Google Analytics Customer Revenue Prediction

-- Yu Liu

Abstract

In real word, the 80/20 rule has proven true for many businesses–only a small percentage of customers produce most of the revenue. So it is difficult for marketing teams to make appropriate investments in promotional strategies. This project is trying to analyze a Google Merchandise Store (also known as GStore, where Google swag is sold) customer dataset, which contains every customers' every view information from August 1st, 2016 to May 1st, 2018, extract useful information and sue some models to predict how much each customer will spend money on Gstore from December 1st, 2018 to January 31st, 2019.

Data explore and munging

The dataset comes from Kaggle competitions, the train data set collects view information of every Gstore customers and their cost from August 1st, 2016 to May 1st, 2018, and test data set collects view information of every Gstore customers from December 1st, 2018 to January 31st, 2019, and the target is to predict how much they will spend on GStore during this period. The main features of data are:

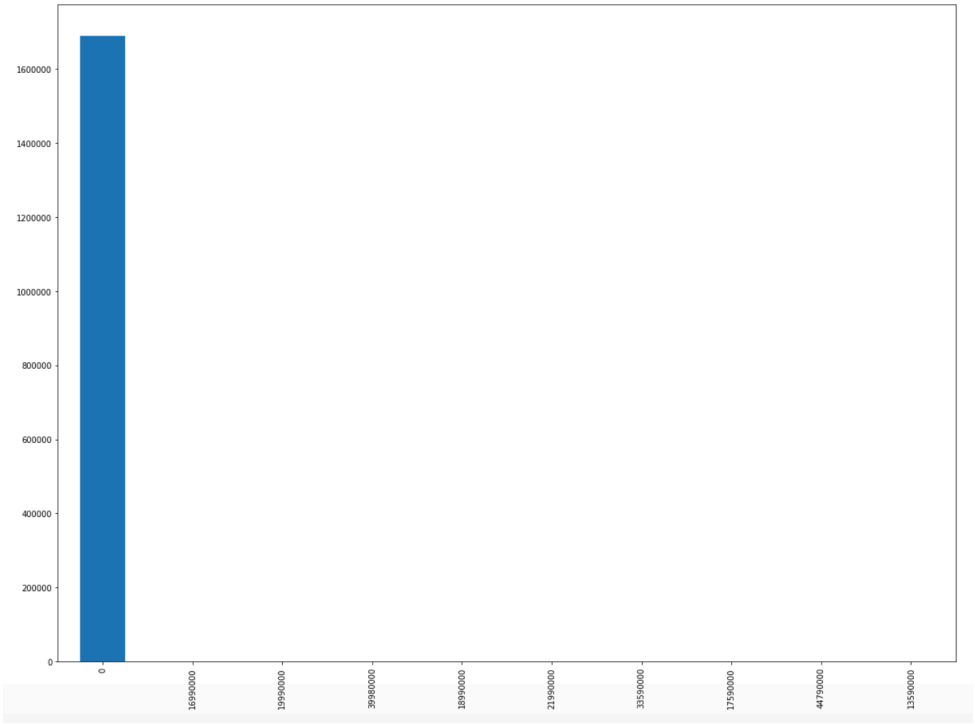
- fullVisitorId- A unique identifier for each user of the Google Merchandise Store.
- channelGrouping - The channel via which the user came to the Store.
- date - The date on which the user visited the Store.
- device - The specifications for the device used to access the Store.
- geoNetwork - This section contains information about the geography of the user.
- socialEngagementType - Engagement type, either "Socially Engaged" or "Not Socially Engaged".

- totals - This section contains aggregate values across the session.
- trafficSource - This section contains information about the Traffic Source from which the session originated.
- visitId - An identifier for this session. This is part of the value usually stored as the _utmb cookie. This is only unique to the user. For a completely unique ID, you should use a combination of fullVisitorId and visitId.
- visitNumber - The session number for this user. If this is the first session, then this is set to 1.
- visitStartTime - The timestamp (expressed as POSIX time).
- hits - This row and nested fields are populated for any and all types of hits. Provides a record of all page visits.
- customDimensions - This section contains any user-level or session-level custom dimensions that are set for a session. This is a repeated field and has an entry for each dimension that is set.
- totals - This set of columns mostly includes high-level aggregate data.

As the dataset is very large, I try to use python's Garbage collection package [1] and read csv file in small chunks to reduce the memory when loading the data.

