CSC 869

Data Mining

Spring 2019

SAN FRANCISCO STATE UNIVERSITY

SAN FRANCISCO, CA

Term Project

ELO Merchant Category Recommendation

Project Report

Submitted By:

Shan Kwan Cho

**Contents**

|  |  |
| --- | --- |
| 1. | Instructions on compiling and running the program(s) |
| 2. | Define the project problem statement |
| 3. | Describe the datasets |
| 4. | Description of Main Strategies |
| 5. | Evaluations & Results |
| 6. | Pros and Cons of Results |
| 7. | Conclusion and Future Direction |

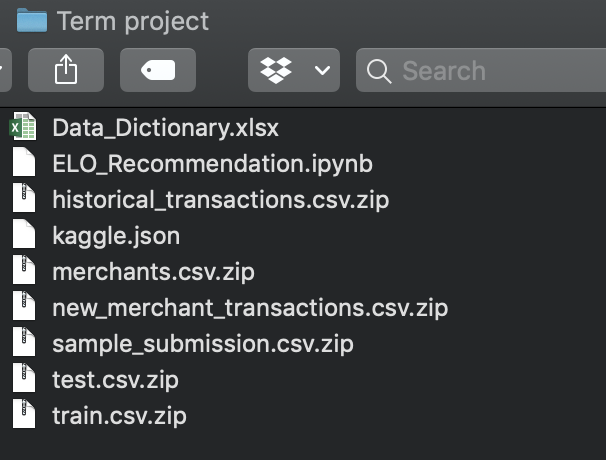
1. **Instructions on compiling and running a program**

In order to run this program, Google Account is required. I used free Jupyter notebook environment that requires no setup and runs entirely in the cloud called [Google Colab](https://colab.research.google.com/notebooks/welcome.ipynb).

Running the program is easy, flexible and less setup environment.

Programming language used for this project is in Python3.

The folder structure for ELO project is shown below as screenshots.

