoNetwork.continent            object
 10  geoNetwork.subContinent         object
 11  geoNetwork.country              object
 12  geoNetwork.region               object
 13  geoNetwork.metro                object
 14  geoNetwork.city                 object
 15  geoNetwork.networkDomain        object
 16  totals.hits                     int32
 17  totals.pageviews                int32
 18  totals.bounces                  int32
 19  totals.newVisits                int32
 20  totals.sessionQualityDim        int32
 21  totals.timeOnSite               int32
 22  totals.transactions             int32
 23  totals.transactionRevenue       float64
 24  totals.totalTransactionRevenue  float64
 25  trafficSource.campaign          object
 26  trafficSource.source            object
 27  trafficSource.medium            object
 28  trafficSource.keyword           object
 29  trafficSource.referralPath      object
 30  trafficSource.isTrueDirect      bool
 31  customDimensions.index          object
 32  customDimensions.value          object
 33  visit_weekday                   object
 34  visit_day                       object
 35  visit_month                     object
 36  visit_year                      object
 37  visit_hour                      object
 38  googleAds                       object
dtypes: bool(2), datetime64[ns](1), float64(2), int32(8), int64(1), object(25)
memory usage: 433.4+ MB
```

[49]:

```
df_train.to_csv("df_train.csv", index = False)
df_test.to_csv("df_test.csv", index = False)
```

## 3. Exploratory Data Analysis

Notes: > totals.totalTransactionRevenue is preferred to be visualized in this part while totals.transactionRevenue will be used for prediction. A thread in Kaggle to discuss transactionRevenue vs totalTransactionRevenue. The demonstration of all the revenue values are divided by 10^6 for better visualization.

[5]:

```
# Set sns style
sns.set(style="darkgrid")
```

```
# Removing the rainbow colors and use only ONE color
base_color = sns.color_palette()[0]
```

1.1.4 3.1. How many of the visits will result in transaction (buying)

[6]:
```
print('There is only {}% completed transaction in dataset.'\
         .format(round(len(df_train[df_train['totals.totalTransactionRevenue'] >␣
 ↪0]) / df_train.shape[0] * 100, 2)))
```

```
There is only 1.08% completed transaction in dataset.
```

- Only 1% of the visits convert into transaction (buying)

1.1.5 3.2. A deeper look on visit frequency and number of transaction (buying)

[7]:
```
# Aggregate by visitor ID to get frequency of visit
visitFreq_agg = df_train.groupby('fullVisitorId').agg(frequency =␣
 ↪('fullVisitorId', 'count') ,
                                                       transactions = ('totals.
 ↪transactions', 'sum'))

visitFreq_agg['visitRange'] = pd.cut(visitFreq_agg['frequency'], [-1, 1, 2, 3,␣
 ↪6, 10, 20, 40, 80, 500],
                                     labels = ['1', '2', '3', '4-6', '7-10',␣
 ↪'11-20', '21-40', '41-80', '81-500'])

visitFreq_agg = visitFreq_agg.groupby('visitRange').agg(visit_frequency =␣
 ↪('frequency','count'),
                                                         no_of_transactions␣
 ↪=('transactions','sum'))
visitFreq_agg['frequency_%'] = (visitFreq_agg['visit_frequency']/
 ↪sum(visitFreq_agg['visit_frequency'])).map("{:.2%}".format)
visitFreq_agg['transactions_%'] = (visitFreq_agg['no_of_transactions']/
 ↪sum(visitFreq_agg['no_of_transactions'])).map("{:.2%}".format)


visitFreq_agg[['visit_frequency','frequency_%','no_of_transactions','transactions_%']]
```

[7]:
```
            visit_frequency frequency_%  no_of_transactions transactions_%
visitRange
1                   1138049      85.97%                5091         26.34%
2                    115694       8.74%                3853         19.93%
3                     34104       2.58%                2590         13.40%
4-6                   26227       1.98%                3984         20.61%
7-10                   6392       0.48%                1924          9.95%
11-20                  2505       0.19%                1162          6.01%
```

```
21-40                   571       0.04%               549          2.84%
41-80                   135       0.01%                73          0.38%
81-500                   53       0.00%               104          0.54%
```

- Most visitors (85.97%) only visit the website once
- For the visitors that visit the site for 4 - 6 times, they have the highest probability of buy something,
  although they only account for 1.98% of total visitors, they contribute 20.61% of the total transactions (buying)

[8]:

```
purchaseFreq_agg = df_train.groupby('fullVisitorId').agg(frequency = ('totals.
 ↪transactions', 'sum'))
purchaseFreq_agg['purchaseRange'] = pd.cut(purchaseFreq_agg['frequency'], [-1,␣
 ↪0, 1, 2, 3, 6, 15, 40],
                                        labels = ['0', '1', '2', '3', '4-6',␣
 ↪'7-15', '16-40'])
purchaseFreq_agg = purchaseFreq_agg.groupby('purchaseRange').agg('count')
purchaseFreq_agg
```

[8]:

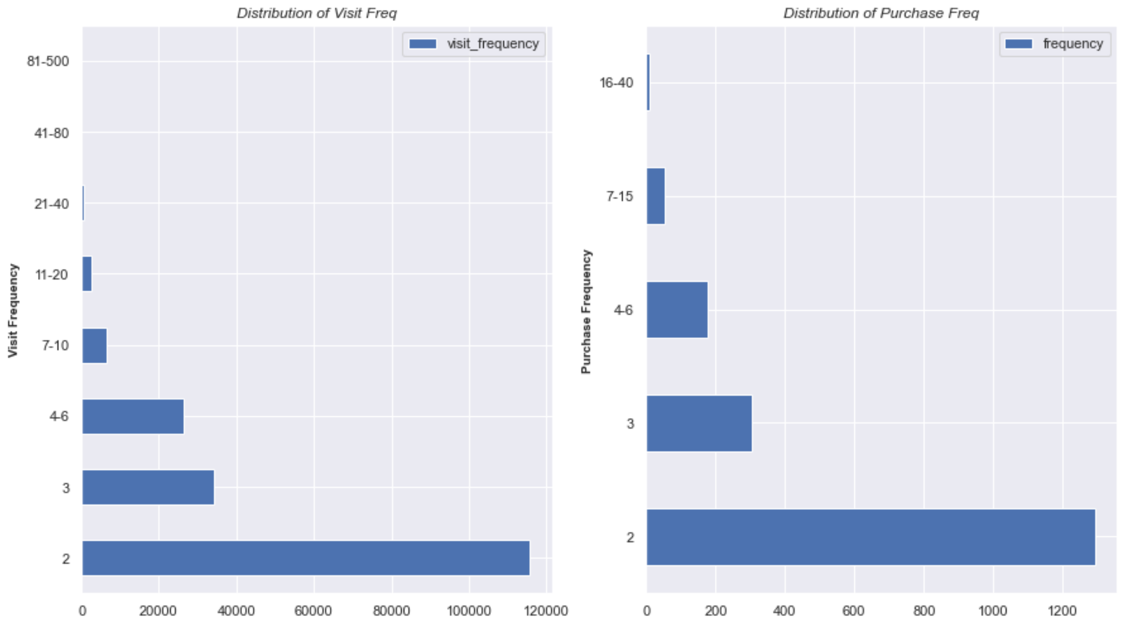
```
               frequency
purchaseRange
0                1307559
1                  14336
2                   1293
3                    304
4-6                  177
7-15                  52
16-40                  9
```

- For the visitor that buying thing most of them only buy one thing

Visualize the visit frequency and purchase frequency (after removing visit freq : 1 and purchaseRange : 0 & 1)

[9]:

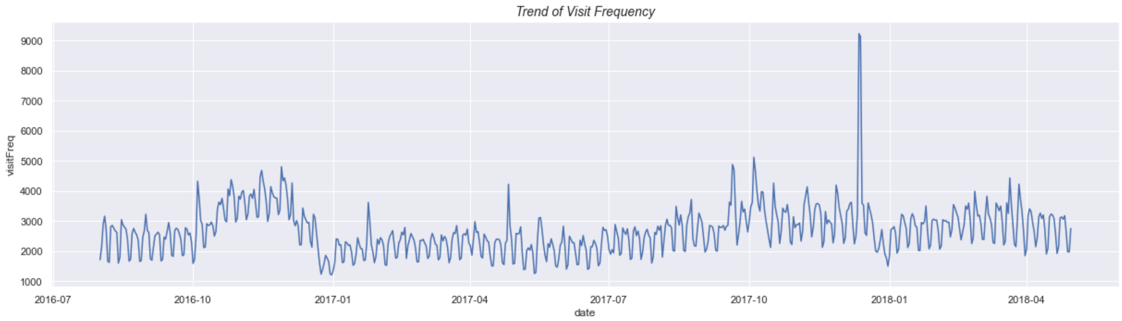
```
fig,axes = plt.subplots(1,2,figsize = (14.70, 8.27))
visitFreq_agg[1:][['visit_frequency']].plot.barh(ax = axes[0])

axes[0].set_title('Distribution of Visit Freq', fontsize = 12, style = 'italic')
axes[0].set_ylabel('Visit Frequency', fontsize = 10, weight = 'bold');

purchaseFreq_agg[2:].plot.barh(ax = axes[1])

axes[1].set_title('Distribution of Purchase Freq', fontsize = 12, style =␣
 ↪'italic')
axes[1].set_ylabel('Purchase Frequency', fontsize = 10, weight = 'bold');
```



1.1.6 3.3. Do the number of visits and purchase (transaction) effected by seasonality?

[10]:
```
GData = df_train.groupby('date').agg(visitFreq = ('fullVisitorId', 'count'),
                                     newVisit = ('totals.newVisits', 'sum'),
                                     transaction = ('totals.transactions',␣
 ↪'sum'))

GData = GData.reset_index()
```

[11]:
```
## Graph for visit frequency

fig, ax =  plt.subplots(1, 1, figsize = (20.70, 5.27))
sns.lineplot(x="date", y="visitFreq", data=GData)

ax.set_title('Trend of Visit Frequency', fontsize = 14, style = 'italic');
```



[12]:

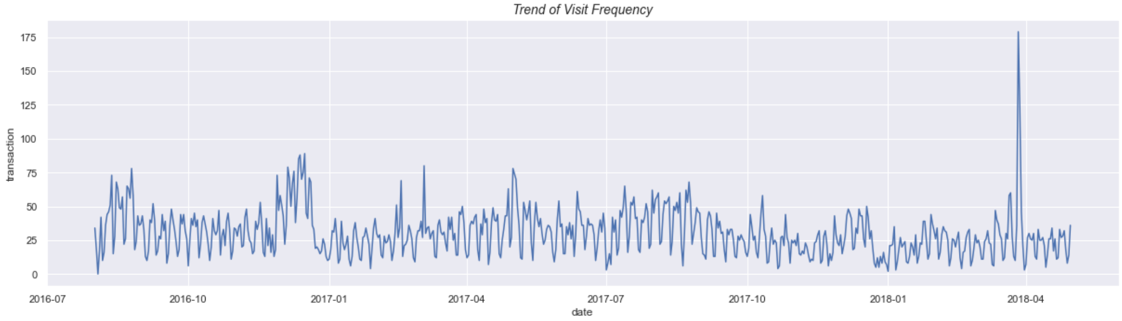
```
GData.nlargest(10,['visitFreq'])
```

[12]:

```
           date  visitFreq  newVisit  transaction
498  2017-12-12       9234      8159           30
499  2017-12-13       9131      7400           48
429  2017-10-04       5122      3822           35
415  2017-09-20       4880      3846           33
119  2016-11-28       4807      3834           73
416  2017-09-21       4715      3637           33
106  2016-11-15       4685      3837           33
430  2017-10-05       4679      3487           25
105  2016-11-14       4466      3558           39
121  2016-11-30       4435      3498           58
```

[13]:

```
## Graph for sum of transaction (buying)

fig, ax =  plt.subplots(1, 1, figsize = (20.70, 5.27))
sns.lineplot(x="date", y="transaction", data=GData)

ax.set_title('Trend of Visit Frequency', fontsize = 14, style = 'italic');
```



[14]:

```
GData.nlargest(10,['transaction'])
```

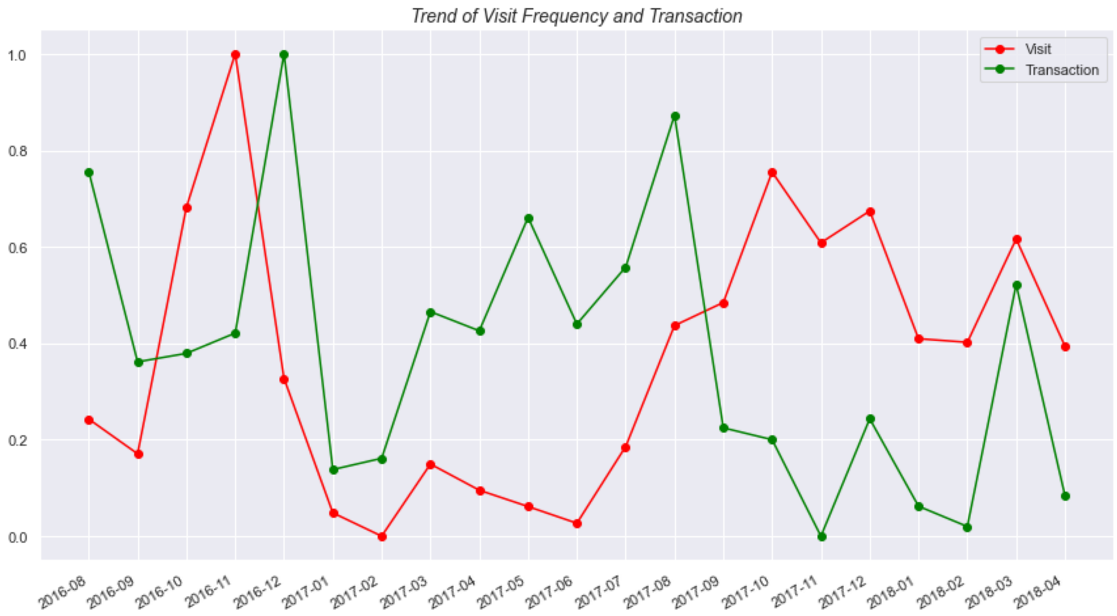
[14]:

```
           date  visitFreq  newVisit  transaction
603  2018-03-27       4227      3057          179
604  2018-03-28       3724      2576          123
137  2016-12-16       2956      2106           89
134  2016-12-13       3166      2256           88
133  2016-12-12       3433      2464           85
215  2017-03-04       1753      1396           80
126  2016-12-05       4265      3217           79
24   2016-08-25       2539      1921           78
273  2017-05-01       2588      1906           78
130  2016-12-09       2830      1967           76
```

1.1.7 Noted something weird? Let compared them in s single graph

[15]:

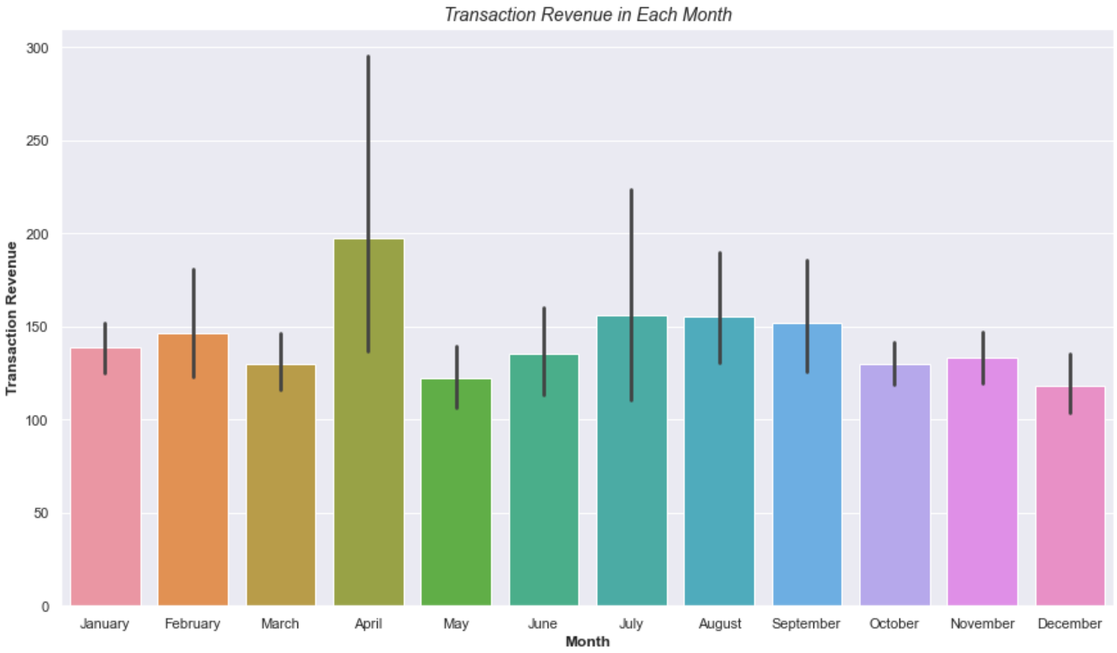
```
df_train['yearMonth'] = df_train['date'].dt.to_period('M')