First, I changed POSIX timestamp in visitStartTime column to readable time format, and use it create other time columns like day, month, year, the day of week. And then, I flatten all json columns, and got 60 columns and 1,708,337 rows. In one of those JSON columns, totals, the sub-column transactionRevenue contains the revenue information we are trying to predict. I plot the histogram of top 10 values, as we see 0 is such much, and we can't see other 9 values. The 98.9% of value we will predict is NaN, which means this customer didn't spend money in this view, so I cnage this NaN value to 0.

As Project's Overview said, "The 80/20 rule has proven true for many businesses–only a small percentage of customers produce most of the revenue", the percentage of customers produce most of the revenue is really low.



Histogram of target value

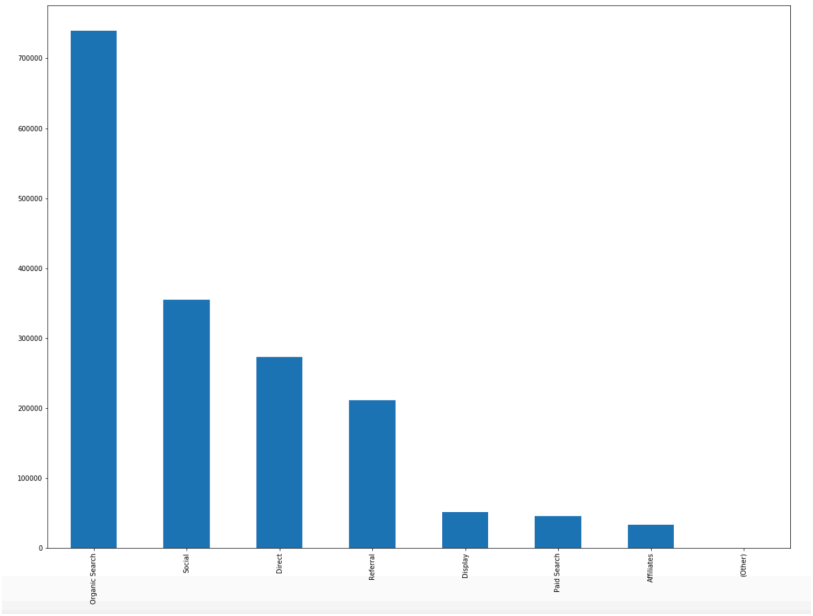
There are some columns with consistent values and these columns are useless when predict.

```
for col in cols:
    if len(train_df[col].value_counts())==1:
        print(col)
```

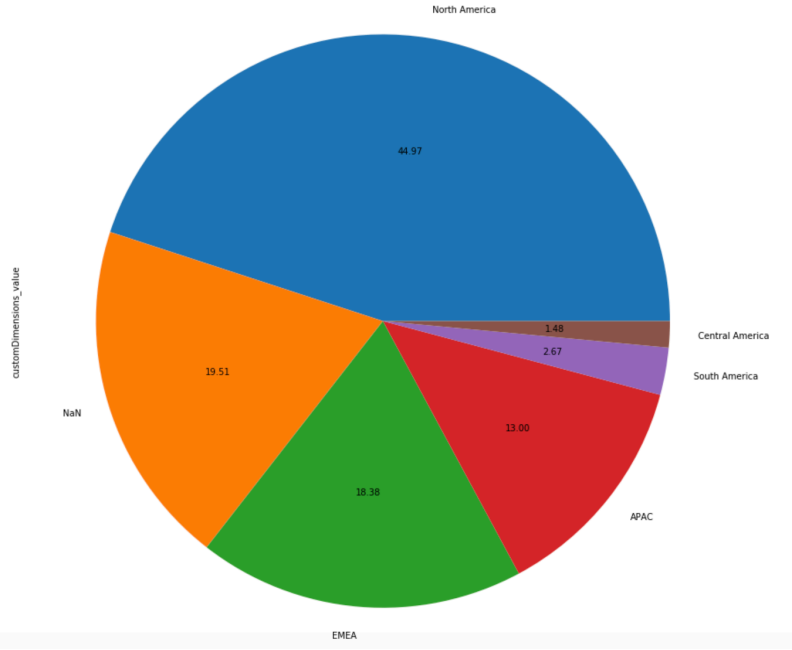
```
socialEngagementType
device_browserSize
device_browserVersion
device_flashVersion
device_language
device_mobileDeviceBranding
device_mobileDeviceInfo
device_mobileDeviceMarketingName
device_mobileDeviceModel
device_mobileInputSelector
device_operatingSystemVersion
device_screenColors
device_screenResolution
geoNetwork_cityId
geoNetwork_latitude
geoNetwork_longitude
geoNetwork_networkLocation
totals_bounces
totals_newVisits
totals_visits
trafficSource_adwordsClickInfo.criteriaParameters
trafficSource_adwordsClickInfo.isVideoAd
trafficSource_campaignCode
trafficSource_isTrueDirect
```