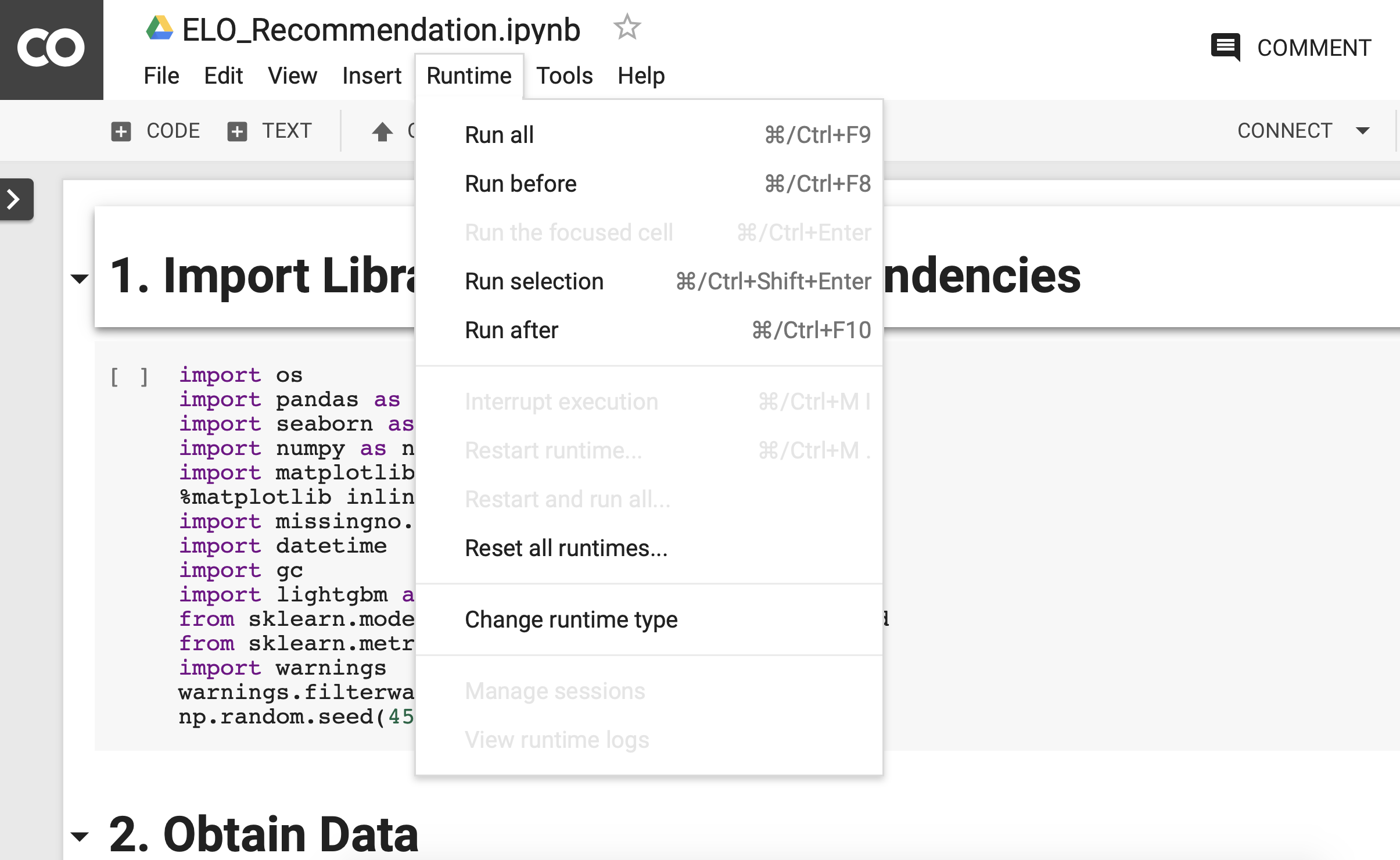


Note: **Kaggle.json** file is key file in order to download the data however I provided my own Kaggle key to able to run and gave a permission to be able to download the data from Kaggle.

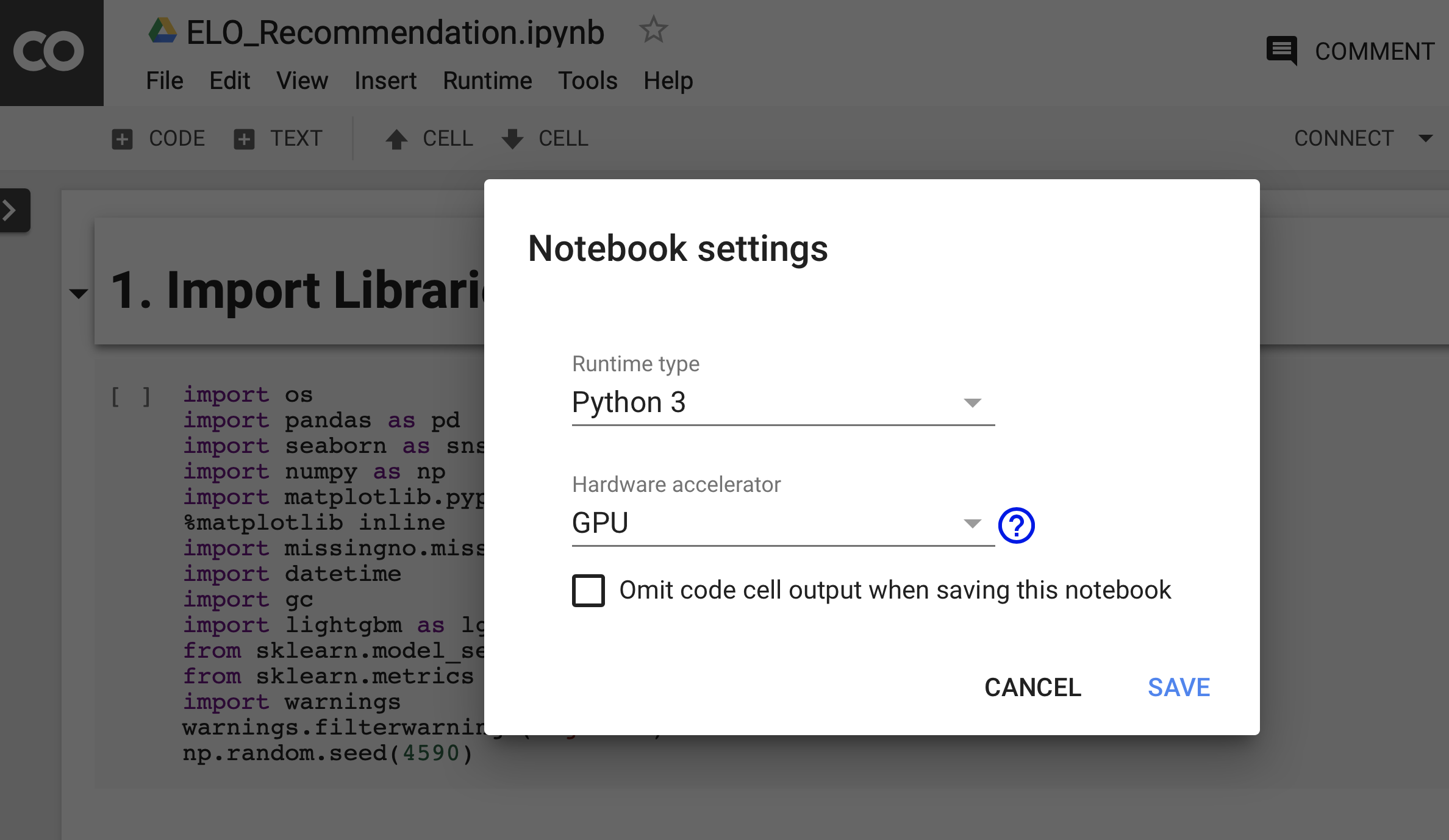
However, the way I set up to obtain the data is downloading through the cloud; saving in google cloud and access in real time from downloaded sources.