monthly_visit = df_train.groupby('yearMonth').agg(visitFreq = ('fullVisitorId',␣
 ↪'count'),
                                                  transaction = ('totals.
 ↪transactions', 'sum'))

monthly_visit.reset_index(inplace = True)
monthly_visit['yearMonth'] = monthly_visit['yearMonth'].astype(str)

min_max_scaler = MinMaxScaler()
monthly_visit['visitFreq'] = min_max_scaler.
 ↪fit_transform(monthly_visit[['visitFreq']])
monthly_visit['transaction'] = min_max_scaler.
 ↪fit_transform(monthly_visit[['transaction']])

fig, ax =  plt.subplots(1, 1, figsize = (14.70, 8.27))

ax.plot_date(monthly_visit['yearMonth'], monthly_visit["visitFreq"],␣
 ↪color="red", label="Visit", linestyle="-")
ax.plot_date(monthly_visit['yearMonth'], monthly_visit["transaction"],␣
 ↪color="green", label="Transaction", linestyle="-")
ax.legend()
ax.set_title('Trend of Visit Frequency and Transaction', fontsize = 14, style =␣
 ↪'italic');
plt.gcf().autofmt_xdate();
```



- Bingo, the period with most visit != the period with highest selling

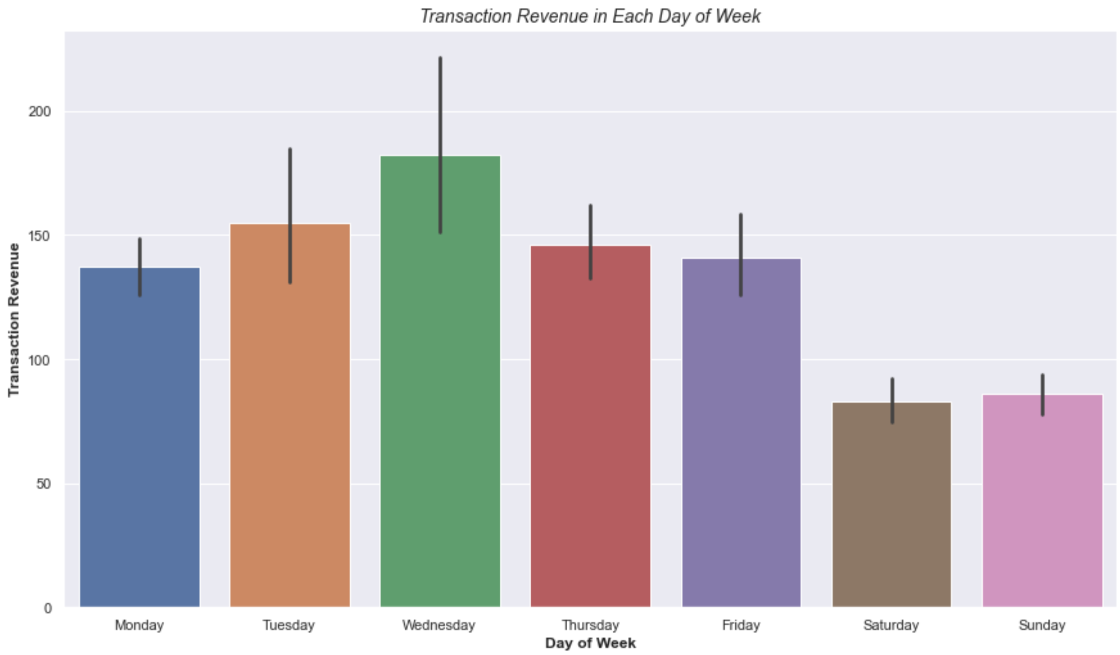
1.1.8 3. When Gstore gain most of the profit

[16]:
```
# Define order of month for better visualization
month_order = ['January', 'February', 'March', 'April', 'May', 'June',
               'July', 'August', 'September', 'October', 'November', 'December']

plt.figure(figsize=(14.70, 8.27))

# Set sns style
sns.set(style="darkgrid")

# Removing the rainbow colors and use only ONE color
base_color = sns.color_palette()[0]

sns.barplot(data = df_train,
            x = 'visit_month',
            y = (df_train['totals.totalTransactionRevenue'] / 10**6),
            order = month_order)
plt.title("Transaction Revenue in Each Month", fontsize = 14, style = 'italic')
plt.xlabel("Month", fontsize = 12, weight = 'bold')
plt.ylabel("Transaction Revenue", fontsize = 12, weight = 'bold');
```



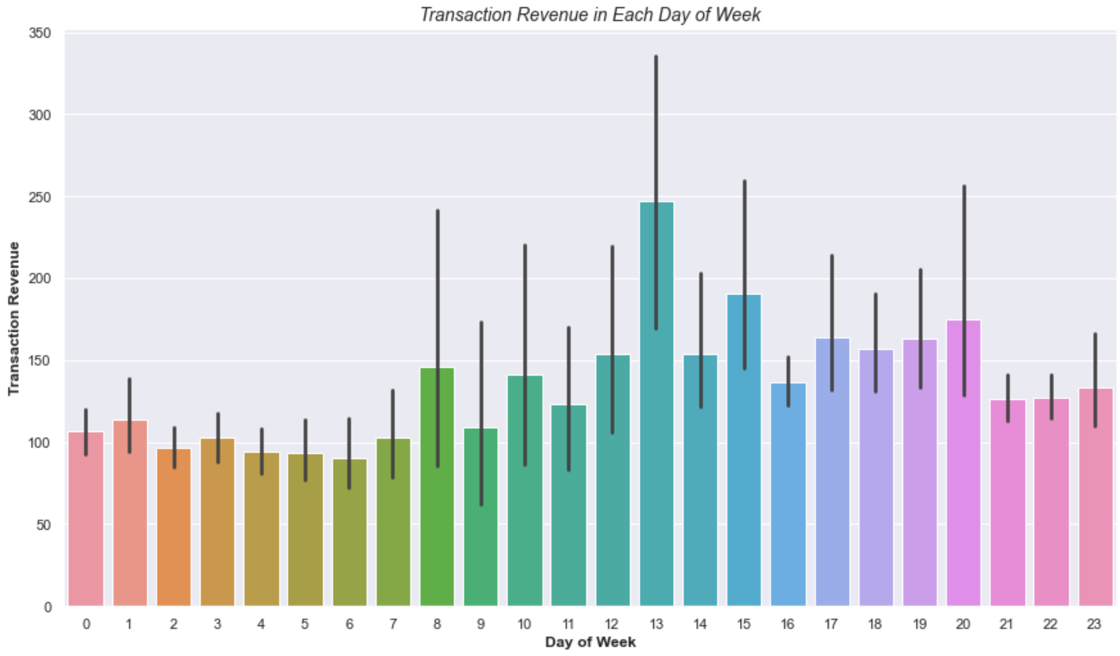
- Revenue peak on April

[17]:
```
# Define order of day for better visualization
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
↪'Saturday', 'Sunday']

plt.figure(figsize=(14.70, 8.27))
sns.barplot(data = df_train,
            x = 'visit_weekday',
            y = (df_train['totals.totalTransactionRevenue'] / 10**6),
            order = day_order)
plt.title("Transaction Revenue in Each Day of Week", fontsize = 14, style =␣
↪'italic')
plt.xlabel("Day of Week", fontsize = 12, weight = 'bold')
plt.ylabel("Transaction Revenue", fontsize = 12, weight = 'bold');
```



- Sales concentrated on Weekday? Interesting

[18]:
```
plt.figure(figsize=(14.70, 8.27))
sns.barplot(data = df_train,
            x = 'visit_hour',
            y = (df_train['totals.totalTransactionRevenue'] / 10**6))
plt.title("Transaction Revenue in Each Day of Week", fontsize = 14, style =␣
↪'italic')
plt.xlabel("Day of Week", fontsize = 12, weight = 'bold')
plt.ylabel("Transaction Revenue", fontsize = 12, weight = 'bold');
```

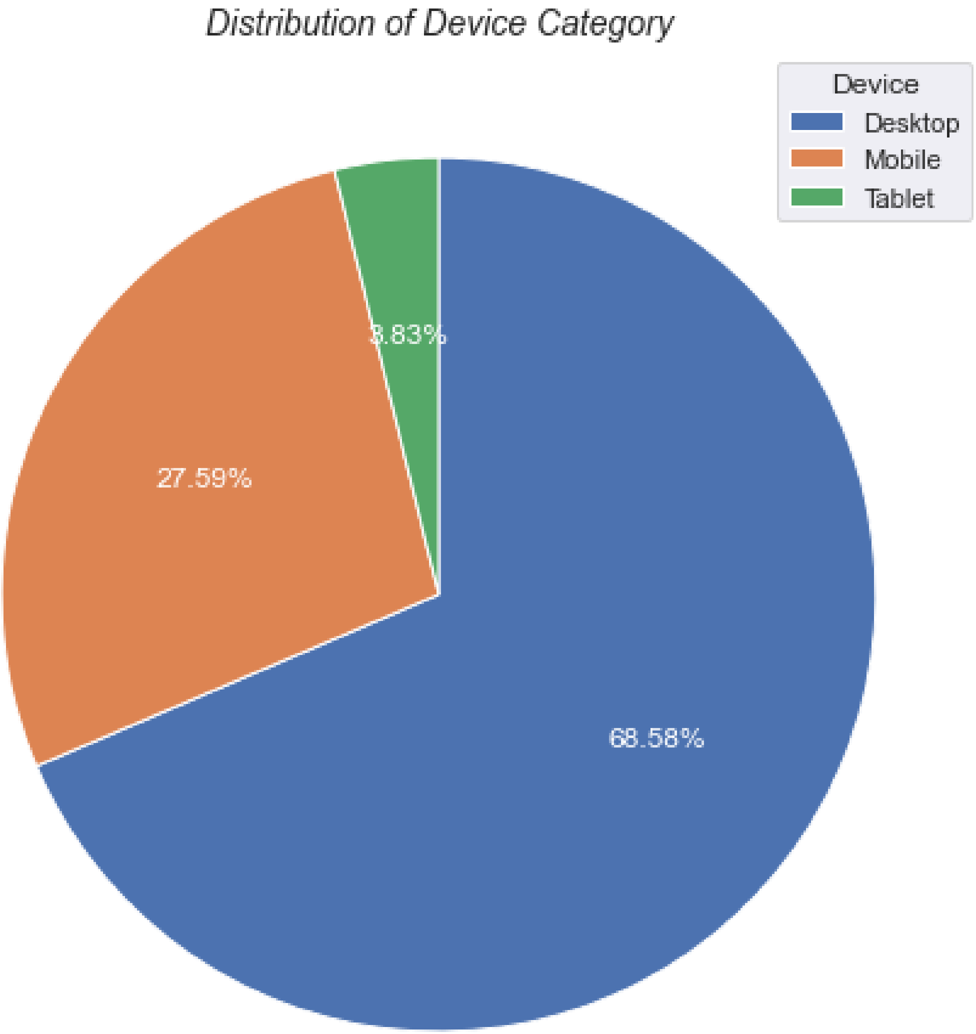


- Most revenue gain by visit in 13:00 (1:00pm), ok some story appeared

1.1.9 3.4. What device visitor use to visit Gstore

[19]:

```
device = df_train['device.deviceCategory'].value_counts()

plt.figure(figsize=[14.70, 8.27])
plt.pie(device, labels = device.index, startangle = 90, autopct= '%1.2f%%',␣
 ↪counterclock = False, textprops={'color':'white'});
plt.legend(['Desktop', 'Mobile', 'Tablet'],
           title='Device',
           loc='upper right')
plt.title('Distribution of Device Category', fontsize = 14,  style ='italic');
```



1.1.10 3.5 The location of the buyer

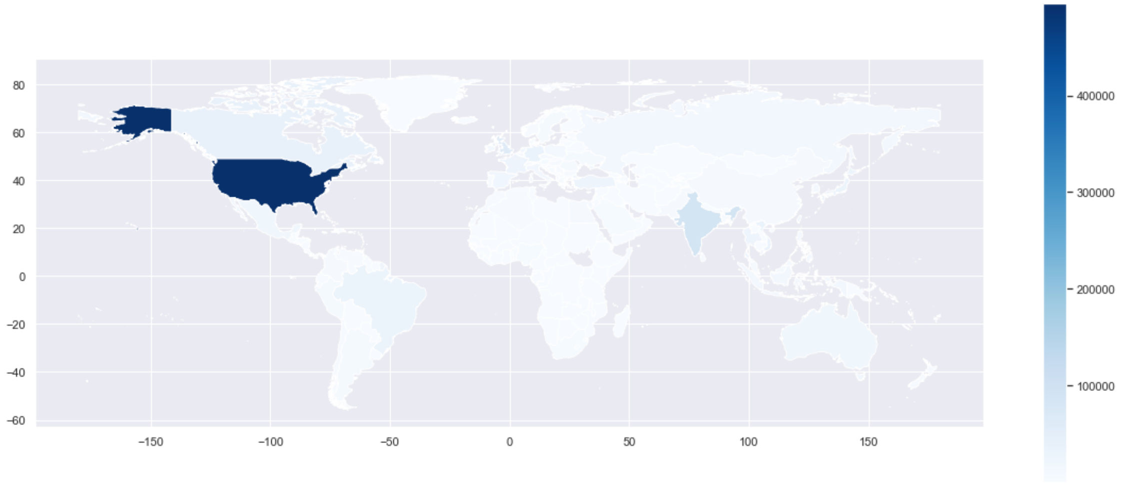
Get a Visualization on the location (country)

[22]:
```
# https://www.naturalearthdata.com/downloads/50m-cultural-vectors/
 ↪50m-admin-0-countries-2/
# Import geolocation shape file

import geopandas as gpd
import pycountry

world = gpd.read_file('map/ne_50m_admin_0_countries.shp')
```

```
#world

country_visitor = df_train.groupby(['geoNetwork.country', 'fullVisitorId']).
 ↪size().reset_index()
country_visitor = country_visitor.groupby(['geoNetwork.country']).agg(count =␣
 ↪('fullVisitorId', 'count')).reset_index()
country_visitor.sort_values(by = 'count', ascending = False)

input_countries = country_visitor['geoNetwork.country']
countries = {}
for country in pycountry.countries:
    countries[country.name] = country.alpha_3

codes = [countries.get(country, 'Unknown code') for country in input_countries]
country_visitor['countryCode'] = codes


# Manual fix unknowmn country code
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Antigua &␣
 ↪Barbuda', 'countryCode'] = 'ATG'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Bolivia',␣
 ↪'countryCode'] = 'BOL'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Bosnia &␣
 ↪Herzegovina', 'countryCode'] = 'BIH'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'British Virgin␣
 ↪Islands', 'countryCode'] = 'VGB'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Brunei',␣
 ↪'countryCode'] = 'BRU'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Cape␣
 ↪Verde']['countryCode'] == 'CPV'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Caribbean␣
 ↪Netherlands', 'countryCode'] = 'BQ'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Congo -␣
 ↪Brazzaville', 'countryCode'] = 'COG'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Congo -␣
 ↪Kinshasa', 'countryCode'] = 'COD'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Côte d’Ivoire',␣
 ↪'countryCode'] = 'CIV'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Iran',␣
 ↪'countryCode'] = 'IRN'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Kosovo',␣
 ↪'countryCode'] = 'KOS'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Laos',␣
 ↪'countryCode'] = 'LAO'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Macau',␣
 ↪'countryCode'] = 'MAC'
```

```
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Macedonia␣
 ↪(FYROM)', 'countryCode'] = 'MKD'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Micronesia',␣
 ↪'countryCode'] = 'FSM'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Palestine',␣
 ↪'countryCode'] = 'PSE'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Russia',␣
 ↪'countryCode'] = 'RUS'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Sint Maarten',␣
 ↪'countryCode'] = 'SXM'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'South Korea',␣
 ↪'countryCode'] = 'KOR'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'St. Barthélemy',␣
 ↪'countryCode'] = 'BLM'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'St. Helena',␣
 ↪'countryCode'] = 'SHN'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'St. Kitts &␣
 ↪Nevis', 'countryCode'] = 'KNA'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'St. Lucia',␣
 ↪'countryCode'] = 'LCA'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'St. Martin',␣
 ↪'countryCode'] = 'MAF'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'St. Pierre &␣
 ↪Miquelon', 'countryCode'] = 'SPM'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'St. Vincent &␣
 ↪Grenadines', 'countryCode'] = 'VCT'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Swaziland',␣
 ↪'countryCode'] = 'SWZ'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Syria',␣
 ↪'countryCode'] = 'SYR'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'São Tomé &␣
 ↪Príncipe', 'countryCode'] = 'STP'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Taiwan',␣
 ↪'countryCode'] = 'TWN'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Tanzania',␣
 ↪'countryCode'] = 'TZA'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Trinidad &␣
 ↪Tobago', 'countryCode'] = 'TTO'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Turks & Caicos␣
 ↪Islands', 'countryCode'] = 'TCA'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'U.S. Virgin␣
 ↪Islands', 'countryCode'] = 'VIR'
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Venezuela',␣
 ↪'countryCode'] = 'VEN'
```