Columns with consistent values

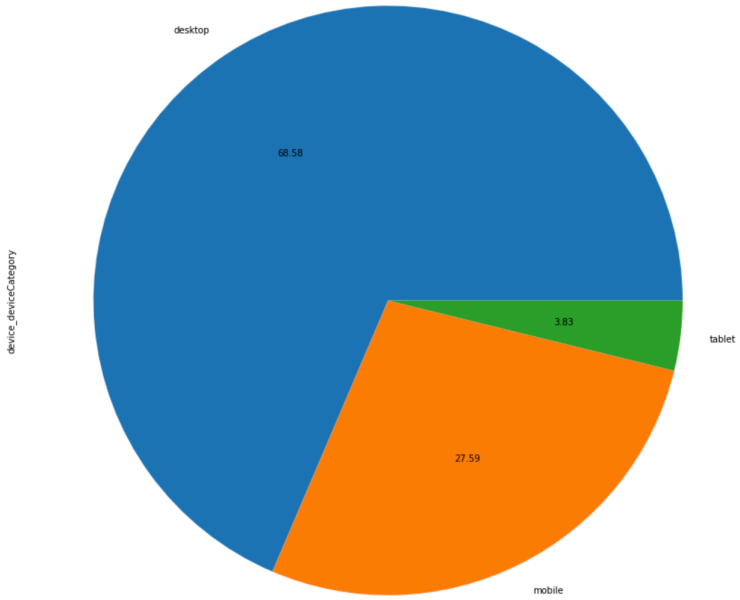
To get useful columns I explored all columns in the data set. In sub-columns of hits json column, it looks like all information of columns in hits chunk is already contained in other columns. For example, we can find information of dataSource column in trafficSource_referralPath column, information of hitNumber in totals_hits column. And the last but not least reason, this chunk will take really much of time to load and flatten, so I drop this chunk in future analysis and prediction. For convenient and efficiency, we decided to use 35 columns from original dataset, and create 4 columns myself. There are some plots of typical columns.



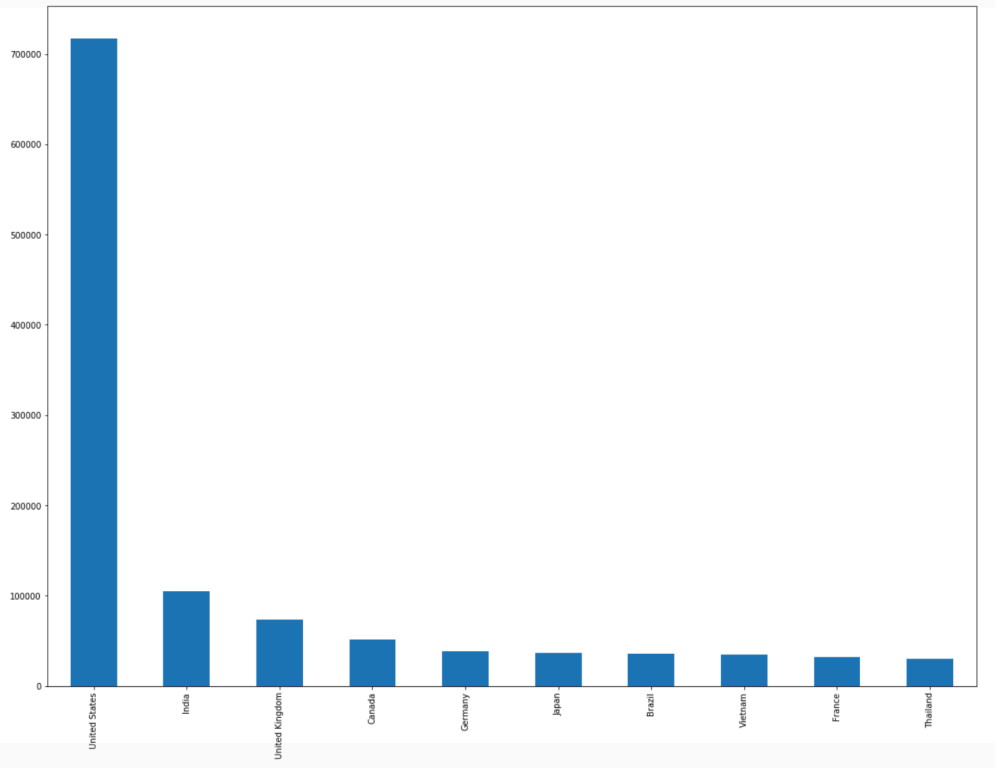
This histogram plot is about how the customer comes to the store page. We can see most of them are via Organic Search.