The main program is ELO\_Recommendation.ipynb and upload it to google drive.

In order to GPU support to get better performance on training the model, click **Runtime** and select **Change runtime type.**



After that, select **Python3** and select **GPU** to get support from GPU provided by Google.



Similar with the Jupyter Notebook, the execute keys are **shift + enter.**

Note: Line by line execution.

**Note: Data can be downloaded via google cloud, In submission folders, it doesn’t include entire dataset due to the limitation storage on iLearn.**

**2. Define the project problem statement**

This project is one of the Kaggle competition which I competed. The project name is called

**“ELO Merchant Category Recommendations”.** ELO is a Brazilian payments company, has partnerships with many merchants and would like to offer promotions or discounts to cardholders. ELO wants to find out that these promotions work for either the consumer or merchant and wants to check the customers’ experiences along with their personalization. But ELO doesn’t have specifically categorization of individual or profile. Thus, ELO created the Kaggle competitions to solve this business problems with their created Machine Learning models and algorithms to understand the most important aspects and preferences in their customers’ lifecycle of their loyalty in order to drive ELO economic growth.

**3. Data Descriptions**

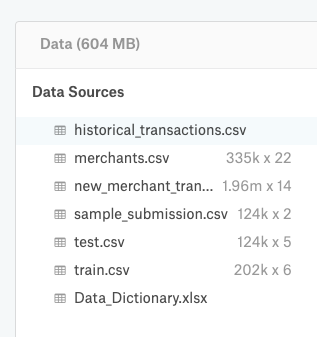
Kaggle provides the entire data including submission sample under data category.

The main two data are **historical\_transactions.csv** and **new\_merchant\_transactions.csv**. Those two csv files contain the information of each cards’ transactions (**historical\_transactions** (14 attributes, 2.85GB) contains up to 3months’ worth of transactions for each card whereas **new\_merchant\_transactions**.csv (14 attributes, 190MB) contains transactions at new merchants over a period of two months.

**Merchants.csv** has additional information about all merchants which include **merchants\_id** in dataset.

There is another file called **new\_merchant\_transactions.csv** contains two-month worth of each **card\_id** include all purchases that **card\_id** made at **merchant\_id** which never visited in historical data.

Data descriptions are shown below:

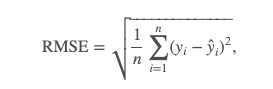


There are some missing data in attributes called **category\_2** in **historical\_transacations.csv**,

Attributes **category\_2** and **category\_3** in **new\_merchant\_transactions.csv.** Handling those missing values (NaN) by filling with mode the fact that numeric values. As well as to make a unique merchant like in **historical\_transactions.csv**, filled with some unique ID.

Kaggle requires submission with y\_target as predicted loyalty score for each card\_id evaluation with the Root Mean Square Error (RMSE) to detect how concentrated the result of the prediction errors.

The Root Mean Square Error (RMSE) equation is as shown below.



**4. Description of Main Strategies**

**Data Preprocessing**

Data preprocessing included filling missing data as well as extracting new features. Extracting new features is needed the fact that to do the aggregation on those features with user-defined functions.