```
country_visitor.loc[country_visitor['geoNetwork.country'] == 'Vietnam',␣
↪'countryCode'] = 'VNM'

merged = world.merge(country_visitor, left_on='ADM0_A3', right_on='countryCode')

# Plot using geopandas
colors = 9
cmap = 'Blues'
figsize=(20, 8.27)
merged.plot(column='count', cmap=cmap, figsize=figsize, k=colors, legend=True);
```



Ok into detail figure

[23]:
```
country = df_train.groupby('geoNetwork.country').agg(visitFreq =␣
↪('fullVisitorId', 'count'),
                                                        no_of_transactions␣
↪=('totals.transactions','sum'))
country['visit_Freq%'] = (country['visitFreq']/sum(country['visitFreq'])).
↪map("{:.2%}".format)
country['trans_Freq%'] = (country['no_of_transactions']/
↪sum(country['no_of_transactions'])).map("{:.2%}".format)

country =␣
↪country[['visitFreq','visit_Freq%','no_of_transactions','trans_Freq%']]


country.sort_values(by=['visitFreq'],ascending = False).head(10)
```

[23]:
```
                     visitFreq visit_Freq%  no_of_transactions trans_Freq%
geoNetwork.country
```

```
United States       717217     42.05%                18349     94.96%
India               105317      6.17%                   18      0.09%
United Kingdom       73341      4.30%                   28      0.14%
Canada               51057      2.99%                  327      1.69%
Germany              38516      2.26%                   12      0.06%
Japan                36637      2.15%                   23      0.12%
Brazil               35432      2.08%                   21      0.11%
Vietnam              34869      2.04%                    0      0.00%
France               32289      1.89%                   16      0.08%
Thailand             29859      1.75%                   10      0.05%
```

- Most of the visits comes from united state, and nearly 95% of the transaction (buying) is come from United State

Now more question, which part of the United State?

[24]:
```
city = df_train.groupby('geoNetwork.city').agg(visitFreq = ('fullVisitorId',␣
 ↪'count'),
                                                no_of_transactions =('totals.
 ↪transactions','sum'))
city['visit_Freq%'] = (city['visitFreq']/sum(country['visitFreq'])).map("{:.
 ↪2%}".format)
city['trans_Freq%'] = (city['no_of_transactions']/
 ↪sum(country['no_of_transactions'])).map("{:.2%}".format)

city = city[['visitFreq','visit_Freq%','no_of_transactions','trans_Freq%']]


city.sort_values(by=['visitFreq'],ascending = False).head(10)

### since geoNetwork.city have more NA compared to geoNetwork.country we divide␣
 ↪the total visitfreq and transaction of a city
### with the sum of geoNetwork.country figure to get the idea of percentage.
```

[24]:
```
                 visitFreq visit_Freq%  no_of_transactions trans_Freq%
geoNetwork.city
Mountain View        74110       4.34%                2156      11.16%
New York             49460       2.90%                2527      13.08%
San Francisco        36960       2.17%                1180       6.11%
Sunnyvale            27923       1.64%                 883       4.57%
London               23622       1.38%                  19       0.10%
San Jose             20141       1.18%                 393       2.03%
Los Angeles          17038       1.00%                 491       2.54%
Chicago              15143       0.89%                 697       3.61%
Bangkok              12468       0.73%                   5       0.03%
Bengaluru            11428       0.67%                   3       0.02%
```

- Interesting info: Mountain View,Sunnyvale and San Jose is cities loacted inside silicon valley, together they account for 7.5% of the total visit and 18% of the total transaction (buying) of Gstore

1.2 Info we get from exploratory analysis

- Most of the visitors (99%) only view and not buy
- Main customer of the sites come from United State, and the biggest group comes from silicon valley (where the company of google located)
- Most buying occured on weekdays and visit time around 1:00pm, possible is a lunch (rest time) of office
- The April and Oct-Dec is some hot season the sites been visited and buying occured (maybe due to gifts of easter day (April) and Chrismas (December)?)

1.2.1 And last we have a view on distribution of figures (transaction revenue) we going to predict

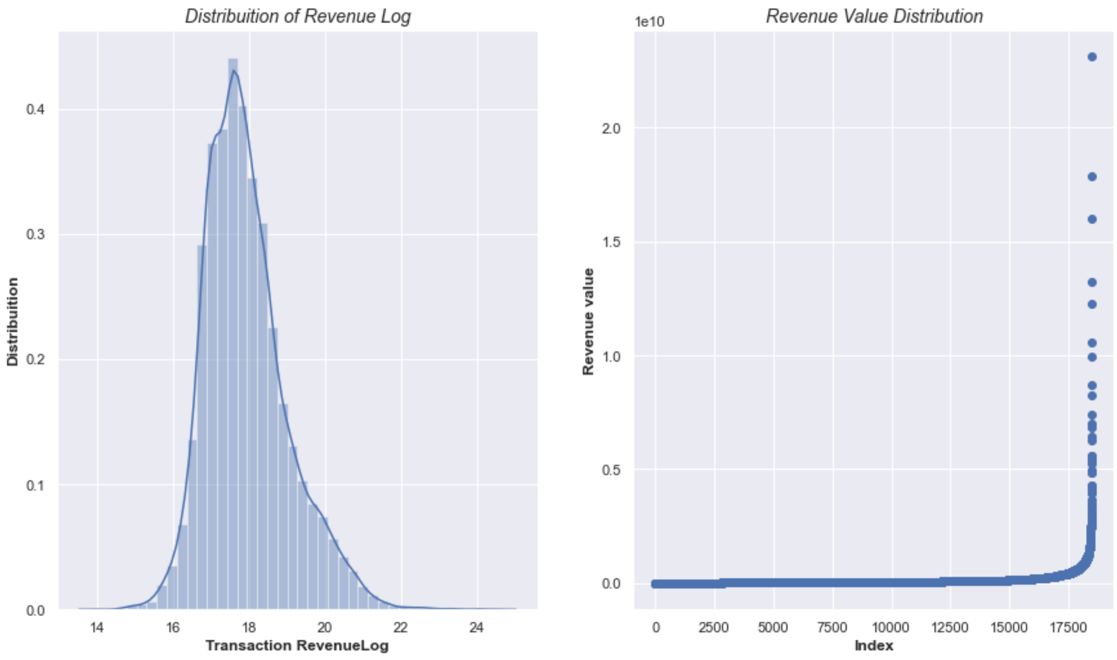
- we are going to predict target, target = log(1+sum(per user transactions))
- we will look at the pdf of target to understand more about its distribution.

[25]:

```
# Printing some statistics of our data
print("Transaction Revenue Min Value: ",
      df_train[df_train['totals.totalTransactionRevenue'] > 0]["totals.
 ↪totalTransactionRevenue"].min() / 10**6) # printing the min value
print("Transaction Revenue Mean Value: ",
      df_train[df_train['totals.totalTransactionRevenue'] > 0]["totals.
 ↪totalTransactionRevenue"].mean() / 10**6) # mean value
print("Transaction Revenue Median Value: ",
      df_train[df_train['totals.totalTransactionRevenue'] > 0]["totals.
 ↪totalTransactionRevenue"].median() / 10**6) # median value
print("Transaction Revenue Max Value: ",
      df_train[df_train['totals.totalTransactionRevenue'] > 0]["totals.
 ↪totalTransactionRevenue"].max() / 10**6) # the max value

# seting the figure size of our plots
plt.figure(figsize=(14.70, 8.27))

# Subplot allow us to plot more than one
# in this case, will be create a subplot grid of 2 x 1
plt.subplot(1,2,1)
# seting the distribuition of our data and normalizing using np.log on values␣
 ↪highest than 0 and +
# also, we will set the number of bins and if we want or not kde on our␣
 ↪histogram
ax = sns.distplot(np.log(df_train[df_train['totals.totalTransactionRevenue'] >␣
 ↪0]["totals.totalTransactionRevenue"] + 0.01), bins=40, kde=True)
ax.set_xlabel('Transaction RevenueLog', fontsize = 12, weight = 'bold') #seting␣
 ↪the xlabel and size of font
```

```
ax.set_ylabel('Distribuition', fontsize = 12, weight = 'bold') #seting the ylabel and size of font
ax.set_title("Distribuition of Revenue Log", fontsize = 14, style = 'italic') #seting the title and size of font

# setting the second plot of our grid of graphs
plt.subplot(1,2,2)
# ordering the total of users and seting the values of transactions to understanding
plt.scatter(range(df_train.shape[0]), np.sort(df_train['totals.transactionRevenue'].values))
plt.xlabel('Index', fontsize = 12, weight = 'bold') # xlabel and size of words
plt.ylabel('Revenue value', fontsize = 12, weight = 'bold') # ylabel and size of words
plt.title("Revenue Value Distribution", fontsize = 14, style = 'italic'); # Setting Title and fonts
```

```
Transaction Revenue Min Value:  1.2
Transaction Revenue Mean Value:  142.81666954736957
Transaction Revenue Median Value:  52.79
Transaction Revenue Max Value:  47082.06
```

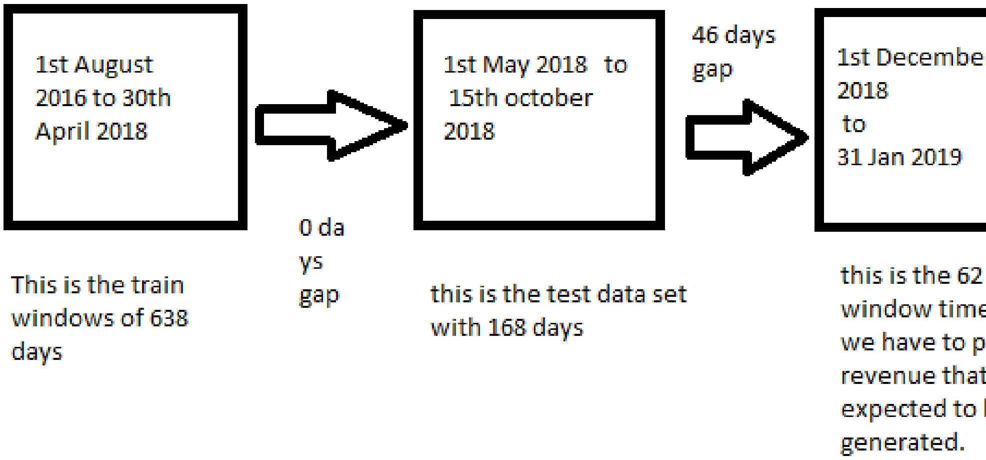


## 4. Data Modeling

1.2.2 4.1 Create Train Set

- Train Dataset (01/08/2016 to 30/04/2018) = 638 days
- Test Dataset (01/05/2018 to 15/10/2018) = 168 days
- Prediction (01/12/2018 to 31/01/2019) = 62 days

[8]:
```
# Display Image in code block
#Image('Data_Modeling_Concept.png')
```


1st August 2016 to 30th April 2018
0 da ys gap
1st May 2018 to 15th october 2018
46 days gap
1st Decembe 2018 to 31 Jan 2019
This is the train windows of 638 days
this is the test data set with 168 days
this is the 62 window time we have to p revenue that expected to l generated.

Display Image in markdown block:

Our objective is to **predict revenue generated by the users in test dataset for a future time period** which is the 3rd window in pc above. As we have 168 days (2018/05/01-2018/10/15) of sessions for customers in the test, 62 days (2018/12/01-2019/01/31) of target calculation period and 46 days (16/10/2018-30/11/2018) of gap between above two windows, it's absolutely clear to construct train data by analogy.

I took 4 non-overlapping windows of 168 days, calculated features for users in each period and calculated target for each user on each corresponding 62-day window. Then those 4 dataframes were combined in one train set.

The idea is first we need to predict whether the user will come to store or not after the "cooling period" of 46 days(or in test period). so for this we will use classification model. If the user visits the store again then we will predict the revenue of that user by using regression model with user data (features).

Data set-1: * train data = 08 Aug 2016 to 15 Jan 2017 (168 days) * test data = 02 Mar 2017 to 03 May 2017 (62 days)

Data set-2: * train data = 16 Jan 2017 to 2 July 2017 (168 days) * test data = 17 Aug 2017 to 18 Oct 2017 (62 days)

Data set-3: * train data = 03 July 2017 to 17 Dec 2017 (168 days) * test data = 01 Feb 2018 to 04 Apr 2018 (62 days)

Data set-4: * train data = 18 Dec 2017 to 04 Jun 2018 (168 days) * test data = 20 July 2018 to 20 Sep 2018 (62 days)

[42]:

```
print("Train Dataset")
print("Start Date: {}".format(min(df_train['date'])))
print("End Date: {}".format(max(df_train['date'])))
print("Time Frame: {}\n".format(max(df_train['date']) - min(df_train['date'])))

print("Test Dataset")
print("Start Date: {}".format(min(df_test['date'])))
print("End Date: {}".format(max(df_test['date'])))
print("Time Frame: {}\n".format(max(df_test['date']) - min(df_test['date'])))

#print("Target Dataset")
#print("Start Date: {}".format(date(2018, 12, 1)))
#print("End Date: {}".format(date(2019, 1, 31)))
#print("Time Frame: {}\n".format(date(2019, 1, 31) - date(2018, 12, 1)))

#print("Gap between test and target: {}".format(date(2018, 12, 1) -␣
 ↪max(df_test['date'])))
```

```
Train Dataset
Start Date: 2016-08-01 00:00:00
End Date: 2018-04-30 00:00:00
Time Frame: 637 days 00:00:00

Test Dataset
Start Date: 2018-05-01 00:00:00
End Date: 2018-10-15 00:00:00
Time Frame: 167 days 00:00:00
```

[ ]:

```
#df1.head(1000).groupby('fullVisitorId').agg(
#    timeOnSite_sum = ('totals.timeOnSite', 'sum'),
#    timeOnSite_count = ('totals.timeOnSite', 'count'),
#    timeOnSite_max = ('totals.timeOnSite', 'max'),
#    timeOnSite_min = ('totals.timeOnSite', 'min')
#)
```

[86]:

```
df_train.isna().sum()
```

[86]:

```
channelGrouping                     0
date                                0
fullVisitorId                       0
visitNumber                         0
visitStartTime                      0
device.browser                      0
device.operatingSystem          11815
device.isMobile                     0
device.deviceCategory               0
```

```
geoNetwork.continent                2517
geoNetwork.subContinent             2517
geoNetwork.country                  2517
geoNetwork.region                 982733
geoNetwork.metro                 1319855
geoNetwork.city                   998826
geoNetwork.networkDomain          768845
totals.hits                            0
totals.pageviews                       0
totals.bounces                         0
totals.newVisits                       0
totals.sessionQualityDim               0
totals.timeOnSite                      0
totals.transactions                    0
totals.transactionRevenue        1689823
totals.totalTransactionRevenue   1689823
trafficSource.campaign           1604526
trafficSource.source                  70
trafficSource.medium              566091
trafficSource.keyword            1621713
trafficSource.referralPath       1280366
trafficSource.isTrueDirect             0
customDimensions.index            333235
customDimensions.value            333235
visit_weekday                          0
visit_day                              0
visit_month                            0
visit_year                             0
visit_hour                             0
googleAds                              0
dtype: int64
```

[89]:

```
#https://stackoverflow.com/questions/19078325/naming-returned-columns-in-pandas-aggregate-function
#https://pandas.pydata.org/pandas-docs/version/0.25.0/user_guide/groupby.html
#https://stackoverflow.com/questions/14529838/apply-multiple-functions-to-multiple-groupby-columns