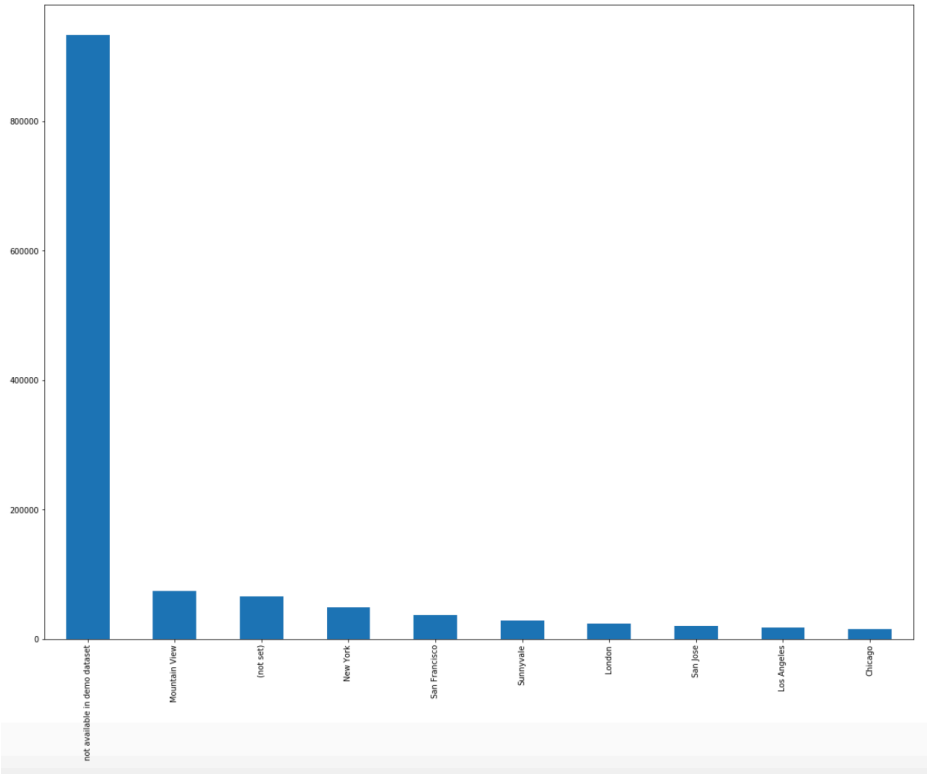
From CustomerDimensions_value column, we can see 44.97% GStore customers are from North America.



From device_deviceCategory column, we can see 68.58% devices are desktop, 27.59% are mobile, and 3.83% are tablet. As this dataset is from 2016-08-01 to 2018-05-01, the result may be different now.



In the histogram of top 10 country, we can see mainly customers are in United States, and the second country is India.



In the histogram of top 10 city, nearly 90% data are not allowed to see or not set, and except these data, top 1 city is Mountain View, followed by New York and San Francisco. This data

set not provide longitude and latitude because of customers' privacy, if we can get more information, I think the prediction will be better.