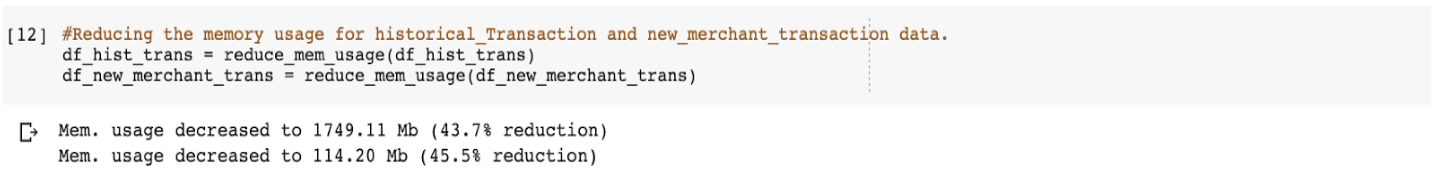
The Locality points are assigned only after 2 months of card usage, which is not given specific in the dataset so that individual time such as creating new columns (extracting year, month, day and time that customers start using card) is necessary. The aggregation features of extracted columns are applying to both **historical\_transacations.csv** and **new\_merchant\_transactions.**

**Step by Step Detail:**

* + Purchase Date, Authorized Flag columns from historical and new\_merchant data
  + Calculated sum, min, max, variance from the features we extracted
  + Created new columns for these calculated features, group by them with card\_id (submission purpose), merge these columns with train data and test data.
  + After adding these data into train and test, deleted historical and new\_merchant.

**Environmental Configuration**

Since the size of the **historical\_transacations.csv** is 2.85GB, in order to run on the google colab cause run time error due to memory shortage. To handle this, use the user-defined function **reduce\_mem\_usage( )** and it reduces almost half percentage of the memory usage.



**Handling Outliers**

In target data has outliers, to compare with the those, creating new outlier columns to impute with the mean of extracted features groups. In detail, adding a new column in training data set for outliers, assigning 1 to the outliers that are below -30, else assign to 0. The fact that target has outlier values which total of 2207. Assigning 1 to the outliers that are below -30 will be able to detect outliers in the column. If this task is not taken, Root Mean Square Error (RMSE) will increase.

**5. Evaluations & Results**

Tuning with various hyper-parameters during the training with stratifiedKFold method applying in Light GBM (Gradient Boosting Method) gets different results and dealing with overfitting problems sometimes. The comparison of tuning with different variables is as shown below.

params = {'boosting': 'gbdt',    ( to tune the hyper-parameters)

        'objective':'regression',

        'metric': 'rmse',

        'learning\_rate': 0.01, # 0.003! #0.005 #0.006

        'num\_leaves': 110, #110 #100 #150 large, but over-fitting

        'max\_bin': 66,   #60 #50 # large,but slower,over-fitting

        'max\_depth': 10, # deal with over-fitting

        'min\_data\_in\_leaf': 30, # deal with over-fitting

        'min\_child\_samples': 20,

        'feature\_fraction': 0.5, #0.5 #0.6 #0.8

        'bagging\_fraction': 0.8,

        'bagging\_freq': 40, #5

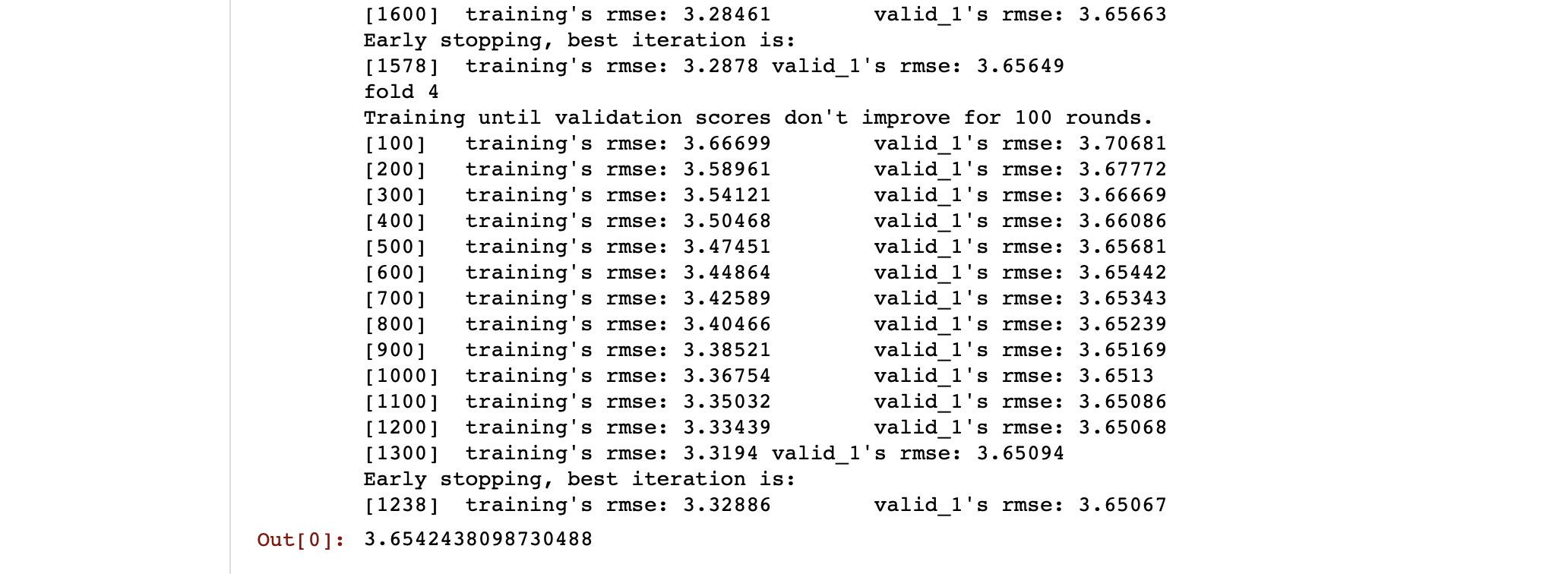
        'bagging\_seed': 11,

        'lambda\_l1': 2, #1.3! #5 #1.2 #

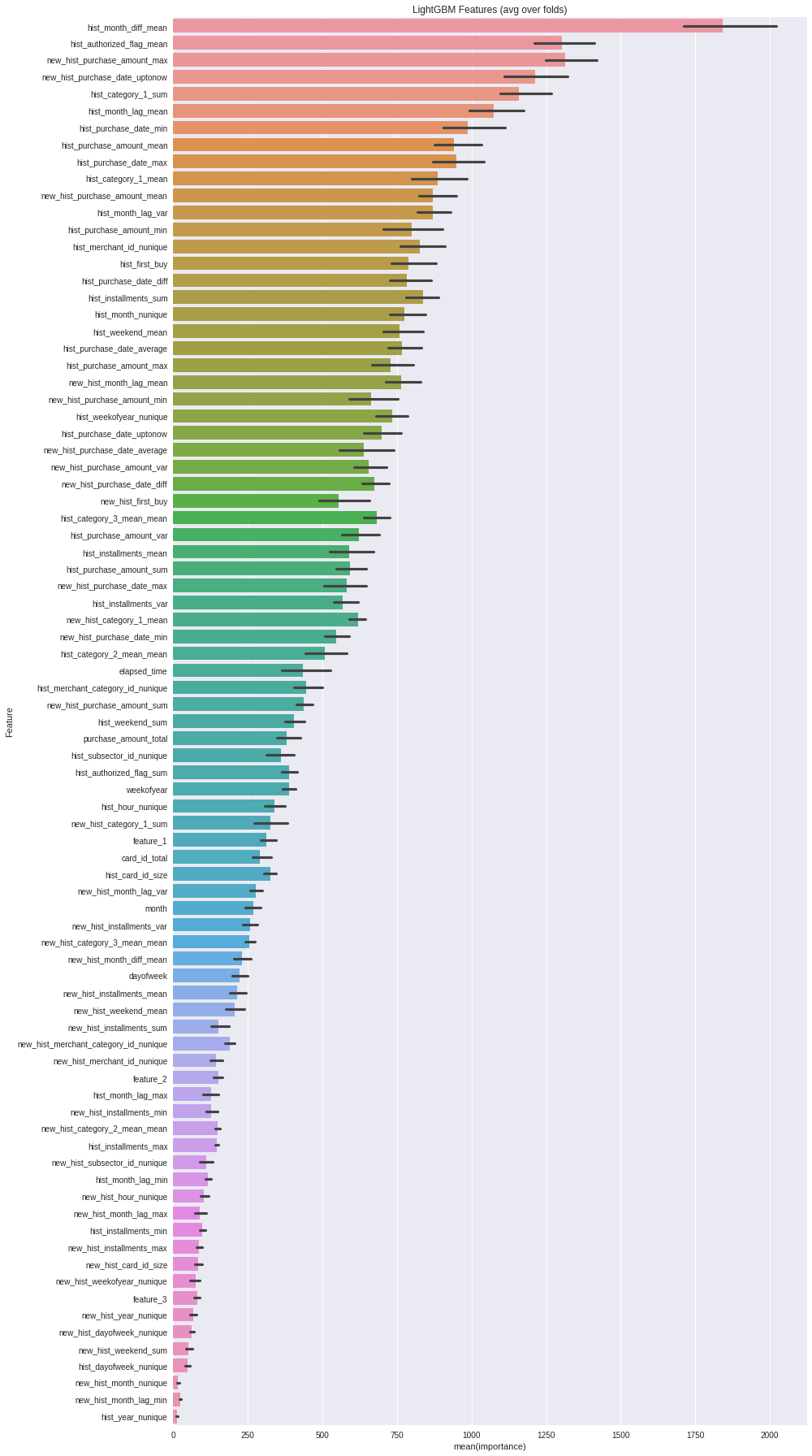
         'lambda\_l2': 0.1 #0.1

       }

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Trial | Early\_stopping  \_rounds | fold | max\_depth | Loss error | Rank on Kaggle |
| 1 | 200 | 5 | Default -1 | 3.6542 |  |
| 2 | 200 | 10 | Default -1 | 3.6514 |  |
| 3 | 100 | 10 | Default -1 | 3.6526 |  |
| 4 | 100 | 5 | 6 (trick) | 3.650 (better) | 425 / 1850 |
| 5 | 200 | 5 | 6 | 3.6504 |  |
| 6 | 100 | 10 | 6 | 3.6488 (best result so far) |  |
| 7 | 300 | 10 | 6 | 3.6505 |  |
| 8 | 100 | 5 | 6 | 3.650536 | Note: Verbose\_Eval = 20, Verbose\_Eval = 500, 100 |
|  |  |  |  |  | Each testing took 20 ~ 25 mins |

****

As a result, after plotting the feature importance, which categories that is more important and affect to the customers’ experiences and loyalty.



**6. Pros and Cons of Results**

While training with Light GBM applying with the stratified K-Folds cross validation, the data is more shuffled and it is more or like normalizing the values as a result of higher accuracy which is one of the pros strategies of training. In more detail, stratification is the process of rearranging the data as to ensure each fold is a good representative of the whole. For example, in a binary classification problem where each class comprises 50% of the data, it is best to arrange the data such that in every fold, each class comprises around half the instances.

There are some limitations such as continuous GPU support and development time is 20 minutes each time of testing so that continuous working on this project is required. There were also run out of memory and killed kernel during training due to the massive amount of data, hyper parameters the model take very long time to finish running of each testing time.

Without the aggregation on extracted columns of both datasets, it is hard to train the model that fact that the result only depends on the 2 months after using card of individual cardholders.

**7. Conclusion and Future Direction**

In conclusion, Light GBM with stratified K-Folds validation is not the best algorithms however in the case of shuffling data within the massive dataset works well. There are other algorithms that under testing mode is the CATBoost Algorithm and XGBoost with Repeated-Fold validation. XGBoost is machine learning optimization algorithm, which is fast and very fast in the case of comparing with other implementations of gradient bossing, computation power and model performance. XGBoost is commonly used in applied machine learning and most of the Kaggle competition.

Surprisingly, the CAT\_Boost Algorithm got the score of 3.620896376 that get lower RMSE, higher score in Kaggle. CatBoost algorithm is better in reduce overfitting where as cause overfitting problems during training with LightGBM and can faster apply trained model and efficiently even to latency-critical tasks and support milti-card configuration for large datasets however it takes a while in completion of training.

XGBoost algorithms took longer and as a result of 3.68351376179364 that does not much increase the accuracy (higher RMSE, lower score on Kaggle). The result text file of running XGBoost is attached with the submission folder.

For Future Direction, some possibilities can be done if time slot available such as trying to manipulate outliers more efficiently from training dataset, trying to run the model isong categorical features the fact that has built-in categorical feature parameters and more adjust with the K-fold validation for better accuracy.

**References**

Algorithms suggestions, common thoughts, ideas are reference from Kaggle competition kernels and discussing forums.