# source code:- https://www.kaggle.com/kostoglot/winning-solution
from datetime import datetime, timedelta

# this function is to create dataframe of fixed time-interval of 168 days
# so we will create 4 windows of non overlapping dataframe
# we have test data of 168 days and then a gap of 46 days and then we have the prediction month from 1 dec 2018 to 31 jan 2019 that is of 62 days
# so we use our train data to create 4 non overlapping windows of 168 days each␣
 ↪and predict target for this window after a gap of 46 days for 62 day time␣
 ↪period

def Timeframewithfeatures(tr_df, k):
    tf = tr_df.loc[(tr_df['date'] >= min(tr_df['date']) + timedelta(days =␣
 ↪168*(k-1)))
               & (tr_df['date'] < min(tr_df['date']) + timedelta(days =␣
 ↪168*k))] # here we are taking dataframe of 168 days depending on k value

    tf_fvid = set(tr_df.loc[(tr_df['date'] >= min(tr_df['date']) + ␣
 ↪timedelta(days = 168*k + 46))
                             & (tr_df['date'] < min(tr_df['date']) + ␣
 ↪timedelta(days = 168*k + 46 + 62))]['fullVisitorId']) # here we are getting␣
 ↪the full visitor ids of the people who have shopped in the particular window␣
 ↪have returned to the store after 46 days gap in 62 days interval

    tf_returned =  tr_df[tr_df['fullVisitorId'].isin(tf_fvid)] #this is the␣
 ↪dataframe of the users in the window who have visited again afer the gap

    tf_tst = tr_df[tr_df['fullVisitorId'].
 ↪isin(set(tf_returned['fullVisitorId']))
             & (tr_df['date'] >= min(tr_df['date']) + timedelta(days=168*k +␣
 ↪46))
             & (tr_df['date'] < min(tr_df['date']) + timedelta(days=168*k + 46␣
 ↪+ 62))]    #making sure that this is in the same time window

    tf_target = tf_tst.groupby('fullVisitorId')['totals.transactionRevenue'].
 ↪sum().apply(lambda x : np.log1p(x)).reset_index() #we are calculating target␣
 ↪in the 62 day range after a gap of 46 days for each window
    tf_target['ret'] = 1 #creating new column with ret = 1 here giving a value␣
 ↪of 1 for returning customers or users.
    tf_target.rename(columns={'totals.transactionRevenue': 'target'},␣
 ↪inplace=True)

    tf_nonret = pd.DataFrame() #similarly getting datframe for non returning␣
 ↪customers
    tf_nonret['fullVisitorId'] = list(set(tf['fullVisitorId']) - tf_fvid)   ␣
 ↪#getting df for no return customer
    tf_nonret['target'] = 0 #if not returning the target is zero
    tf_nonret['ret'] = 0 #for non returning customer we create a column with␣
 ↪zero as value which indicates the non returning of the customer

    tf_target = pd.concat([tf_target, tf_nonret], axis=0).reset_index(drop=True)
    # len(set(tf['fullVisitorId'])), len(set(tf_target['fullVisitorId']))
    tf_maxdate = max(tf['date'])
    tf_mindate = min(tf['date'])

    # Generating new features
    tf = tf.groupby('fullVisitorId').agg({
        'channelGrouping': [('channelGrouping', 'max')],
        'date': [('first_ses_from_the_period_start', lambda x: x.dropna().min()␣
 ↪- tf_mindate), #time of the first session start from the window strat
             ('last_ses_from_the_period_end', lambda x: tf_maxdate - x.
 ↪dropna().max()), # gives last session from window end of user
             ('interval_dates', lambda x: x.max() - x.min())], #time␣
 ↪between first and last session of the user
        'visitStartTime': [('visitStartTime_counts', 'count')], #number of␣
 ↪times a visitor has visited the site
        'visitNumber': [('visitNumber_max', 'max')],
        'device.browser':  [('browser', 'max')],
        'device.operatingSystem': [('operatingSystem', lambda x: x.dropna().
 ↪max())],
        'device.isMobile': [('isMobile', 'max')],
        'device.deviceCategory': [('deviceCategory', 'max')],
        'geoNetwork.continent': [('continent', lambda x: x.dropna().max())],␣
 ↪#which contitnent category occurs max time for that user
        'geoNetwork.subContinent': [('subContinent', lambda x: x.dropna().
 ↪max())],
        'geoNetwork.country': [('country', lambda x: x.dropna().max())],
        'geoNetwork.region': [('region', lambda x: x.dropna().max())],
        'geoNetwork.metro': [('metro', lambda x: x.dropna().max())],
        'geoNetwork.city': [('city', lambda x: x.dropna().max())],
        'geoNetwork.networkDomain': [('networkDomain', lambda x: x.dropna().
 ↪max())],# for categorical column max function will take the max no. og␣
 ↪occurance of that particaular category when grouped by full
          #`totals.visits` is excluded because same accross dataset
        'totals.bounces': [
            #('bounces_count', lambda x: x.dropna().count()),
            #('bounces_sum', lambda x: x.dropna().sum()),
            ('bounces_mean', 'mean')],
        'totals.newVisits': [('newVisits', 'min')], # if user visit more than␣
 ↪one time, get 0, else 1
        'totals.hits': [('hits_sum', 'sum'),
                        ('hits_min', 'min'),
                        ('hits_max', 'max'),
                        ('hits_mean', 'mean')],
        'totals.pageviews': [('pageviews_sum', 'sum'), #getting toal number of␣
 ↪pageviews for each visitor
                              ('pageviews_min', 'min'), #getting min number of␣
 ↪pageviews for each visitor
```

```
                             ('pageviews_max', 'max'), #getting max number of␣
↪pageviews for each visitor
                             ('pageviews_mean', 'mean')],

        'totals.sessionQualityDim': [('sessionQualityDimMin', 'min'),
                                     ('sessionQualityDimMax', 'max'),
                                     ('sessionQualityDimMean', 'mean'),
                                     ('sessionQualityDimSum', 'sum')],
        'totals.timeOnSite': [('timeOnSite_sum', 'sum'),#total time spent on␣
↪site
                              ('timeOnSite_min', 'mean'), #min time spent on␣
↪site
                              ('timeOnSite_max', 'max'), # max time spent on␣
↪site
                              ('timeOnSite_mean', 'mean')], #mean time spent on␣
↪site
        'totals.transactions' : [('transactions', lambda x:x.dropna().sum())],
        'totals.transactionRevenue':  [('transactionRevenue_sum',  lambda x:x.
↪dropna().sum())],
        'trafficSource.campaign': [('campaign', lambda x: x.dropna().max())],
        'trafficSource.source': [('source', lambda x: x.dropna().max())],
        'trafficSource.medium': [('medium', lambda x: x.dropna().max())],
        'trafficSource.keyword': [('keyword', lambda x: x.dropna().max())],
        'googleAds': [('googleAds', lambda x: x.dropna().max())],
        'trafficSource.referralPath': [('referralPath', lambda x: x.dropna().
↪max())],
        'trafficSource.isTrueDirect': [('isTrueDirect', lambda x: x.dropna().
↪max())],
        #'trafficSource.adwordsClickInfo.gclId': [('gclId', lambda x: x.
↪dropna().max())],
        #'customDimensions.index': [('customDimensions_index', lambda x: x.
↪dropna().max())],
        'customDimensions.value': [('customDimensions_value', lambda x: x.
↪dropna().max())]
        })

    tf.columns = tf.columns.droplevel()

    tf = pd.merge(tf, tf_target, left_on='fullVisitorId',␣
↪right_on='fullVisitorId')# merging target dataframe with feature dataframe
    return tf
```

[90]:
```
%%time
train_1 = Timeframewithfeatures(df_train, k = 1)
print('1st part complete')
```

```
1st part complete
Wall time: 33min 47s
```

```
[91]: %%time
      train_2 = Timeframewithfeatures(df_train, k = 2)
      print('2nd part complete')
```

```
2nd part complete
Wall time: 25min 6s
```

```
[92]: %%time
      train_3 = Timeframewithfeatures(df_train, k = 3)
      print('3rd part complete')
```

```
3rd part complete
Wall time: 33min 10s
```

```
[93]: %%time
      train_4 = Timeframewithfeatures(df_train, k = 4)
      print('4th part complete')
```

```
4th part complete
Wall time: 25min 29s
```

```
[58]: train_1.describe()
```

```
[58]:       first_ses_from_the_period_start last_ses_from_the_period_end  \
      count                          377186                       377186
      mean          83 days 16:52:23.149533      81 days 14:07:17.704474
      std           44 days 06:21:35.022014      44 days 05:35:13.133775
      min                   0 days 00:00:00              0 days 00:00:00
      25%                  48 days 00:00:00             48 days 00:00:00
      50%                  88 days 00:00:00             77 days 00:00:00
      75%                 118 days 00:00:00            117 days 00:00:00
      max                 167 days 00:00:00            167 days 00:00:00

                      interval_dates  visitStartTime_counts  visitNumber_max  \
      count                   377186          377186.000000    377186.000000
      mean    1 days 17:00:19.145991               1.235088         1.311226
      std     9 days 14:05:22.806362               1.273345         2.067036
      min            0 days 00:00:00               1.000000         1.000000
      25%            0 days 00:00:00               1.000000         1.000000
      50%            0 days 00:00:00               1.000000         1.000000
      75%            0 days 00:00:00               1.000000         1.000000
      max          167 days 00:00:00             186.000000       303.000000

              bounces_mean      newVisits       hits_sum       hits_min  \
      count  377186.000000  377186.000000  377186.000000  377186.000000
```

```
mean        0.523020       0.872220       5.886677       3.453402
std         0.487111       0.333846      18.314076       7.084872
min         0.000000       0.000000       1.000000       1.000000
25%         0.000000       1.000000       1.000000       1.000000
50%         0.750000       1.000000       2.000000       1.000000
75%         1.000000       1.000000       4.000000       3.000000
max         1.000000       1.000000    2248.000000     500.000000

            hits_max  …  sessionQualityDimMean  sessionQualityDimMax  \
count  377186.000000  …               377186.0              377186.0
mean        4.776630  …                    0.0                   0.0
std        10.881616  …                    0.0                   0.0
min         1.000000  …                    0.0                   0.0
25%         1.000000  …                    0.0                   0.0
50%         2.000000  …                    0.0                   0.0
75%         4.000000  …                    0.0                   0.0
max       500.000000  …                    0.0                   0.0

       timeOnSite_sum  timeOnSite_min  timeOnSite_max  timeOnSite_mean  \
count   377186.000000   377186.000000   377186.000000    377186.000000
mean       151.932832       75.897491      125.916177        95.378134
std        723.292035      233.705292      368.719906       257.231898
min          0.000000        0.000000        0.000000         0.000000
25%          0.000000        0.000000        0.000000         0.000000
50%          3.000000        0.000000        3.000000         2.000000
75%         77.000000       53.000000       74.000000        67.000000
max     225163.000000    10046.000000    19017.000000     10046.000000

        transactions  transactionRevenue_sum         target            ret
count  377186.000000            3.771860e+05  377186.000000  377186.000000
mean        0.015361            1.890948e+06       0.007292       0.004939
std         0.163354            5.123977e+07       0.368080       0.070106
min         0.000000            0.000000e+00       0.000000       0.000000
25%         0.000000            0.000000e+00       0.000000       0.000000
50%         0.000000            0.000000e+00       0.000000       0.000000
75%         0.000000            0.000000e+00       0.000000       0.000000
max        27.000000            1.602375e+10      24.653951       1.000000

[8 rows x 25 columns]
```

[94]:
```
### Construction of the test-set (by analogy as train-set)
print('Get test')
train_5 = df_test[df_test['date'] >= pd.to_datetime(20180501,␣
 ↪infer_datetime_format=True, format="%Y%m%d")] #using test data to create␣
 ↪train 5th window but we keep the target here as np.nan since we have to␣
 ↪predict that, later we will seperate this out
train_5_maxdate = max(train_5['date'])
```

```
train_5_mindate = min(train_5['date'])
```

Get test

[95]:

```
%%time
#calculating the features for test
train_5 = train_5.groupby('fullVisitorId').agg({
    'channelGrouping': [('channelGrouping', lambda x: x.dropna().max())],
    'date': [('first_ses_from_the_period_start', lambda x: x.dropna().min() -␣
↪train_5_mindate), #time of the first session start from the window strat
             ('last_ses_from_the_period_end', lambda x: train_5_maxdate - x.
↪dropna().max()), # gives last session from window end of user
             ('interval_dates', lambda x: x.max() - x.min())], #time between␣
↪first and last session of the user
    'visitStartTime': [('visitStartTime_counts', 'count')], #number of times a␣
↪visitor has visited the site
    'visitNumber': [('visitNumber_max', 'max')],
    'device.browser':  [('browser', 'max')],
    'device.operatingSystem': [('operatingSystem', lambda x: x.dropna().max())],
    'device.isMobile': [('isMobile', 'max')],
    'device.deviceCategory': [('deviceCategory', 'max')],
    'geoNetwork.continent': [('continent', lambda x: x.dropna().max())], #which␣
↪contitent category occurs max time for that user
    'geoNetwork.subContinent': [('subContinent', lambda x: x.dropna().max())],
    'geoNetwork.country': [('country', lambda x: x.dropna().max())],
    'geoNetwork.region': [('region', lambda x: x.dropna().max())],
    'geoNetwork.metro': [('metro', lambda x: x.dropna().max())],
    'geoNetwork.city': [('city', lambda x: x.dropna().max())],
    'geoNetwork.networkDomain': [('networkDomain', lambda x: x.dropna().
↪max())],# for categorical column max function will take the max no. og␣
↪occurance of that particaular category when grouped by full
     #`totals.visits` is excluded because same accross dataset
    'totals.bounces': [
        #('bounces_count', lambda x: x.dropna().count()),
        #('bounces_sum', lambda x: x.dropna().sum()),
        ('bounces_mean', 'mean')],
    'totals.newVisits': [('newVisits', 'min')], # if user visit more than one␣
↪time, get 0, else 1
    'totals.hits': [('hits_sum', 'sum'),
                    ('hits_min', 'min'),
                    ('hits_max', 'max'),
                    ('hits_mean', 'mean')],
    'totals.pageviews': [('pageviews_sum', 'sum'), #getting toal number of␣
↪pageviews for each visitor
                         ('pageviews_min', 'min'), #getting min number of␣
↪pageviews for each visitor
```

```
                         ('pageviews_max', 'max'), #getting max number of␣
 ↪pageviews for each visitor
                         ('pageviews_mean', 'mean')],

    'totals.sessionQualityDim': [('sessionQualityDimMin', 'min'),
                                 ('sessionQualityDimMax', 'max'),
                                 ('sessionQualityDimMean', 'mean'),
                                 ('sessionQualityDimSum', 'sum')],
    'totals.timeOnSite': [('timeOnSite_sum', 'sum'),#total time spent on site
                          ('timeOnSite_min', 'mean'), #min time spent on site
                          ('timeOnSite_max', 'max'), # max time spent on site
                          ('timeOnSite_mean', 'mean')], #mean time spent on site
    'totals.transactions' : [('transactions', lambda x:x.dropna().sum())],
    'totals.transactionRevenue':  [('transactionRevenue_sum',  lambda x:x.
 ↪dropna().sum())],
    'trafficSource.campaign': [('campaign', lambda x: x.dropna().max())],
    'trafficSource.source': [('source', lambda x: x.dropna().max())],
    'trafficSource.medium': [('medium', lambda x: x.dropna().max())],
    'trafficSource.keyword': [('keyword', lambda x: x.dropna().max())],
    'googleAds': [('googleAds', lambda x: x.dropna().max())],
    'trafficSource.referralPath': [('referralPath', lambda x: x.dropna().
 ↪max())],
    'trafficSource.isTrueDirect': [('isTrueDirect', lambda x: x.dropna().
 ↪max())],
    #'trafficSource.adwordsClickInfo.gclId': [('gclId', lambda x: x.dropna().
 ↪max())],
    #'customDimensions.index': [('customDimensions_index', lambda x: x.dropna().
 ↪max())],
    'customDimensions.value': [('customDimensions_value', lambda x: x.dropna().
 ↪max())]

})

train_5.columns = train_5.columns.droplevel()
train_5['target'] = np.nan #becomes easy to seperate train and test data later
train_5['ret'] = np.nan
```

```
Wall time: 26min 19s
```

[177]:

```
#save the file
train_5.to_pickle('pickle/train_5')
```

[96]:

```
train_5 = train_5.reset_index()
```

[97]:

```
train_5 = train_5[['fullVisitorId', 'channelGrouping',␣
 ↪'first_ses_from_the_period_start', 'last_ses_from_the_period_end',
                   'interval_dates', 'visitStartTime_counts',␣
 ↪'visitNumber_max', 'browser',  'operatingSystem', 'isMobile',
                   'deviceCategory', 'continent', 'subContinent', 'country',␣
 ↪'metro', 'region', 'city', 'networkDomain',
                   'bounces_mean', 'newVisits', 'hits_sum', 'hits_min',␣
 ↪'hits_max', 'hits_mean', 'pageviews_sum',
                   'pageviews_min', 'pageviews_max', 'pageviews_mean',␣
 ↪'sessionQualityDimMin', 'sessionQualityDimMax',
                   'sessionQualityDimSum', 'sessionQualityDimMean',␣
 ↪'timeOnSite_sum', 'timeOnSite_min', 'timeOnSite_max',
                   'timeOnSite_mean', 'transactions', 'transactionRevenue_sum',␣
 ↪'campaign', 'source', 'medium', 'keyword',
                   'isTrueDirect', 'referralPath', 'googleAds',␣
 ↪'customDimensions_value', 'target', 'ret']]

train_5.columns
```

[97]:

```
Index(['fullVisitorId', 'channelGrouping', 'first_ses_from_the_period_start',
       'last_ses_from_the_period_end', 'interval_dates',
       'visitStartTime_counts', 'visitNumber_max', 'browser',
       'operatingSystem', 'isMobile', 'deviceCategory', 'continent',
       'subContinent', 'country', 'metro', 'region', 'city', 'networkDomain',
       'bounces_mean', 'newVisits', 'hits_sum', 'hits_min', 'hits_max',
       'hits_mean', 'pageviews_sum', 'pageviews_min', 'pageviews_max',
       'pageviews_mean', 'sessionQualityDimMin', 'sessionQualityDimMax',
       'sessionQualityDimSum', 'sessionQualityDimMean', 'timeOnSite_sum',
       'timeOnSite_min', 'timeOnSite_max', 'timeOnSite_mean', 'transactions',
       'transactionRevenue_sum', 'campaign', 'source', 'medium', 'keyword',
       'isTrueDirect', 'referralPath', 'googleAds', 'customDimensions_value',
       'target', 'ret'],
      dtype='object')
```

[98]:

```
# Concat all train sets
final_train = pd.concat([train_1, train_2, train_3, train_4, train_5], axis=0,␣
 ↪sort=False).reset_index(drop=True)
final_train['interval_dates'] = final_train['interval_dates'].dt.days
final_train['first_ses_from_the_period_start'] =␣
 ↪final_train['first_ses_from_the_period_start'].dt.days
final_train['last_ses_from_the_period_end'] =␣
 ↪final_train['last_ses_from_the_period_end'].dt.days
```

[99]:

```
#final_train.to_pickle('pickle/final_train_0521')
```

[2]:

```
"""
import pickle
x = open('pickle/final_train_0519', 'rb')
```

```
final_train = pickle.load(x)
"""
```

```
[100]: #test data
       test = final_train[final_train['target'].isnull()] #seperating the test from␣
        ↪the final dataset
       test
```

```
[100]:                 fullVisitorId channelGrouping  first_ses_from_the_period_start  \
       1344567  0000018966949534117  Organic Search                              104
       1344568  0000039738481224681          Direct                               43
       1344569  0000073585230191399  Organic Search                               33
       1344570  0000087588448856385  Organic Search                               36
       1344571  0000149787903119437  Organic Search                               20
       …                        …             …                              …
       1641092  9999862054614696520  Organic Search                                7
       1641093  9999898168621645223  Organic Search                               67
       1641094   999990167740728398          Direct                               93
       1641095  9999915620249883537  Organic Search                               46
       1641096  9999947552481876143  Organic Search                              108

                last_ses_from_the_period_end  interval_dates  visitStartTime_counts  \
       1344567                            63               0                      1
       1344568                           124               0                      1
       1344569                           134               0                      1
       1344570                           131               0                      1
       1344571                           147               0                      1
       …                                 …             …                    …
       1641092                           160               0                      1
       1641093                           100               0                      1
       1641094                            74               0                      1
       1641095                           121               0                      1
       1641096                            59               0                      1

                visitNumber_max     browser operatingSystem  isMobile  … campaign  \
       1344567                1      Chrome       Macintosh     False  …      NaN
       1344568                1      Chrome         Android      True  …      NaN
       1344569                1      Safari             iOS      True  …      NaN
       1344570                1      Chrome         Windows     False  …      NaN
       1344571                1      Chrome         Android      True  …      NaN
       …                    …         …             …       …  …      …
       1641092                1      Chrome       Macintosh     False  …      NaN
       1641093                1      Safari             iOS      True  …      NaN
       1641094                1  Opera Mini             NaN      True  …      NaN
       1641095                1      Chrome         Windows     False  …      NaN
       1641096                1      Chrome         Windows     False  …      NaN
```

```
           source   medium keyword googleAds referralPath isTrueDirect  \
1344567    google  organic     NaN         1          NaN        False
1344568  (direct)      NaN     NaN         1          NaN         True
1344569    google  organic     NaN         1          NaN        False
1344570    google  organic     NaN         1          NaN        False
1344571    google  organic     NaN         1          NaN        False
…           …       …     …       …          …          …
1641092    google  organic     NaN         1          NaN        False
1641093    google  organic     NaN         1          NaN        False
1641094  (direct)      NaN     NaN         1          NaN         True
1641095     yahoo  organic     NaN         1          NaN        False
1641096    google  organic     NaN         1          NaN        False

        customDimensions_value  target  ret
1344567          North America     NaN  NaN
1344568          North America     NaN  NaN
1344569          North America     NaN  NaN
1344570                   APAC     NaN  NaN
1344571                    NaN     NaN  NaN
…                         …     …  …
1641092                   APAC     NaN  NaN
1641093          North America     NaN  NaN
1641094                   APAC     NaN  NaN
1641095          North America     NaN  NaN
1641096          North America     NaN  NaN

[296530 rows x 48 columns]
```

[101]:

```
#train data
train = final_train[final_train['target'].notnull()]
train
```

[101]:

```
               fullVisitorId channelGrouping   first_ses_from_the_period_start  \
0        0000010278554503158  Organic Search                                80
1        0000020424342248747  Organic Search                               121
2         000005103959234087  Organic Search                                20
3        0000093957001069502  Organic Search                                57
4        0000114156543135683          Social                                 7
…                       …             …                               …
1344562  9999818112872622034          Direct                                85
1344563  9999882818693474736        Referral                               132
1344564  9999941518946450908  Organic Search                                18
1344565  9999969142283897422  Organic Search                                70
1344566  9999985820452794361     Paid Search                               122

         last_ses_from_the_period_end  interval_dates  visitStartTime_counts  \
0                                  87               0                      1
```

```
1                                  46               0                      1
2                                 147               0                      1
3                                 110               0                      1
4                                 160               0                      1
…                               …             …                    …
1344562                            48               0                      1
1344563                             1               0                      1
1344564                           115               0                      1
1344565                            63               0                      1
1344566                            11               0                      2

         visitNumber_max browser operatingSystem  isMobile  …  \
0                      1  Chrome       Macintosh     False  …
1                      1  Chrome         Windows     False  …
2                      1  Chrome         Android      True  …
3                      1  Chrome         Windows     False  …
4                      1  Safari       Macintosh     False  …
…                  …     …             …       …  …
1344562                1  Chrome         Android      True  …
1344563                1  Chrome       Macintosh     False  …
1344564                1  Chrome           Linux     False  …
1344565                1  Chrome         Windows     False  …
1344566                2  Chrome         Windows     False  …

                                                  campaign       source  \
0                                                      NaN       google
1                                                      NaN     (direct)
2                                                      NaN       google
3                                                      NaN     (direct)
4                                                      NaN  youtube.com
…                                                    …          …
1344562                                                NaN     (direct)
1344563                                                NaN     (direct)
1344564                                                NaN       google
1344565                                                NaN       google
1344566  "google + redesign/Accessories March 17" All U…       google

           medium                          keyword googleAds referralPath  \
0         organic                              NaN         1          NaN
1             NaN                              NaN         1          NaN
2         organic                              NaN         1          NaN
3             NaN                              NaN         1          NaN
4        referral                              NaN         1   /yt/about/
…           …                              …       …          …
1344562       NaN                              NaN         1          NaN
1344563       NaN                              NaN         1          NaN
1344564   organic                              NaN         1          NaN
```

```
1344565   organic                              NaN        1          NaN
1344566   organic  (Remarketing/Content targeting)        1          NaN

        isTrueDirect customDimensions_value  target  ret
0              False                    NaN     0.0  0.0
1              False                    NaN     0.0  0.0
2              False          North America     0.0  0.0
3              False          North America     0.0  0.0
4              False                   EMEA     0.0  0.0
…              …                          …      …   …
1344562         True                    NaN     0.0  0.0
1344563        False          North America     0.0  0.0
1344564        False                    NaN     0.0  0.0
1344565        False          North America     0.0  0.0
1344566        False          North America     0.0  0.0

[1344567 rows x 48 columns]
```

[102]:

```
target_cols = ['target', 'ret', 'transactionRevenue_sum', 'fullVisitorId']
target_test = test['transactionRevenue_sum'].astype('float').apply(lambda x: np.
 ↪log1p(x))
```

[103]:

```
# train data
train_x = train.drop(target_cols, axis = 1)
train_x_id = train['fullVisitorId'].astype('str')
train_y = train['target']
train_ret = train['ret']

test_x = test.drop(target_cols, axis = 1)
test_x_id = test['fullVisitorId'].astype('str')
test_y = target_test
test_ret = test['ret']
```

[104]:

```
#correct dtype train
train_x['isMobile'] = train_x['isMobile'].astype('object')
train_x['isTrueDirect'] = train_x['isTrueDirect'].astype('object')

#correct dtype test
test_x['isMobile'] = test_x['isMobile'].astype('object')
test_x['isTrueDirect'] = test_x['isTrueDirect'].astype('object')
```

[105]:

```
#taking cat and numerical columns seperately
cat_cols = [x for x in train_x.columns if train_x[x].dtype == 'object']
num_cols = set(train_x.columns) - set(cat_cols)
```

[106]:

```
print(num_cols)
print('\n')
print(cat_cols)
```

```
{'pageviews_mean', 'hits_min', 'visitStartTime_counts', 'sessionQualityDimMin',
'pageviews_sum', 'pageviews_max', 'sessionQualityDimSum',
'sessionQualityDimMax', 'last_ses_from_the_period_end', 'hits_max', 'newVisits',
'bounces_mean', 'timeOnSite_max', 'visitNumber_max',
'first_ses_from_the_period_start', 'hits_mean', 'transactions',
'timeOnSite_mean', 'interval_dates', 'pageviews_min', 'timeOnSite_min',
'sessionQualityDimMean', 'hits_sum', 'timeOnSite_sum'}


['channelGrouping', 'browser', 'operatingSystem', 'isMobile', 'deviceCategory',
'continent', 'subContinent', 'country', 'region', 'metro', 'city',
'networkDomain', 'campaign', 'source', 'medium', 'keyword', 'googleAds',
'referralPath', 'isTrueDirect', 'customDimensions_value']
```

[107]:

```
# fill missing in train
for col in cat_cols:
    train_x[col].fillna('missing', inplace=True)
    test_x[col].fillna('missing', inplace=True)

for col in num_cols:
    train_x[col].fillna(0, inplace=True)
    test_x[col].fillna(0, inplace=True)
```

[108]:

```
train_x_copy = train_x.copy()
test_x_copy = test_x.copy()
```

1.2.3 4.2 Label Encoding

[109]:

```
from sklearn.preprocessing import LabelEncoder
for col in cat_cols:
    print("transform column {}".format(col))
    lbe = LabelEncoder()
    lbe.fit(pd.concat([train_x[col],test_x[col]]).astype("str"))
    train_x[col] = lbe.transform(train_x[col].astype("str"))
    test_x[col] = lbe.transform(test_x[col].astype("str"))
```

```
transform column channelGrouping
transform column browser
transform column operatingSystem
transform column isMobile
transform column deviceCategory
transform column continent
transform column subContinent
transform column country
transform column region
transform column metro
transform column city
transform column networkDomain
transform column campaign
transform column source
transform column medium
transform column keyword
transform column googleAds
transform column referralPath
transform column isTrueDirect
transform column customDimensions_value
```

1.2.4 4.3 Normalization

[110]:

```
from sklearn.preprocessing import StandardScaler
for col in num_cols:
    norm = StandardScaler()
    norm.fit(train_x[col].values.reshape(-1,1))
    train_x[col] = norm.transform(train_x[col].values.reshape(-1,1))
    test_x[col] = norm.transform(test_x[col].values.reshape(-1,1))
```

1.2.5 4.4 LightGBM

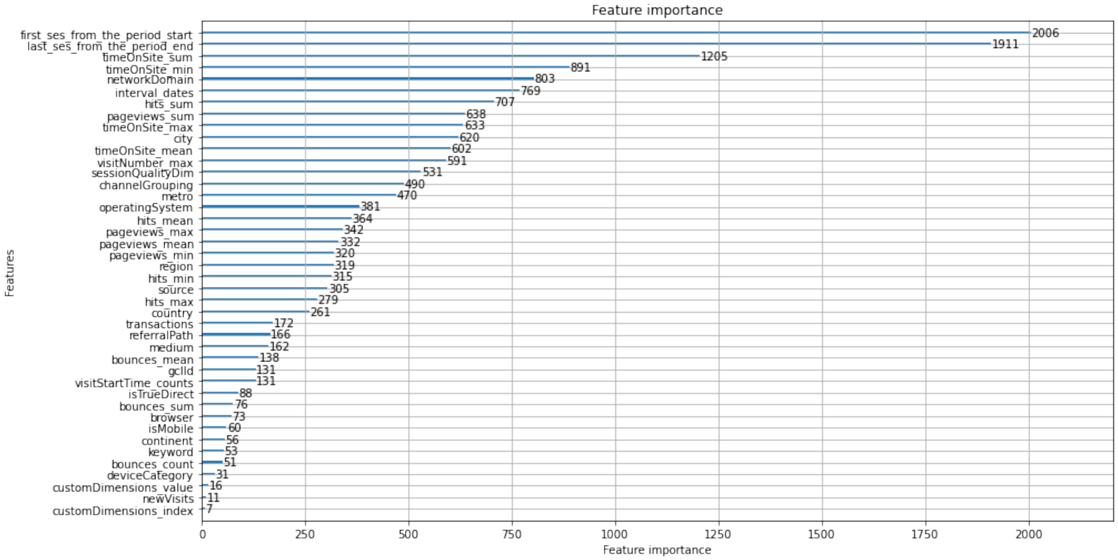
[15]:

```
#lets specify thr parametrs first
import lightgbm as lgb
params1 = {'task': 'train',
           'num_iterations': 50, #no of boost round or number of estimators
           'boosting':'gbdt', #gradient boosted decision tree
           'objective' : 'regression',
           'metric': ['rmse'], #roort mean suqre error
           'is_training_metric': True, #return metric on train data
           'learning_rate' :0.01,
           'max_leaves': 1000,
           'feature_fraction': 0.8, #col sampling
           'bagging_fraction': 0.8,
           'colsample_bytree': 1.0,
           'max_depth':30,
           'min_child_samples': 100,
           'reg_alpha': 1,
           'reg_lambda': 1}

#create train data
dtrain1 = lgb.Dataset(data = train_x, label = list(train_y.values))

#training
model = lgb.train(params1, dtrain1)
```

```
cv = lgb.cv(params1, dtrain1, nfold= 5, stratified=False, metrics = 'rmse',␣
↪eval_train_metric= True)
#predicting

pred_train = model.predict(train_x)
pred_test = model.predict(test_x)

lgb.plot_importance(model, figsize=(14.70, 8.27));
```

```
C:\Users\FORGE-15 I7\Anaconda3\lib\site-packages\lightgbm\engine.py:148:
UserWarning: Found `num_iterations` in params. Will use it instead of argument
  warnings.warn("Found `{}` in params. Will use it instead of
argument".format(alias))
C:\Users\FORGE-15 I7\Anaconda3\lib\site-packages\lightgbm\engine.py:503:
UserWarning: Found `num_iterations` in params. Will use it instead of argument
  warnings.warn("Found `{}` in params. Will use it instead of
argument".format(alias))
```

[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204b524d608>
```



[81]:
```
from sklearn.metrics import mean_squared_error
train_mse = mean_squared_error(train_y.values, pred_train)
print('train_rmse',np.sqrt(train_mse))

test_mse = mean_squared_error(test_y.values, pred_test)
print('test_rmse',np.sqrt(test_mse))
```

```
x = pd.DataFrame(data ={'fullVisitorId': test['fullVisitorId'].astype('str'),␣
 ↪'PredictedLogRevenue': pred_test})
x.to_csv('results/submission_0521_lgbm.csv', index = False)
```

```
train_rmse 0.305258151647652
test_rmse 2.108077508142151
```