After getting all useful columns, except the target value there are 11 columns have NaN values, fill them with different ways according to the columns theirself.

```
big_df['totals_pageviews'].fillna(1, inplace=True)
big_df['totals_newVisits'].fillna(0, inplace=True)
big_df['totals_sessionQualityDim'].fillna(0, inplace=True)
big_df['totals_bounces'].fillna(0, inplace=True)
big_df['totals_timeOnSite'].fillna(0, inplace=True)
big_df['totals_transactions'].fillna(0, inplace=True)
```

```
big_df['trafficSource_keyword'].fillna('(not provided)', inplace=True)
big_df['trafficSource_referralPath'].fillna('/', inplace=True)
big_df['trafficSource_isTrueDirect'].fillna(False, inplace=True)
big_df['trafficSource_adContent'].fillna('unknown', inplace=True)
big_df['trafficSource_adwordsClickInfo.page'].fillna(0, inplace=True)
```

I also used LabelEncoder function in sklearn library to encode object values in object columns. And the last thing of data munging, the target is to predict every customer revenue in the Gstore, and row in the datasets is the record of every view of every customer. So we should combine all views belong to same customer (same fullVisitorId). When merge rows with same fullVisitorId, we need to decide what to do with columns. I use some diferent ways to different columns to get better result.

```
to_median = ['channelGrouping', 'device_browser', 'device_deviceCategory', 'device_isMobile',
             'device_operatingSystem', 'geoNetwork_city', 'geoNetwork_continent', 'geoNetwork_country'
             'geoNetwork_metro', 'geoNetwork_networkDomain', 'geoNetwork_region',
             'geoNetwork_subContinent', 'trafficSource_adContent',
             'trafficSource_adwordsClickInfo.page', 'trafficSource_campaign',
             'trafficSource_isTrueDirect', 'trafficSource_keyword', 'trafficSource_medium',
             'trafficSource_referralPath', 'trafficSource_source', 'customDimensions_index',
             'customDimensions_value', 'year', 'month', 'day', 'weekday']
to_sum = ['visitNumber', 'totals_bounces', 'totals_hits', 'totals_newVisits', 'totals_pageviews',
          'totals_timeOnSite', 'totals_transactions']
to_mean = ['totals_bounces', 'totals_hits', 'totals_newVisits', 'totals_pageviews',
           'totals_sessionQualityDim', 'totals_timeOnSite']
to_std = ['totals_bounces', 'totals_hits', 'totals_pageviews', 'totals_timeOnSite']
```

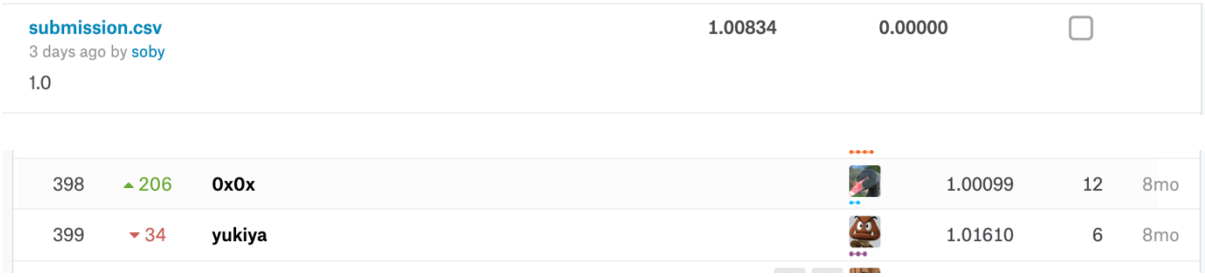
Like channelGrouping and device_browser columns, get their median value is enough, and for totals_bounces and some other columns I get their sum, mean and standard deviation.

Prediction

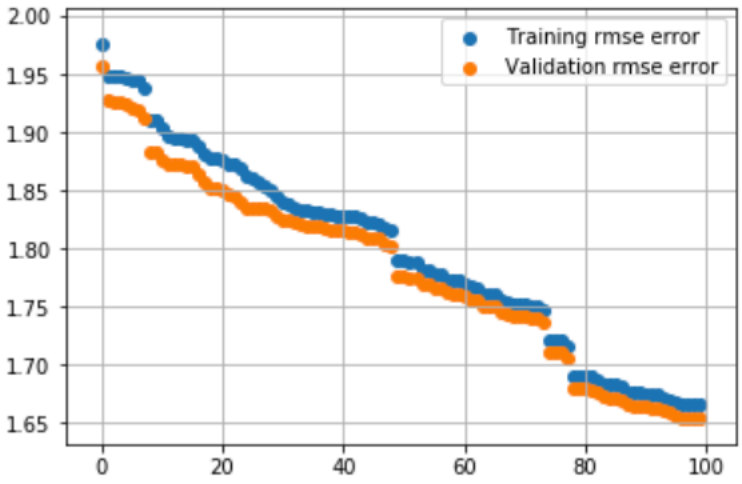
We are predicting the natural log of the sum of all transactions per user. For every user in the test set, the target is: $target_{user} = \ln(y_{user} + 1)$

As what we see in Data explore part, the 98.9% of value we will predict is NaN, it we use decision tree to split data in to NaN and not NaN for some features, and continue to predict the value may be a good idea, so model like random forest and xgboost may have good performance. And as experience, the XGBoost [2] perform very well in machine learning. So I try to use this model to predict first.

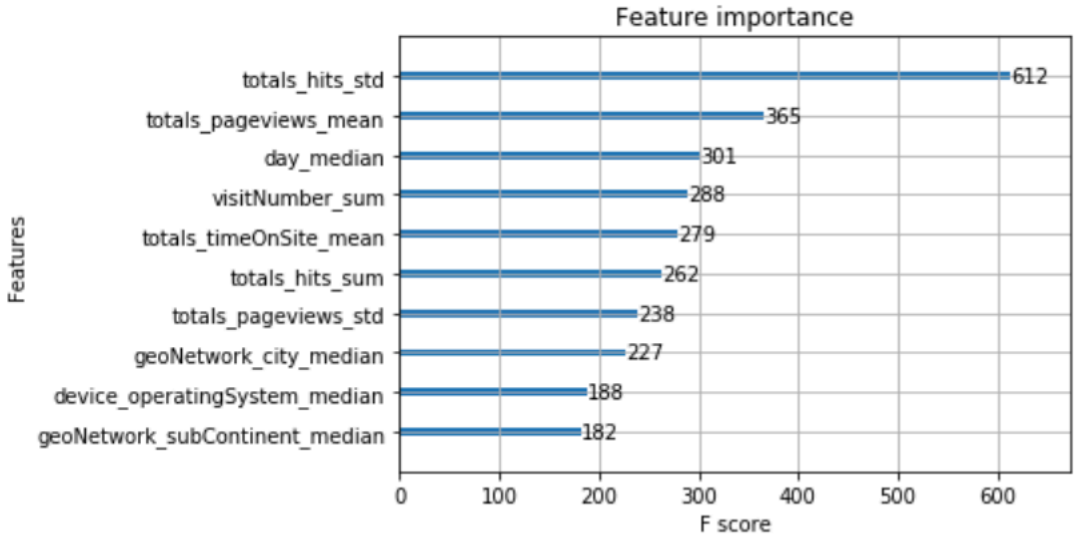
I tried to use GridSearchCV to tune parameters, but because the data set is too large and the power of my laptop is not good enough, the auto tuning process will take much time, so I only manual tuned parameters. And after several times, I got the best score in Kaggle is 1.00834, and the rank of it is about 400.



Because this competition is closed, so I can't publish my score.



From the the root-mean-square-error plot we can see, as the iteration increase, both Training rmse and Validation rmse are becoming smaller, and Training rmse is getting colser to Validation rmse, but with iteration increase further, may overfitting, so I choose the number of iteration is 100.



From the feature importance plot, we can see the totals_hits feature is most important, which means the number of total hits this customer hit in Gstore. The second important feature is totals_pageviews, which means how many pages this customer view in Gstore. The day of month is third important, which seems that the day of month the customer view Gstore may realted to the customer' revenue. And the customers' geo graph is also important.

As I don't use the hits columns, and I discard some columns look like useless (truth may be not), the prediction result is not good enough. The methods I group the same ID can also be improved to get a better result.

Conclusion

I also tried some other models, like LightGBM [3], but their result are not better than XGBoost. And as I don't use the hits columns, and I discard some columns look like useless (truth may be not), the prediction result is not good enough. The methods I group the same ID

can also be improved to get a better result. I think the score I get can be further improved, if I continue tune parameters or with better data mining process, but the cost is time. It's a pity that I don't have more time to continue, hope I can review this project in the future.

Citation

[1]. Garbage Collection package From (https://rushter.com/blog/python-garbage-collector/)

[2]. XGBoost Documentation From (https://xgboost.readthedocs.io/en/latest/)

[3]. LightGBM Documentation From (https://lightgbm.readthedocs.io/en/latest/)

[4]. Business Model Analysis for Online Social Shopping Companies Case Study: RunToShop Oy From (http://epub.lib.aalto.fi/en/ethesis/pdf/12190/hse_ethesis_12190.pdf)