1.2.6 4.5 CatBoost

```
[127]: train_x[['channelGrouping', 'first_ses_from_the_period_start',␣
        ↪'last_ses_from_the_period_end', 'interval_dates',
               'visitStartTime_counts', 'visitNumber_max', 'isMobile',␣
        ↪'operatingSystem', 'deviceCategory', 'continent',
               'country', 'region', 'city', 'networkDomain', 'bounces_mean',␣
        ↪'hits_sum', 'hits_min', 'hits_max', 'hits_mean',
               'pageviews_sum', 'pageviews_min', 'pageviews_max', 'pageviews_mean',␣
        ↪'sessionQualityDimMax', 'sessionQualityDimMean',
               'timeOnSite_max', 'timeOnSite_mean', 'timeOnSite_sum', 'transactions',␣
        ↪'campaign', 'source', 'medium', 'referralPath']]
```

```
[127]:          channelGrouping  first_ses_from_the_period_start  \
        0                      4                        -0.018261
        1                      4                         0.875995
        2                      4                        -1.326927
        3                      4                        -0.519916
        4                      7                        -1.610472
        …                    …                             …
        1344562                2                         0.090795
        1344563                6                         1.115917
        1344564                4                        -1.370550
        1344565                4                        -0.236372
        1344566                5                         0.897806

                 last_ses_from_the_period_end  interval_dates  visitStartTime_counts  \
        0                            0.219949       -0.185269              -0.207383
        1                           -0.677161       -0.185269              -0.207383
        2                            1.532793       -0.185269              -0.207383
        3                            0.723206       -0.185269              -0.207383
        4                            1.817242       -0.185269              -0.207383
        …                                 …             …                     …
        1344562                     -0.633399       -0.185269              -0.207383
        1344563                     -1.661794       -0.185269              -0.207383
        1344564                      0.832609       -0.185269              -0.207383
        1344565                     -0.305189       -0.185269              -0.207383
        1344566                     -1.442986       -0.185269               0.559146

                 visitNumber_max  browser  operatingSystem  isMobile  deviceCategory  \
0              -0.165488        7                6         0               0
1              -0.165488        7               21         0               0
2              -0.165488        7                0         1               1
3              -0.165488        7               21         0               0
4              -0.165488       23                6         0               0
…                  …      …              …       …             …
1344562        -0.165488        7                0         1               1
1344563        -0.165488        7                6         0               0
1344564        -0.165488        7                5         0               0
1344565        -0.165488        7               21         0               0
1344566         0.275053        7               21         0               0

         …  timeOnSite_mean  transactions  campaign  source  medium  \
0        …         0.350777     -0.092882        41     110       4
1        …         0.736632     -0.092882        41       0       3
2        …         0.380746     -0.092882        41     110       4
3        …        -0.360993     -0.092882        41       0       3
4        …        -0.375978     -0.092882        41     380       5
…      …              …           …       …     …     …
1344562  …         0.268362     -0.092882        41       0       3
1344563  …         2.115218      6.367849        41       0       3
1344564  …        -0.375978     -0.092882        41     110       4
1344565  …         0.140992     -0.092882        41     110       4
1344566  …         0.335793     -0.092882         1     110       4

         keyword  googleAds  referralPath  isTrueDirect  \
0              8          1          3571             0
1              8          1          3571             0
2              8          1          3571             0
3              8          1          3571             0
4              8          1          2900             0
…          …        …           …           …
1344562        8          1          3571             1
1344563        8          1          3571             0
1344564        8          1          3571             0
1344565        8          1          3571             0
1344566        0          1          3571             0

         customDimensions_value
0                             5
1                             5
2                             3
3                             3
4                             2
…                         …
1344562                       5
1344563                       3
```

```
1344564                        5
1344565                        3
1344566                        3

[1344567 rows x 44 columns]
```

[27]:

```
from catboost import CatBoostRegressor

grid = {'learning_rate': [0.03, 0.05, 0.1],
        'depth': [8, 10, 12],
        'l2_leaf_reg': [1, 3, 5, 7, 9],
        'bagging_temperature' : [0, 0.2, 0.5]}

model = CatBoostRegressor(iterations = 1000,
                          #metric_period = 50,
                          od_wait = 20,
                          eval_metric='RMSE')

grid_search_result = model.grid_search(grid,
                                       X = train_x,
                                       y = train_y,
                                       cv = 5,
                                       refit = True,
                                       plot = True)
```

```
<IPython.core.display.HTML object>


MetricVisualizer(layout=Layout(align_self='stretch', height='500px'))


Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.322660322
bestIteration = 125

0:      loss: 0.3226603 best: 0.3226603 (0)     total: 13.5s    remaining: 30m
3s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231627942
bestIteration = 35
```

```
1:      loss: 0.3231628 best: 0.3226603 (0)     total: 18.4s    remaining: 20m
24s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3236238823
bestIteration = 3

2:      loss: 0.3236239 best: 0.3226603 (0)     total: 20.7s    remaining: 15m
11s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230988267
bestIteration = 49

3:      loss: 0.3230988 best: 0.3226603 (0)     total: 26.9s    remaining: 14m
40s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.323270111
bestIteration = 24

4:      loss: 0.3232701 best: 0.3226603 (0)     total: 30.8s    remaining: 13m
20s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232080941
bestIteration = 64

5:      loss: 0.3232081 best: 0.3226603 (0)     total: 38.1s    remaining: 13m
38s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231862317
```

```
bestIteration = 51

6:      loss: 0.3231862 best: 0.3226603 (0)     total: 44.5s    remaining: 13m
33s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231208476
bestIteration = 69

7:      loss: 0.3231208 best: 0.3226603 (0)     total: 52.7s    remaining: 13m
56s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3229508022
bestIteration = 60

8:      loss: 0.3229508 best: 0.3226603 (0)     total: 1m       remaining: 14m
6s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232353633
bestIteration = 69

9:      loss: 0.3232354 best: 0.3226603 (0)     total: 1m 8s    remaining: 14m
19s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232664204
bestIteration = 45

10:     loss: 0.3232664 best: 0.3226603 (0)     total: 1m 14s   remaining: 14m
4s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.
```

```
Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3233285383
bestIteration = 51

11:     loss: 0.3233285 best: 0.3226603 (0)     total: 1m 21s   remaining: 13m
57s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235121174
bestIteration = 57

12:     loss: 0.3235121 best: 0.3226603 (0)     total: 1m 29s   remaining: 13m
55s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231903231
bestIteration = 66

13:     loss: 0.3231903 best: 0.3226603 (0)     total: 1m 36s   remaining: 13m
58s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235751148
bestIteration = 26

14:     loss: 0.3235751 best: 0.3226603 (0)     total: 1m 41s   remaining: 13m
31s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3227134191
bestIteration = 140

15:     loss: 0.3227134 best: 0.3226603 (0)     total: 2m 34s   remaining: 19m
11s
```

```
Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231078607
bestIteration = 30

16:     loss: 0.3231079 best: 0.3226603 (0)     total: 2m 51s   remaining: 19m
47s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231089907
bestIteration = 8

17:     loss: 0.3231090 best: 0.3226603 (0)     total: 3m       remaining: 19m
33s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231716647
bestIteration = 79

18:     loss: 0.3231717 best: 0.3226603 (0)     total: 3m 32s   remaining: 21m
38s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230439136
bestIteration = 66

19:     loss: 0.3230439 best: 0.3226603 (0)     total: 4m 3s    remaining: 23m
17s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232121213
bestIteration = 12
```

```
20:     loss: 0.3232121 best: 0.3226603 (0)     total: 4m 13s   remaining: 22m
57s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3227871207
bestIteration = 93

21:     loss: 0.3227871 best: 0.3226603 (0)     total: 4m 50s   remaining: 24m
52s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232131631
bestIteration = 35

22:     loss: 0.3232132 best: 0.3226603 (0)     total: 5m 9s    remaining: 25m
5s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232831229
bestIteration = 12

23:     loss: 0.3232831 best: 0.3226603 (0)     total: 5m 21s   remaining: 24m
48s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3233319789
bestIteration = 58

24:     loss: 0.3233320 best: 0.3226603 (0)     total: 5m 47s   remaining: 25m
30s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231277088
```

```
bestIteration = 73

25:     loss: 0.3231277 best: 0.3226603 (0)     total: 6m 26s   remaining: 26m
59s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232521386
bestIteration = 13

26:     loss: 0.3232521 best: 0.3226603 (0)     total: 6m 39s   remaining: 26m
36s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231605364
bestIteration = 119

27:     loss: 0.3231605 best: 0.3226603 (0)     total: 7m 24s   remaining: 28m
19s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3236634193
bestIteration = 36

28:     loss: 0.3236634 best: 0.3226603 (0)     total: 7m 46s   remaining: 28m
24s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235110817
bestIteration = 50

29:     loss: 0.3235111 best: 0.3226603 (0)     total: 8m 9s    remaining: 28m
32s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.
```

```
Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235360582
bestIteration = 40

30:     loss: 0.3235361 best: 0.3226603 (0)     total: 8m 43s   remaining: 29m
14s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3236142722
bestIteration = 15

31:     loss: 0.3236143 best: 0.3226603 (0)     total: 9m 2s    remaining: 29m
5s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3239267074
bestIteration = 12

32:     loss: 0.3239267 best: 0.3226603 (0)     total: 9m 20s   remaining: 28m
51s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230916954
bestIteration = 69

33:     loss: 0.3230917 best: 0.3226603 (0)     total: 10m 8s   remaining: 30m
8s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3229143039
bestIteration = 59

34:     loss: 0.3229143 best: 0.3226603 (0)     total: 10m 51s  remaining: 31m
2s
```

```
Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.323425614
bestIteration = 18

35:     loss: 0.3234256 best: 0.3226603 (0)     total: 11m 14s  remaining: 30m
54s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232016515
bestIteration = 60

36:     loss: 0.3232017 best: 0.3226603 (0)     total: 12m 5s   remaining: 32m
1s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3229396962
bestIteration = 35

37:     loss: 0.3229397 best: 0.3226603 (0)     total: 12m 36s  remaining: 32m
10s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230268767
bestIteration = 40

38:     loss: 0.3230269 best: 0.3226603 (0)     total: 13m 9s   remaining: 32m
23s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3234954548
bestIteration = 60
```

```
39:     loss: 0.3234955 best: 0.3226603 (0)     total: 13m 57s  remaining: 33m
9s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235545059
bestIteration = 64

40:     loss: 0.3235545 best: 0.3226603 (0)     total: 14m 49s  remaining: 33m
59s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230662202
bestIteration = 77

41:     loss: 0.3230662 best: 0.3226603 (0)     total: 15m 53s  remaining: 35m
10s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3234598239
bestIteration = 77

42:     loss: 0.3234598 best: 0.3226603 (0)     total: 16m 56s  remaining: 36m
14s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231765363
bestIteration = 149

43:     loss: 0.3231765 best: 0.3226603 (0)     total: 18m 43s  remaining: 38m
43s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232707916
```

```
bestIteration = 62

44:     loss: 0.3232708 best: 0.3226603 (0)     total: 19m 28s  remaining: 38m
57s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.322660322
bestIteration = 125

45:     loss: 0.3226603 best: 0.3226603 (0)     total: 19m 42s  remaining: 38m
8s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231627942
bestIteration = 35

46:     loss: 0.3231628 best: 0.3226603 (0)     total: 19m 48s  remaining: 37m
4s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3236238823
bestIteration = 3

47:     loss: 0.3236239 best: 0.3226603 (0)     total: 19m 50s  remaining: 35m
57s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230988267
bestIteration = 49

48:     loss: 0.3230988 best: 0.3226603 (0)     total: 19m 57s  remaining: 35m
2s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.
```

```
Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.323270111
bestIteration = 24

49:     loss: 0.3232701 best: 0.3226603 (0)     total: 20m 2s   remaining: 34m
3s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232080941
bestIteration = 64

50:     loss: 0.3232081 best: 0.3226603 (0)     total: 20m 10s  remaining: 33m
13s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231862317
bestIteration = 51

51:     loss: 0.3231862 best: 0.3226603 (0)     total: 20m 16s  remaining: 32m
22s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231208476
bestIteration = 69

52:     loss: 0.3231208 best: 0.3226603 (0)     total: 20m 25s  remaining: 31m
36s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3229508022
bestIteration = 60

53:     loss: 0.3229508 best: 0.3226603 (0)     total: 20m 33s  remaining: 30m
50s
```

```
Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232353633
bestIteration = 69

54:     loss: 0.3232354 best: 0.3226603 (0)     total: 20m 42s  remaining: 30m
6s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232664204
bestIteration = 45

55:     loss: 0.3232664 best: 0.3226603 (0)     total: 20m 48s  remaining: 29m
21s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3233285383
bestIteration = 51

56:     loss: 0.3233285 best: 0.3226603 (0)     total: 20m 55s  remaining: 28m
38s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235121174
bestIteration = 57

57:     loss: 0.3235121 best: 0.3226603 (0)     total: 21m 3s   remaining: 27m
57s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231903231
bestIteration = 66
```

```
58:     loss: 0.3231903 best: 0.3226603 (0)     total: 21m 11s  remaining: 27m
17s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235751148
bestIteration = 26

59:     loss: 0.3235751 best: 0.3226603 (0)     total: 21m 16s  remaining: 26m
35s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3227134191
bestIteration = 140

60:     loss: 0.3227134 best: 0.3226603 (0)     total: 22m 12s  remaining: 26m
56s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231078607
bestIteration = 30

61:     loss: 0.3231079 best: 0.3226603 (0)     total: 22m 29s  remaining: 26m
29s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231089907
bestIteration = 8

62:     loss: 0.3231090 best: 0.3226603 (0)     total: 22m 39s  remaining: 25m
53s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231716647
```

```
bestIteration = 79

63:     loss: 0.3231717 best: 0.3226603 (0)     total: 23m 12s  remaining: 25m
44s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230439136
bestIteration = 66

64:     loss: 0.3230439 best: 0.3226603 (0)     total: 23m 42s  remaining: 25m
31s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232121213
bestIteration = 12

65:     loss: 0.3232121 best: 0.3226603 (0)     total: 23m 53s  remaining: 24m
58s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3227871207
bestIteration = 93

66:     loss: 0.3227871 best: 0.3226603 (0)     total: 24m 31s  remaining: 24m
53s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232131631
bestIteration = 35

67:     loss: 0.3232132 best: 0.3226603 (0)     total: 24m 49s  remaining: 24m
27s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.
```

```
Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232831229
bestIteration = 12

68:     loss: 0.3232831 best: 0.3226603 (0)     total: 25m 1s   remaining: 23m
55s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3233319789
bestIteration = 58

69:     loss: 0.3233320 best: 0.3226603 (0)     total: 25m 27s  remaining: 23m
38s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231277088
bestIteration = 73

70:     loss: 0.3231277 best: 0.3226603 (0)     total: 25m 58s  remaining: 23m
25s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232521386
bestIteration = 13

71:     loss: 0.3232521 best: 0.3226603 (0)     total: 26m 9s   remaining: 22m
53s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231605364
bestIteration = 119

72:     loss: 0.3231605 best: 0.3226603 (0)     total: 26m 56s  remaining: 22m
52s
```

```
Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3236634193
bestIteration = 36

73:     loss: 0.3236634 best: 0.3226603 (0)     total: 27m 15s  remaining: 22m
28s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235110817
bestIteration = 50

74:     loss: 0.3235111 best: 0.3226603 (0)     total: 27m 40s  remaining: 22m
8s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235360582
bestIteration = 40

75:     loss: 0.3235361 best: 0.3226603 (0)     total: 28m 16s  remaining: 21m
56s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3236142722
bestIteration = 15

76:     loss: 0.3236143 best: 0.3226603 (0)     total: 28m 36s  remaining: 21m
33s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3239267074
bestIteration = 12
```

```
77:     loss: 0.3239267 best: 0.3226603 (0)     total: 28m 55s  remaining: 21m
8s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230916954
bestIteration = 69

78:     loss: 0.3230917 best: 0.3226603 (0)     total: 29m 43s  remaining: 21m
4s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3229143039
bestIteration = 59

79:     loss: 0.3229143 best: 0.3226603 (0)     total: 30m 31s  remaining: 20m
58s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.323425614
bestIteration = 18

80:     loss: 0.3234256 best: 0.3226603 (0)     total: 30m 51s  remaining: 20m
34s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232016515
bestIteration = 60

81:     loss: 0.3232017 best: 0.3226603 (0)     total: 31m 34s  remaining: 20m
24s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3229396962
```

```
bestIteration = 35

82:     loss: 0.3229397 best: 0.3226603 (0)     total: 32m 3s   remaining: 20m
5s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230268767
bestIteration = 40

83:     loss: 0.3230269 best: 0.3226603 (0)     total: 32m 34s  remaining: 19m
46s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3234954548
bestIteration = 60

84:     loss: 0.3234955 best: 0.3226603 (0)     total: 33m 14s  remaining: 19m
33s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235545059
bestIteration = 64

85:     loss: 0.3235545 best: 0.3226603 (0)     total: 33m 57s  remaining: 19m
20s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230662202
bestIteration = 77

86:     loss: 0.3230662 best: 0.3226603 (0)     total: 34m 49s  remaining: 19m
12s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.
```

```
Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3234598239
bestIteration = 77

87:     loss: 0.3234598 best: 0.3226603 (0)     total: 35m 40s  remaining: 19m
3s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231765363
bestIteration = 149

88:     loss: 0.3231765 best: 0.3226603 (0)     total: 37m 12s  remaining: 19m
13s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232707916
bestIteration = 62

89:     loss: 0.3232708 best: 0.3226603 (0)     total: 37m 58s  remaining: 18m
59s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.322660322
bestIteration = 125

90:     loss: 0.3226603 best: 0.3226603 (0)     total: 38m 12s  remaining: 18m
28s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231627942
bestIteration = 35

91:     loss: 0.3231628 best: 0.3226603 (0)     total: 38m 17s  remaining: 17m
53s
```

```
Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3236238823
bestIteration = 3

92:     loss: 0.3236239 best: 0.3226603 (0)     total: 38m 19s  remaining: 17m
18s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230988267
bestIteration = 49

93:     loss: 0.3230988 best: 0.3226603 (0)     total: 38m 26s  remaining: 16m
45s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.323270111
bestIteration = 24

94:     loss: 0.3232701 best: 0.3226603 (0)     total: 38m 30s  remaining: 16m
12s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232080941
bestIteration = 64

95:     loss: 0.3232081 best: 0.3226603 (0)     total: 38m 38s  remaining: 15m
41s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231862317
bestIteration = 51
```

```
96:     loss: 0.3231862 best: 0.3226603 (0)     total: 38m 45s  remaining: 15m
10s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231208476
bestIteration = 69

97:     loss: 0.3231208 best: 0.3226603 (0)     total: 38m 53s  remaining: 14m
40s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3229508022
bestIteration = 60

98:     loss: 0.3229508 best: 0.3226603 (0)     total: 39m      remaining: 14m
11s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232353633
bestIteration = 69

99:     loss: 0.3232354 best: 0.3226603 (0)     total: 39m 9s   remaining: 13m
42s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232664204
bestIteration = 45

100:    loss: 0.3232664 best: 0.3226603 (0)     total: 39m 16s  remaining: 13m
13s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3233285383
```

```
bestIteration = 51

101:    loss: 0.3233285 best: 0.3226603 (0)     total: 39m 23s  remaining: 12m
44s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235121174
bestIteration = 57

102:    loss: 0.3235121 best: 0.3226603 (0)     total: 39m 30s  remaining: 12m
16s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231903231
bestIteration = 66

103:    loss: 0.3231903 best: 0.3226603 (0)     total: 39m 39s  remaining: 11m
49s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235751148
bestIteration = 26

104:    loss: 0.3235751 best: 0.3226603 (0)     total: 39m 44s  remaining: 11m
21s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3227134191
bestIteration = 140

105:    loss: 0.3227134 best: 0.3226603 (0)     total: 40m 48s  remaining: 11m
9s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.
```

```
Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231078607
bestIteration = 30

106:    loss: 0.3231079 best: 0.3226603 (0)     total: 41m 6s   remaining: 10m
45s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231089907
bestIteration = 8

107:    loss: 0.3231090 best: 0.3226603 (0)     total: 41m 16s  remaining: 10m
19s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231716647
bestIteration = 79

108:    loss: 0.3231717 best: 0.3226603 (0)     total: 41m 51s  remaining: 9m
59s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230439136
bestIteration = 66

109:    loss: 0.3230439 best: 0.3226603 (0)     total: 42m 22s  remaining: 9m
37s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232121213
bestIteration = 12

110:    loss: 0.3232121 best: 0.3226603 (0)     total: 42m 33s  remaining: 9m
12s
```

```
Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3227871207
bestIteration = 93

111:    loss: 0.3227871 best: 0.3226603 (0)     total: 43m 12s  remaining: 8m
52s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232131631
bestIteration = 35

112:    loss: 0.3232132 best: 0.3226603 (0)     total: 43m 34s  remaining: 8m
29s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232831229
bestIteration = 12

113:    loss: 0.3232831 best: 0.3226603 (0)     total: 43m 47s  remaining: 8m 3s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3233319789
bestIteration = 58

114:    loss: 0.3233320 best: 0.3226603 (0)     total: 44m 15s  remaining: 7m
41s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231277088
bestIteration = 73
```

```
115:    loss: 0.3231277 best: 0.3226603 (0)     total: 44m 47s  remaining: 7m
20s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232521386
bestIteration = 13

116:    loss: 0.3232521 best: 0.3226603 (0)     total: 44m 59s  remaining: 6m
55s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231605364
bestIteration = 119

117:    loss: 0.3231605 best: 0.3226603 (0)     total: 45m 52s  remaining: 6m
36s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3236634193
bestIteration = 36

118:    loss: 0.3236634 best: 0.3226603 (0)     total: 46m 12s  remaining: 6m
12s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235110817
bestIteration = 50

119:    loss: 0.3235111 best: 0.3226603 (0)     total: 46m 36s  remaining: 5m
49s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235360582
```

```
bestIteration = 40

120:    loss: 0.3235361 best: 0.3226603 (0)     total: 47m 11s  remaining: 5m
27s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3236142722
bestIteration = 15

121:    loss: 0.3236143 best: 0.3226603 (0)     total: 47m 32s  remaining: 5m 3s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3239267074
bestIteration = 12

122:    loss: 0.3239267 best: 0.3226603 (0)     total: 47m 51s  remaining: 4m
40s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230916954
bestIteration = 69

123:    loss: 0.3230917 best: 0.3226603 (0)     total: 48m 39s  remaining: 4m
19s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3229143039
bestIteration = 59

124:    loss: 0.3229143 best: 0.3226603 (0)     total: 49m 21s  remaining: 3m
56s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)
```

```
bestTest = 0.323425614
bestIteration = 18

125:    loss: 0.3234256 best: 0.3226603 (0)     total: 49m 42s  remaining: 3m
33s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232016515
bestIteration = 60

126:    loss: 0.3232017 best: 0.3226603 (0)     total: 50m 28s  remaining: 3m
10s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3229396962
bestIteration = 35

127:    loss: 0.3229397 best: 0.3226603 (0)     total: 51m 1s   remaining: 2m
47s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230268767
bestIteration = 40

128:    loss: 0.3230269 best: 0.3226603 (0)     total: 51m 36s  remaining: 2m
24s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3234954548
bestIteration = 60

129:    loss: 0.3234955 best: 0.3226603 (0)     total: 52m 22s  remaining: 2m

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.
```

```
Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3235545059
bestIteration = 64

130:    loss: 0.3235545 best: 0.3226603 (0)     total: 53m 11s  remaining: 1m
37s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3230662202
bestIteration = 77

131:    loss: 0.3230662 best: 0.3226603 (0)     total: 54m 9s   remaining: 1m
13s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3234598239
bestIteration = 77

132:    loss: 0.3234598 best: 0.3226603 (0)     total: 55m 4s   remaining: 49.7s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3231765363
bestIteration = 149

133:    loss: 0.3231765 best: 0.3226603 (0)     total: 56m 37s  remaining: 25.4s

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.

Stopped by overfitting detector  (20 iterations wait)

bestTest = 0.3232707916
bestIteration = 62

134:    loss: 0.3232708 best: 0.3226603 (0)     total: 57m 25s  remaining: 0us
Estimating final quality…

Warning: Overfitting detector is active, thus evaluation metric is calculated on
every iteration. 'metric_period' is ignored for evaluation metric.
```

```
Stopped by overfitting detector  (20 iterations wait)
```

[35]:

```
grid_search_result
```

[35]:

```
{'params': {'bagging_temperature': 0,
  'depth': 8,
  'l2_leaf_reg': 1,
  'learning_rate': 0.03},
 'cv_results': defaultdict(list,
             {'iterations': [0, 50, 100],
              'test-RMSE-mean': [0.31262457277675615,
               0.30926595220322,
               0.3088373652611456],
              'test-RMSE-std': [0.017219832204078,
               0.019309654803887896,
               0.019709545510680115],
              'train-RMSE-mean': [0.31249789516746046,
               0.2977011740452338,
               0.2915985496683285],
              'train-RMSE-std': [0.004320782129691766,
               0.004648132919798574,
               0.004806073448529715]})}
```

[148]:

```
train_x.columns
```

[148]:

```
Index(['channelGrouping', 'first_ses_from_the_period_start',
       'last_ses_from_the_period_end', 'interval_dates',
       'visitStartTime_counts', 'visitNumber_max', 'browser',
       'operatingSystem', 'isMobile', 'deviceCategory', 'continent',
       'subContinent', 'country', 'region', 'metro', 'city', 'networkDomain',
       'bounces_mean', 'newVisits', 'hits_sum', 'hits_min', 'hits_max',
       'hits_mean', 'pageviews_sum', 'pageviews_min', 'pageviews_max',
       'pageviews_mean', 'sessionQualityDimMin', 'sessionQualityDimMax',
       'sessionQualityDimMean', 'sessionQualityDimSum', 'timeOnSite_sum',
       'timeOnSite_min', 'timeOnSite_max', 'timeOnSite_mean', 'transactions',
       'campaign', 'source', 'medium', 'keyword', 'googleAds', 'referralPath',
       'isTrueDirect', 'customDimensions_value'],
      dtype='object')
```

[160]:

```
from catboost import CatBoostRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

def rmse(y_true, y_pred):
    return round(np.sqrt(mean_squared_error(y_true, y_pred)), 5)
```

```
X_train, X_validation, y_train, y_validation = train_test_split(train_x,␣
 ↪train_y, test_size=0.15, random_state=1)

clf = CatBoostRegressor(iterations = 1000,
                        learning_rate = 0.03,
                        depth = 8,
                        l2_leaf_reg = 1,
                        eval_metric='RMSE',
                        od_wait = 10)

clf.fit(X_train, y_train,
        eval_set = (X_validation, y_validation),
        use_best_model = True,
        verbose=True)

y_pred_train = clf.predict(X_train)
y_pred_validation = clf.predict(X_validation)
y_pred_test = clf.predict(test_x)

print(f"CatB: RMSE val: {rmse(y_validation, y_pred_validation)}  - RMSE train:␣
 ↪{rmse(y_train, y_pred_train)}")

x = pd.DataFrame(data ={'fullVisitorId': test['fullVisitorId'].astype('str'),␣
 ↪'PredictedLogRevenue': y_pred_test})
x.to_csv('results/submission_0521_catboost_WithValidationSet.csv', index =␣
 ↪False)
```

```
0:      learn: 0.3127711        test: 0.2967622 best: 0.2967622 (0)     total:
121ms    remaining: 2m
1:      learn: 0.3122908        test: 0.2967424 best: 0.2967424 (1)     total:
283ms    remaining: 2m 21s
2:      learn: 0.3118323        test: 0.2967267 best: 0.2967267 (2)     total:
422ms    remaining: 2m 20s
3:      learn: 0.3113568        test: 0.2964765 best: 0.2964765 (3)     total:
553ms    remaining: 2m 17s
4:      learn: 0.3109081        test: 0.2963568 best: 0.2963568 (4)     total:
691ms    remaining: 2m 17s
5:      learn: 0.3103888        test: 0.2960738 best: 0.2960738 (5)     total:
857ms    remaining: 2m 21s
6:      learn: 0.3099586        test: 0.2959168 best: 0.2959168 (6)     total:
995ms    remaining: 2m 21s
7:      learn: 0.3095630        test: 0.2959104 best: 0.2959104 (7)     total:
1.13s    remaining: 2m 20s
8:      learn: 0.3091638        test: 0.2958102 best: 0.2958102 (8)     total:
1.26s    remaining: 2m 19s
9:      learn: 0.3088096        test: 0.2957350 best: 0.2957350 (9)     total:
1.39s    remaining: 2m 18s
```

```
10:     learn: 0.3084326        test: 0.2956306 best: 0.2956306 (10)    total:
1.53s    remaining: 2m 17s
11:     learn: 0.3080684        test: 0.2955228 best: 0.2955228 (11)    total:
1.66s    remaining: 2m 16s
12:     learn: 0.3077153        test: 0.2955180 best: 0.2955180 (12)    total:
1.82s    remaining: 2m 18s
13:     learn: 0.3073688        test: 0.2954095 best: 0.2954095 (13)    total:
1.99s    remaining: 2m 20s
14:     learn: 0.3070580        test: 0.2953325 best: 0.2953325 (14)    total:
2.12s    remaining: 2m 18s
15:     learn: 0.3067270        test: 0.2952020 best: 0.2952020 (15)    total:
2.25s    remaining: 2m 18s
16:     learn: 0.3064449        test: 0.2950634 best: 0.2950634 (16)    total:
2.4s     remaining: 2m 18s
17:     learn: 0.3061203        test: 0.2949483 best: 0.2949483 (17)    total:
2.54s    remaining: 2m 18s
18:     learn: 0.3057528        test: 0.2947362 best: 0.2947362 (18)    total:
2.7s     remaining: 2m 19s
19:     learn: 0.3054889        test: 0.2946671 best: 0.2946671 (19)    total:
2.86s    remaining: 2m 20s
20:     learn: 0.3051916        test: 0.2946568 best: 0.2946568 (20)    total:
3.01s    remaining: 2m 20s
21:     learn: 0.3049214        test: 0.2946794 best: 0.2946568 (20)    total:
3.17s    remaining: 2m 20s
22:     learn: 0.3046568        test: 0.2946010 best: 0.2946010 (22)    total:
3.28s    remaining: 2m 19s
23:     learn: 0.3044311        test: 0.2946190 best: 0.2946010 (22)    total:
3.43s    remaining: 2m 19s
24:     learn: 0.3041141        test: 0.2944244 best: 0.2944244 (24)    total:
3.61s    remaining: 2m 20s
25:     learn: 0.3038817        test: 0.2942517 best: 0.2942517 (25)    total:
3.74s    remaining: 2m 20s
26:     learn: 0.3036096        test: 0.2940639 best: 0.2940639 (26)    total:
3.89s    remaining: 2m 20s
27:     learn: 0.3033526        test: 0.2939416 best: 0.2939416 (27)    total:
4.03s    remaining: 2m 19s
28:     learn: 0.3031060        test: 0.2938486 best: 0.2938486 (28)    total:
4.17s    remaining: 2m 19s
29:     learn: 0.3028514        test: 0.2938045 best: 0.2938045 (29)    total:
4.31s    remaining: 2m 19s
30:     learn: 0.3026264        test: 0.2936629 best: 0.2936629 (30)    total:
4.45s    remaining: 2m 18s
31:     learn: 0.3023872        test: 0.2935452 best: 0.2935452 (31)    total:
4.57s    remaining: 2m 18s
32:     learn: 0.3021760        test: 0.2934868 best: 0.2934868 (32)    total:
4.69s    remaining: 2m 17s
33:     learn: 0.3019646        test: 0.2934973 best: 0.2934868 (32)    total:
4.82s    remaining: 2m 17s
```

```
34:     learn: 0.3017654        test: 0.2934398 best: 0.2934398 (34)    total:
4.99s    remaining: 2m 17s
35:     learn: 0.3015687        test: 0.2933996 best: 0.2933996 (35)    total:
5.12s    remaining: 2m 17s
36:     learn: 0.3013569        test: 0.2933183 best: 0.2933183 (36)    total:
5.27s    remaining: 2m 17s
37:     learn: 0.3011526        test: 0.2932038 best: 0.2932038 (37)    total:
5.38s    remaining: 2m 16s
38:     learn: 0.3009620        test: 0.2931186 best: 0.2931186 (38)    total:
5.5s     remaining: 2m 15s
39:     learn: 0.3007934        test: 0.2930475 best: 0.2930475 (39)    total:
5.64s    remaining: 2m 15s
40:     learn: 0.3006062        test: 0.2928955 best: 0.2928955 (40)    total:
5.78s    remaining: 2m 15s
41:     learn: 0.3004381        test: 0.2927306 best: 0.2927306 (41)    total:
5.92s    remaining: 2m 14s
42:     learn: 0.3002104        test: 0.2925911 best: 0.2925911 (42)    total:
6.07s    remaining: 2m 15s
43:     learn: 0.3000548        test: 0.2925720 best: 0.2925720 (43)    total:
6.21s    remaining: 2m 15s
44:     learn: 0.2998336        test: 0.2924029 best: 0.2924029 (44)    total:
6.35s    remaining: 2m 14s
45:     learn: 0.2996222        test: 0.2923076 best: 0.2923076 (45)    total:
6.48s    remaining: 2m 14s
46:     learn: 0.2994738        test: 0.2922919 best: 0.2922919 (46)    total:
6.59s    remaining: 2m 13s
47:     learn: 0.2993424        test: 0.2923091 best: 0.2922919 (46)    total:
6.73s    remaining: 2m 13s
48:     learn: 0.2992105        test: 0.2923279 best: 0.2922919 (46)    total:
6.86s    remaining: 2m 13s
49:     learn: 0.2990130        test: 0.2921925 best: 0.2921925 (49)    total:
6.99s    remaining: 2m 12s
50:     learn: 0.2988837        test: 0.2921878 best: 0.2921878 (50)    total:
7.11s    remaining: 2m 12s
51:     learn: 0.2987519        test: 0.2921381 best: 0.2921381 (51)    total:
7.23s    remaining: 2m 11s
52:     learn: 0.2986031        test: 0.2920852 best: 0.2920852 (52)    total:
7.35s    remaining: 2m 11s
53:     learn: 0.2984409        test: 0.2919753 best: 0.2919753 (53)    total:
7.48s    remaining: 2m 11s
54:     learn: 0.2982714        test: 0.2917510 best: 0.2917510 (54)    total:
7.59s    remaining: 2m 10s
55:     learn: 0.2981522        test: 0.2917338 best: 0.2917338 (55)    total:
7.72s    remaining: 2m 10s
56:     learn: 0.2980578        test: 0.2917064 best: 0.2917064 (56)    total:
7.84s    remaining: 2m 9s
57:     learn: 0.2978950        test: 0.2916422 best: 0.2916422 (57)    total:
7.99s    remaining: 2m 9s
```

```
58:     learn: 0.2977583        test: 0.2916124 best: 0.2916124 (58)    total:
8.18s    remaining: 2m 10s
59:     learn: 0.2976611        test: 0.2915541 best: 0.2915541 (59)    total:
8.32s    remaining: 2m 10s
60:     learn: 0.2974485        test: 0.2914671 best: 0.2914671 (60)    total:
8.49s    remaining: 2m 10s
61:     learn: 0.2972941        test: 0.2913840 best: 0.2913840 (61)    total:
8.62s    remaining: 2m 10s
62:     learn: 0.2971515        test: 0.2912467 best: 0.2912467 (62)    total:
8.73s    remaining: 2m 9s
63:     learn: 0.2969783        test: 0.2911679 best: 0.2911679 (63)    total:
8.88s    remaining: 2m 9s
64:     learn: 0.2968335        test: 0.2909759 best: 0.2909759 (64)    total:
9s       remaining: 2m 9s
65:     learn: 0.2967502        test: 0.2909750 best: 0.2909750 (65)    total:
9.14s    remaining: 2m 9s
66:     learn: 0.2966012        test: 0.2909232 best: 0.2909232 (66)    total:
9.3s     remaining: 2m 9s
67:     learn: 0.2965230        test: 0.2908912 best: 0.2908912 (67)    total:
9.41s    remaining: 2m 9s
68:     learn: 0.2964373        test: 0.2908454 best: 0.2908454 (68)    total:
9.53s    remaining: 2m 8s
69:     learn: 0.2962954        test: 0.2907749 best: 0.2907749 (69)    total:
9.66s    remaining: 2m 8s
70:     learn: 0.2961887        test: 0.2907415 best: 0.2907415 (70)    total:
9.82s    remaining: 2m 8s
71:     learn: 0.2960745        test: 0.2907720 best: 0.2907415 (70)    total:
9.98s    remaining: 2m 8s
72:     learn: 0.2959810        test: 0.2907487 best: 0.2907415 (70)    total:
10.2s    remaining: 2m 8s
73:     learn: 0.2958861        test: 0.2907264 best: 0.2907264 (73)    total:
10.3s    remaining: 2m 9s
74:     learn: 0.2958059        test: 0.2906966 best: 0.2906966 (74)    total:
10.4s    remaining: 2m 8s
75:     learn: 0.2957044        test: 0.2906591 best: 0.2906591 (75)    total:
10.6s    remaining: 2m 8s
76:     learn: 0.2956258        test: 0.2906614 best: 0.2906591 (75)    total:
10.7s    remaining: 2m 8s
77:     learn: 0.2954929        test: 0.2906196 best: 0.2906196 (77)    total:
10.9s    remaining: 2m 8s
78:     learn: 0.2954026        test: 0.2905882 best: 0.2905882 (78)    total:
11s      remaining: 2m 8s
79:     learn: 0.2953214        test: 0.2906049 best: 0.2905882 (78)    total:
11.2s    remaining: 2m 8s
80:     learn: 0.2952384        test: 0.2905989 best: 0.2905882 (78)    total:
11.3s    remaining: 2m 8s
81:     learn: 0.2951404        test: 0.2906005 best: 0.2905882 (78)    total:
11.5s    remaining: 2m 8s
```

```
82:     learn: 0.2950533        test: 0.2905598 best: 0.2905598 (82)    total:
11.6s    remaining: 2m 8s
83:     learn: 0.2949415        test: 0.2904800 best: 0.2904800 (83)    total:
11.7s    remaining: 2m 8s
84:     learn: 0.2948378        test: 0.2904993 best: 0.2904800 (83)    total:
11.9s    remaining: 2m 8s
85:     learn: 0.2947544        test: 0.2904795 best: 0.2904795 (85)    total:
12.1s    remaining: 2m 8s
86:     learn: 0.2946542        test: 0.2904384 best: 0.2904384 (86)    total:
12.2s    remaining: 2m 7s
87:     learn: 0.2945843        test: 0.2904299 best: 0.2904299 (87)    total:
12.3s    remaining: 2m 7s
88:     learn: 0.2945198        test: 0.2903919 best: 0.2903919 (88)    total:
12.5s    remaining: 2m 7s
89:     learn: 0.2943937        test: 0.2903826 best: 0.2903826 (89)    total:
12.7s    remaining: 2m 7s
90:     learn: 0.2943327        test: 0.2903657 best: 0.2903657 (90)    total:
12.8s    remaining: 2m 7s
91:     learn: 0.2942299        test: 0.2903057 best: 0.2903057 (91)    total:
12.9s    remaining: 2m 7s
92:     learn: 0.2941245        test: 0.2903070 best: 0.2903057 (91)    total:
13s      remaining: 2m 6s
93:     learn: 0.2940563        test: 0.2902742 best: 0.2902742 (93)    total:
13.2s    remaining: 2m 6s
94:     learn: 0.2939482        test: 0.2902220 best: 0.2902220 (94)    total:
13.3s    remaining: 2m 6s
95:     learn: 0.2938662        test: 0.2902150 best: 0.2902150 (95)    total:
13.4s    remaining: 2m 6s
96:     learn: 0.2938075        test: 0.2902223 best: 0.2902150 (95)    total:
13.6s    remaining: 2m 6s
97:     learn: 0.2937185        test: 0.2901847 best: 0.2901847 (97)    total:
13.7s    remaining: 2m 6s
98:     learn: 0.2936553        test: 0.2901557 best: 0.2901557 (98)    total:
13.8s    remaining: 2m 5s
99:     learn: 0.2936033        test: 0.2901507 best: 0.2901507 (99)    total:
14s      remaining: 2m 5s
100:    learn: 0.2935215        test: 0.2901423 best: 0.2901423 (100)   total:
14.2s    remaining: 2m 6s
101:    learn: 0.2934411        test: 0.2901526 best: 0.2901423 (100)   total:
14.4s    remaining: 2m 6s
102:    learn: 0.2933195        test: 0.2901392 best: 0.2901392 (102)   total:
14.5s    remaining: 2m 6s
103:    learn: 0.2932739        test: 0.2901585 best: 0.2901392 (102)   total:
14.7s    remaining: 2m 6s
104:    learn: 0.2931998        test: 0.2901235 best: 0.2901235 (104)   total:
14.8s    remaining: 2m 6s
105:    learn: 0.2930954        test: 0.2901010 best: 0.2901010 (105)   total:
15s      remaining: 2m 6s
```

```
106:    learn: 0.2929967        test: 0.2900887 best: 0.2900887 (106)   total:
15.1s    remaining: 2m 6s
107:    learn: 0.2929051        test: 0.2900788 best: 0.2900788 (107)   total:
15.3s    remaining: 2m 6s
108:    learn: 0.2927847        test: 0.2900315 best: 0.2900315 (108)   total:
15.4s    remaining: 2m 6s
109:    learn: 0.2927070        test: 0.2899970 best: 0.2899970 (109)   total:
15.5s    remaining: 2m 5s
110:    learn: 0.2926363        test: 0.2899964 best: 0.2899964 (110)   total:
15.7s    remaining: 2m 5s
111:    learn: 0.2925007        test: 0.2899557 best: 0.2899557 (111)   total:
15.9s    remaining: 2m 6s
112:    learn: 0.2924106        test: 0.2899540 best: 0.2899540 (112)   total:
16s      remaining: 2m 5s
113:    learn: 0.2923517        test: 0.2899432 best: 0.2899432 (113)   total:
16.3s    remaining: 2m 6s
114:    learn: 0.2922447        test: 0.2899356 best: 0.2899356 (114)   total:
16.5s    remaining: 2m 6s
115:    learn: 0.2921759        test: 0.2898866 best: 0.2898866 (115)   total:
16.6s    remaining: 2m 6s
116:    learn: 0.2921047        test: 0.2898293 best: 0.2898293 (116)   total:
16.7s    remaining: 2m 6s
117:    learn: 0.2919977        test: 0.2897945 best: 0.2897945 (117)   total:
16.9s    remaining: 2m 6s
118:    learn: 0.2919352        test: 0.2897965 best: 0.2897945 (117)   total:
17s      remaining: 2m 5s
119:    learn: 0.2918274        test: 0.2897894 best: 0.2897894 (119)   total:
17.2s    remaining: 2m 6s
120:    learn: 0.2917598        test: 0.2897904 best: 0.2897894 (119)   total:
17.4s    remaining: 2m 6s
121:    learn: 0.2916630        test: 0.2898036 best: 0.2897894 (119)   total:
17.5s    remaining: 2m 5s
122:    learn: 0.2916048        test: 0.2898113 best: 0.2897894 (119)   total:
17.7s    remaining: 2m 6s
123:    learn: 0.2915374        test: 0.2898103 best: 0.2897894 (119)   total:
17.8s    remaining: 2m 5s
124:    learn: 0.2915051        test: 0.2898113 best: 0.2897894 (119)   total:
17.9s    remaining: 2m 5s
125:    learn: 0.2914528        test: 0.2898075 best: 0.2897894 (119)   total:
18.1s    remaining: 2m 5s
126:    learn: 0.2914237        test: 0.2897897 best: 0.2897894 (119)   total:
18.4s    remaining: 2m 6s
127:    learn: 0.2913136        test: 0.2897165 best: 0.2897165 (127)   total:
18.5s    remaining: 2m 6s
128:    learn: 0.2911707        test: 0.2897031 best: 0.2897031 (128)   total:
18.7s    remaining: 2m 6s
129:    learn: 0.2910897        test: 0.2897105 best: 0.2897031 (128)   total:
18.9s    remaining: 2m 6s
```

```
130:    learn: 0.2910320        test: 0.2897114 best: 0.2897031 (128)   total:
19s      remaining: 2m 6s
131:    learn: 0.2909663        test: 0.2897231 best: 0.2897031 (128)   total:
19.2s    remaining: 2m 6s
132:    learn: 0.2909026        test: 0.2897176 best: 0.2897031 (128)   total:
19.3s    remaining: 2m 5s
133:    learn: 0.2908229        test: 0.2897311 best: 0.2897031 (128)   total:
19.4s    remaining: 2m 5s
134:    learn: 0.2907036        test: 0.2896958 best: 0.2896958 (134)   total:
19.6s    remaining: 2m 5s
135:    learn: 0.2906552        test: 0.2896949 best: 0.2896949 (135)   total:
19.8s    remaining: 2m 5s
136:    learn: 0.2906009        test: 0.2896855 best: 0.2896855 (136)   total:
19.9s    remaining: 2m 5s
137:    learn: 0.2905493        test: 0.2896220 best: 0.2896220 (137)   total:
20.1s    remaining: 2m 5s
138:    learn: 0.2904793        test: 0.2896204 best: 0.2896204 (138)   total:
20.2s    remaining: 2m 5s
139:    learn: 0.2904283        test: 0.2896024 best: 0.2896024 (139)   total:
20.3s    remaining: 2m 4s
140:    learn: 0.2903415        test: 0.2895959 best: 0.2895959 (140)   total:
20.5s    remaining: 2m 5s
141:    learn: 0.2902821        test: 0.2895622 best: 0.2895622 (141)   total:
20.7s    remaining: 2m 5s
142:    learn: 0.2902244        test: 0.2895209 best: 0.2895209 (142)   total:
20.8s    remaining: 2m 4s
143:    learn: 0.2901283        test: 0.2895367 best: 0.2895209 (142)   total:
21s      remaining: 2m 5s
144:    learn: 0.2900279        test: 0.2895389 best: 0.2895209 (142)   total:
21.2s    remaining: 2m 4s
145:    learn: 0.2899306        test: 0.2895478 best: 0.2895209 (142)   total:
21.3s    remaining: 2m 4s
146:    learn: 0.2898806        test: 0.2895490 best: 0.2895209 (142)   total:
21.5s    remaining: 2m 4s
147:    learn: 0.2898224        test: 0.2895545 best: 0.2895209 (142)   total:
21.6s    remaining: 2m 4s
148:    learn: 0.2897727        test: 0.2895631 best: 0.2895209 (142)   total:
21.8s    remaining: 2m 4s
149:    learn: 0.2897189        test: 0.2895715 best: 0.2895209 (142)   total:
21.9s    remaining: 2m 4s
150:    learn: 0.2896667        test: 0.2895626 best: 0.2895209 (142)   total:
22.1s    remaining: 2m 4s
151:    learn: 0.2895926        test: 0.2895558 best: 0.2895209 (142)   total:
22.3s    remaining: 2m 4s
152:    learn: 0.2895429        test: 0.2895568 best: 0.2895209 (142)   total:
22.4s    remaining: 2m 4s
Stopped by overfitting detector  (10 iterations wait)
```

```
bestTest = 0.2895209322
bestIteration = 142

Shrink model to first 143 iterations.
CatB: RMSE val: 0.28952  - RMSE train: 0.29022
```

[155]:
```
clf.get_feature_importance()
```

[155]:
```
array([ 3.56728406, 11.09210622,  8.44488576, 14.68203887,  1.26667715,
        1.92006298,  2.78682327,  0.13408075,  1.54573195,  1.06355022,
        1.29918452,  2.29794529,  2.9234394 ,  1.62685563,  2.89517628,
        0.89106466,  6.02650912,  1.7348324 ,  2.01413553,  4.38580814,
        0.60534303,  2.14313691,  1.554905  ,  0.85718245,  2.93245607,
        0.4177713 , 17.1580181 ,  0.06110598,  0.87427786,  0.35509214,
        0.44251895])
```

[ ]:
```
"""
import catboost as cb

clf = cb.CatBoostRegressor(iterations=1000,
                             learning_rate=0.05,
                             depth=10,
                             eval_metric='RMSE',
                             random_seed = 42,
                             bagging_temperature = 0.2,
                             od_type='Iter',
                             metric_period = 50,
                             od_wait=20)

clf.fit(train_x,
        train_y,
        verbose=True)

y_pred_train = clf.predict(train_x)
y_pred_test = clf.predict(test_x)

x = pd.DataFrame(data ={'fullVisitorId': test['fullVisitorId'].astype('str'),␣
 ↪'PredictedLogRevenue': y_pred_test})
x.to_csv('results/submission_0521_catboost_tune.csv', index = False)
"""

"""
from catboost import CatBoostRegressor

clf = CatBoostRegressor()

clf.fit(train_x, train_y)
```

```
y_pred_train = clf.predict(train_x)
y_pred_test = clf.predict(test_x)

x = pd.DataFrame(data ={'fullVisitorId': test['fullVisitorId'].astype('str'),␣
 ↪'PredictedLogRevenue': y_pred_test})
x.to_csv('submission_0519_catboost_withoutTune.csv', index = False)
"""
```

[ ]:

## 5. Conclusion

[181]:

```
#https://zhuanlan.zhihu.com/p/50525264
```

- There is only 1.08% completed transaction in dataset. Almost 99% of visitor just visit the GStore without purchase anything.
- Trabsaction occur frequently during weekdays.
- Chrome browser is the most favor browser used to surt GStore.
- Most of the users/visitors are come from United State.

## 6. References

Plotly library to visualize data in map: >- Choropleth Maps in Python - Continuous Color Scales and Color Bars in Python - Quick Start: Creating a US State Choropleth Map with Python Plotly - Choropleth Map with Plotly

Geopandas library to visualize data in map: >- Creating a GeoDataFrame from a DataFrame with coordinates - A Beginners Guide to Create a Cloropleth Map in Python using GeoPandas and Matplotlib - Using GeoPandas to Display Data in Spatial Context - Let's make a map! Using Geopandas, Pandas and Matplotlib to make a Choropleth map - Creating a Choropleth Map of the World in Python using GeoPandas

Pandas library: >- pandas.cut - Group By: split-apply-combine

Related Work: The data we use is from competation in Kaggle, there are some related work we take as reference in our project.

1.) The winning solution of the competation - The is a research done in r language - The research propose a unique feature engenering that seperate the date base on time-range: a period of 168 days follow by a gap of 46 days to optimize the prediction - The research propose a hurdel model (classification then regression) to solve the prediction problem

2.) GOOGLE ANALYTICS CUSTOMER REVENUE PREDICTION by Nikhil Bokade (36th place solutio nof the competation) - This research discuss in detail on the data preprocessing & feature engenerring for the model

3.) Gstore Analysis from H.BO's Data Analysis - In this blog, the author suggest an alternative way to eplore the dataset

[ ]: