GA Customer Revenue Prediction

AI-511 Machine Learning

Team : Globetrotters

Abhishek Garg
MT2020021
IIIT Bangalore
abhishek.garg@iiitb.org

Nitesh Jain
MT2020118
IIIT Bangalore
nitesh.jain@iiitb.org

Mohammad Asad Ansari
MT2020147
IIIT Bangalore
mohammadasad.ansari@iiitb.org

***Abstract*—This project is inspired from a live Kaggle contest on predicting the revenue of customers using Google Analytics data from a Google merchandise store. The modified project aim is three-fold, identifying revenue generating customers, predicting the revenue generated by the identified customers and finally computing the life time value of the said customers. After processing and understanding the data through Exploratory Data Analysis, the task of identifying the revenue generating customers follows. This has been done by using Logistic Regression, Random Forest and Gradient Boosting Algorithms-LGBM,XGB,CatBoost.To evaluate the classification models AUC and Recall scores are employed. Similarly, for the regression models:Root Mean Squared Error (RMSE) was used.**

***Index Terms*—Exploratory data analysis, Data preprocessing, Feature engineering, encoding, Model Ensembling, Stacking, Supervised Learning.**

Problem Statement

The 80/20 rule has proven true for many businesses–only a small percentage of customers produce most of the revenue. As such, marketing teams are challenged to make appropriate investments in promotional strategies.

In this competition, you're challenged to analyze a Google Merchandise Store (also known as GStore, where Google swag is sold) customer dataset to predict revenue per customer. Hopefully, the outcome will be more actionable operational changes and a better use of marketing budgets for those companies who choose to use data analysis on top of GA data.

Dataset

The data set given in the competition contain multiple feature properties and each row in this dataset corresponds to a different feature, uniquely identified by a column named ID. The training data given is of size:(632000, 12).There are 12 columns in the dataset.

While the dataset provided here has been roughly split by time, the complications and sampling requirements mentioned above may mean you may see imperfect agreement between your cross validation, public, and private scores!

Each row in the dataset corresponds to the details about a customer who visited the Google store.The details include many details about the customer including its location, browser type, device type etc. The dataset needs to be used in order to predict the total revenue earned.

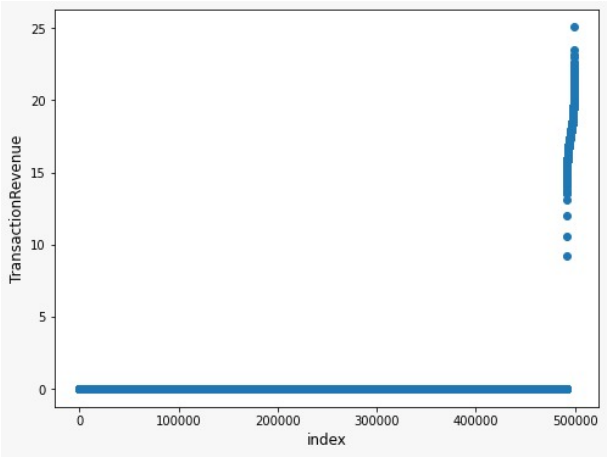

Fig. 1. Imbalance Data Set

I. Introduction

In the current e-commerce world, not every customer produces significant amount of revenue to the business and a thumb rule of 80-20 holds good. This poses a significant challenge for marketing teams to understand the revenue generating base of the customers.

Deriving actionable insights from Google Analytics would help or enable them to optimize their re-targeting and re-marketing strategies and eventually improve their Return on Investment (ROI) from marketing. This can be scaled for those companies who choose to use data analysis on top of Google Analytics.

We will be deploying classification and regression algorithms (Supervised Learning) to identify revenue generating customers, forecast the revenue generated by the identified customer and finally formulate their lifetime value.

II. Data Visualization

The extracted data is a transaction level data of Google Analytics belonging to the E-Commerce domain. The data

contains 632000 transactions and 54 features. Out of these 54 features, 9 are numerical and 45 are categorical(including binary). Upon first look, 11 features had more than 50 % missing values and these were removed from the feature set. As we were in a global competition, some information could be critical to GAStore, so for keeping that info hidden they replaced the whole column with one value ('Not available in demo dataset') For the classification aspect of the project the data was grouped by visitor ID.

- Features that belong to similar groupings are tagged as such in the feature names (for ex: device, totals, geoNetwork and trafficSource).
- Feature names include the postfix bin to indicate binary features and cat to indicate categorical features.
- Features without these designations are either continuous or ordinal.

Feature	Total Missing Count	% of Total Observations
trafficSource_adContent	624199	98.765665
totals_transactionRevenue	623895	98.717563
trafficSource_adwordsClickInfo.adNetworkType	616866	97.605380
trafficSource_adwordsClickInfo.slot	616866	97.605380
trafficSource_adwordsClickInfo.page	616866	97.605380
trafficSource_adwordsClickInfo.isVideoAd	616866	97.605380
trafficSource_adwordsClickInfo.gclId	616791	97.593513
trafficSource_isTrueDirect	440213	69.653956
trafficSource_referralPath	400735	63.407437
trafficSource_keyword	351881	55.677373
totals_bounces	316642	50.101582
totals_newVisits	140503	22.231487
totals_pageviews	69	0.010918
device_browser	0	0.000000
device_mobileDeviceModel	0	0.000000

Fig. 2. Missing Value Percentage

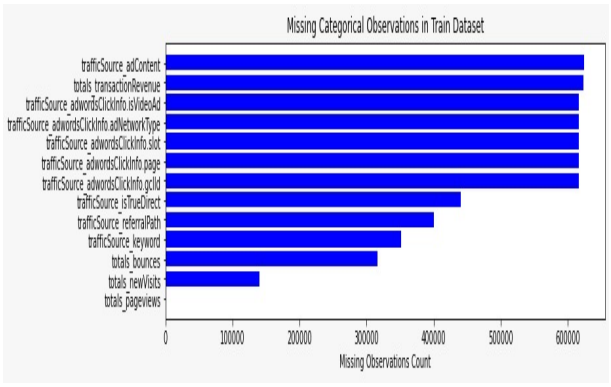

Fig. 3. Missing Data Values

- Impute: Imputation preserves all cases by replacing missing data with an estimated value based on other available information. Once all missing values have been imputed, the data set can then be analysed using standard techniques for complete data. To handle missing values, in the numerical coloumns we used 0 or base level replacement policy(i.e., we replaced the null values by 0 or 1 as required) and for categorical coloumns, we replaced them with a different value like "Unknown"

III. Exploratory Data Analysis(EDA)

The extracted data is a transaction level data of Google Analytics belonging to the E-Commerce domain. The data contains 632000 transactions and 55 features. Out of these 55 features, 12 are numerical and 43 are categorical(including binary). Upon first look, 12 features had more than 50 % missing values and these were removed from the feature set. Apart from this, 18 single level features were also dropped. For the classification aspect of the project the data was grouped by visitor ID and for the target variable, a new flag was created for the customers who generated revenue.

A. Exploring Date and VisitStartTime

The date field is broken down to extract multiple sub-coloumn named day of the week, date of month, month and year. Further visitStartTime was replaced with visithour. Then we made the plot of each of them to analyse them all. The information we came to know that in the starting of the month there was less sale but in the middle of the month the sale went high

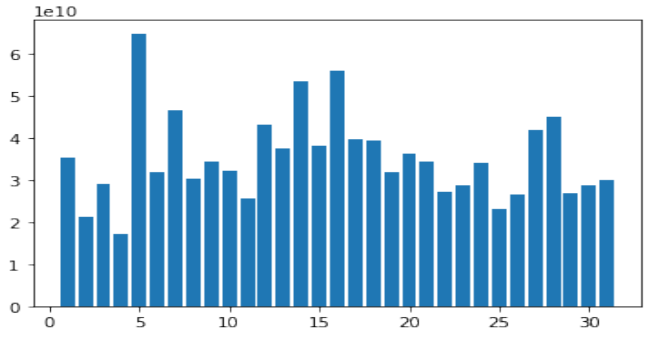

Fig. 4. Day Wise Sale Analysis

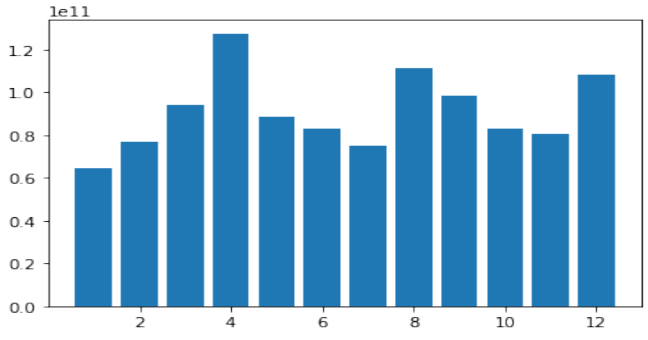

Fig. 5. Month Wise Sale Analysis

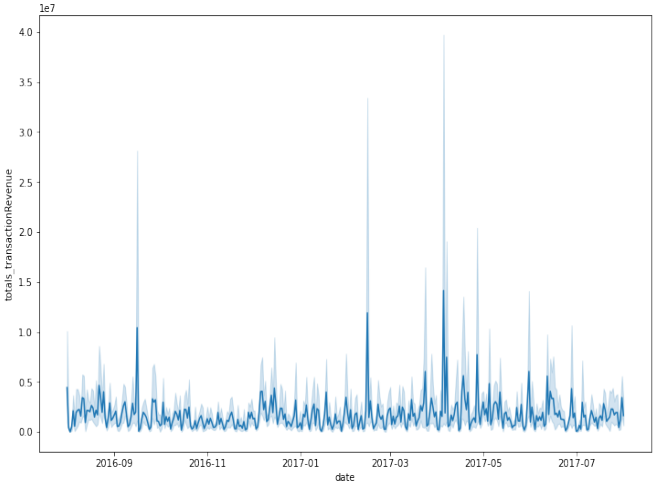

Fig. 6. Complete Year Wise Revenue

B. Analysis of Channel Grouping

In channel grouping, we try to find out that if a customer comes to the store then by what medium has he come to the site. In this organic search, we find out by which we medium that is by which referral is the buyer coming to the site.

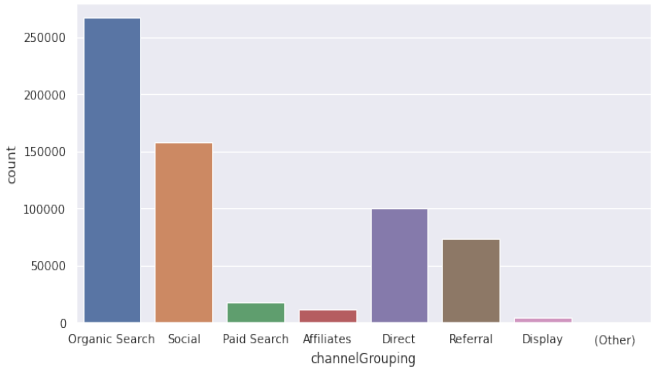

Fig. 7. Hit Counts of Channel Grouping

C. Extract Coloumn specific Category

IN our dataset columns, certain columns were from same category but giving different information, Like all the columns which start with 'device' gave device info like devicebrowser, deviceoperatingSystem, deviceisMobile, devicedeviceCategory. So to extract these subcolumns of particular category along with the number of unique values each subcolumn have we will use column extract.

D. Explore Device Coloumn and subColoumn

In this section, we took 4 plots with different axis values that are count, count of non zero revenue, mean transaction values, total revenues. These plots help us visualize the most popular browser, browser that gives the highest mean transaction revenue, the browser that gives us the maximum total revenue. We can also see that the mean transaction value is the highest for Firefox browser but the total revenue is highest for Chrome. From this we can interpret that the users that use firefox using users usually buy products if they visit the site. Similarly, we can see that most of the customers use chrome.

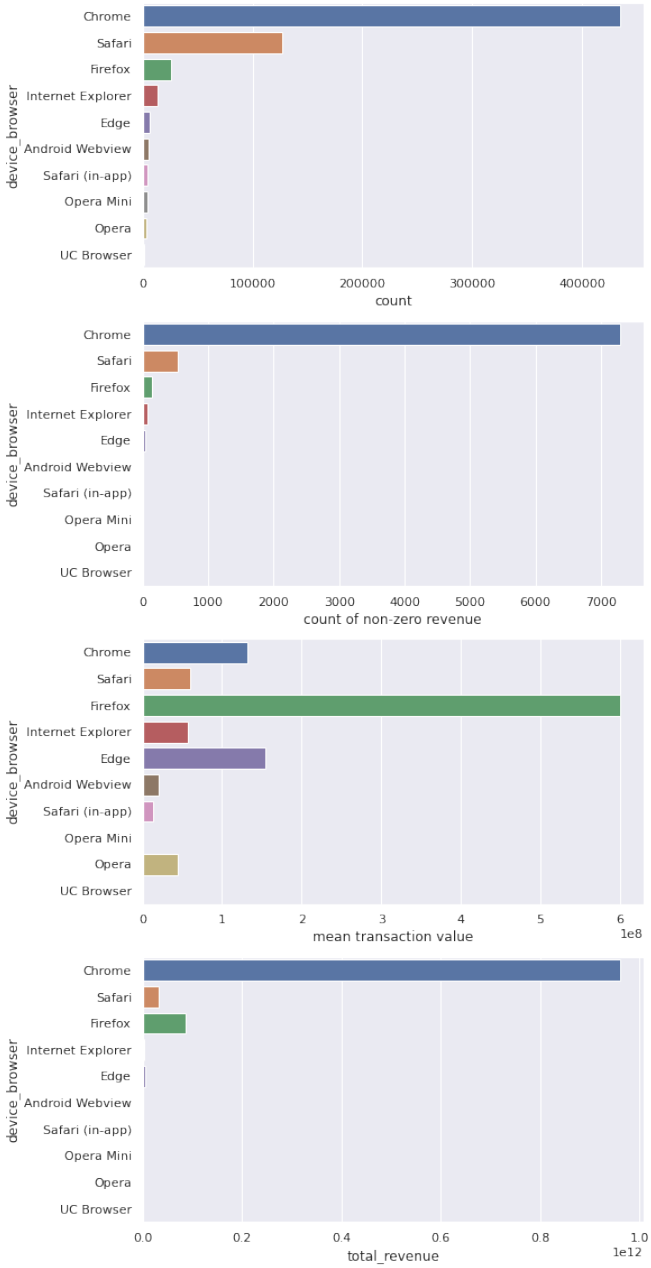

Fig. 8. Revenue Generation By Device Browser

E. Explore GeoNetwork Coloumn

Similarly, in this section also we make the same 4 plots on the geonetwork locations that are continent, subcontinent, country and network domain to visualize the same things that we did in the device coloumn to get some intuitions about the geonetwork locations of the visitors.These plots help us

visualize the continent with most vistiors, continenet that gives the highest mean transaction revenue, the continent that gives us the maximum total revenue. We can also see that the mean transaction value is the highest for Africa but the total revenue is highest for Americans. From this we can interpret that the users that are Africans usually buy products if they visit the site.

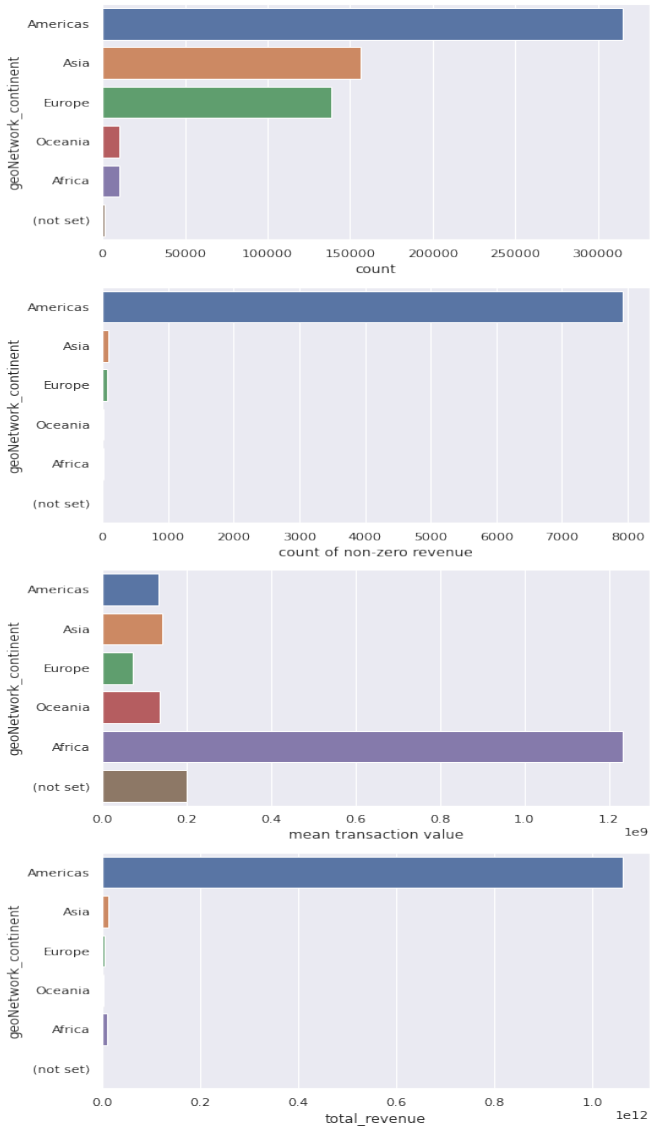

Fig. 9. Revenue Generation Based On GeoNetwork Location

IV. Data Pre-processing

There are total of 59 features in the train dataset while in test dataset there are 58 features. It is observed that Target feature is missing. It is also observed from the dataset that the data is highly skewed and contains a lots of missing values. In the dataset, there are both numerical as well as categorical features.

Handling Missing Data

The data visualization tells us that some of the data has very large number of missing values. These missing values need to be handled in some way. Here, we have handled both numerical and categorical missing values in different ways.

A. Numerical missing values

```
train_date[numerical_cols].isnull().sum()

visitNumber                  0
totals_transactionRevenue    0
_weekday                     0
_day                         0
_month                       0
_year                        0
_visitHour                   0
totals_bounces          316642
totals_hits                  0
totals_newVisits        140503
totals_pageviews            69
dtype: int64
```

```
[ ] test_date[numerical_cols].isnull().sum()

visitNumber                  0
totals_transactionRevenue    0
_weekday                     0
_day                         0
_month                       0
_year                        0
_visitHour                   0
totals_bounces          136381
totals_hits                  0
totals_newVisits         60090
totals_pageviews            31
dtype: int64
```

Fig. 10. Number of missing values in Numerical data

For the numerical data, we did the null values implementation where we fill the values that has null in it with some other values like 0 or 1 using the method fillNA(0). This completes the row and leaves no values as null so that we can smoothly run our model to get the predictions.

B. Categorical missing values

```
channelGrouping                          0.000000
device_browser                           0.000000
device_operatingSystem                   0.000000
device_isMobile                          0.000000
device_deviceCategory                    0.000000
geoNetwork_continent                     0.000000
geoNetwork_subContinent                  0.000000
geoNetwork_country                       0.000000
geoNetwork_region                        0.000000
geoNetwork_metro                         0.000000
geoNetwork_city                          0.000000
geoNetwork_networkDomain                 0.000000
trafficSource_campaign                   0.000000
trafficSource_source                     0.000000
trafficSource_medium                     0.000000
trafficSource_keyword                   55.677373
trafficSource_referralPath              63.407437
trafficSource_isTrueDirect              69.653956
trafficSource_adwordsClickInfo.page     97.605380
trafficSource_adwordsClickInfo.slot     97.605380
trafficSource_adwordsClickInfo.gclId    97.593513
trafficSource_adwordsClickInfo.adNetworkType  97.605380
trafficSource_adwordsClickInfo.isVideoAd      97.605380
trafficSource_adContent                 98.765665
dtype: float64
```

Fig. 11. Number of missing values in categorical data

To handle missing data amongst the numeric features we employ mode/ most_frequent strategy of imputation. For categorical features Label Encoder automatically assigns a numeric category to missing (NaN) values.

V. Feature Engineering

We also created 2 columns, namely meanhitspernetwork-Domain and meanpageViewspernetworkDomain just to get an extra information of the average page hits a user does in a network domain and average page views a user do in a network domain. This is created by intuitions considering that a normal person of different countries or region have different traits of online shopping also it can also be seen in Figure that GAStore have different sales in different continents. We also tried to do multiple changes with data like:

1) 'Device_Browser' had 49 unique values, we took most popular and contribution values based on mean transaction and overall transaction values. Like 'Amazon Silk' is not that popular among users but its users have large mean_transaction_values and total_revenue.

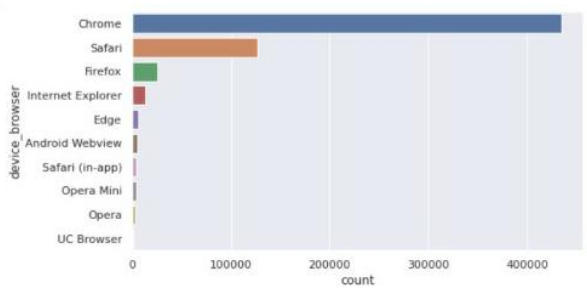

Fig. 12. Number of user of particular device browser

2) Similarly we took most occuring and most contributing 12 categories from 'Device_OperatingSystem' among 17 categories and marking others as 'unpopular'. This is done just to reduce model complexity and to prevent model from overfitting.

3) We also tried for 'geoNetwork_NetworkDomain' as it have 22823 categories or values. on basis of most popular and contributing ones but it has a huge variation in data there were categories which occured alot but were not giving revenue and also there were some which didn't occured alot but generating good revenue.

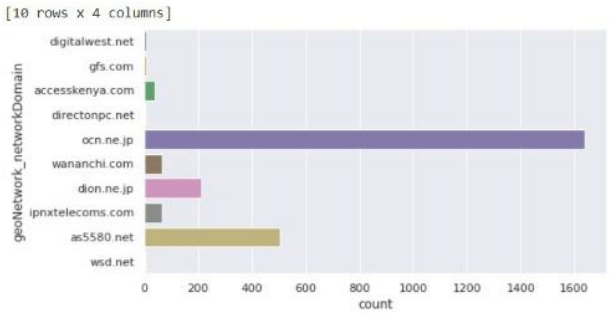

Fig. 13. Number of user of particular networkDomains

4) There was similar case as in 'geoNetwork_NetworkDomain' for columns 'trafficSource_referralPath' and 'trafficSource_keyword'.

5) For column 'trafficSource_adContent' we were also trying to reduce its categories but as each ad has different impact we didnt want to lose out on information.

A. Encoding

We applied three type encoding techniques.

- Label Encoding
- Frequency Encoding

Label Encoding: refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

Frequnecy Encoding: It is a way to utilize the frequency of the categories as labels. In the cases where the frequency is related somewhat with the target variable, it helps the model to understand and assign the weight in direct and inverse proportion, depending on the nature of the data.

VI. Model Ensembling

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. The ensemble model then aggregates the prediction of each base model and results in once final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction.

A. Group Kfold

K-fold iterator variant with non-overlapping groups. The same group will not appear in two different folds (the number of distinct groups has to be at least equal to the number of folds). The folds are approximately balanced in the sense that the number of distinct groups is approximately the same in each fold. Group Kfold is used to divide dataset in K parts such that same group will not appear in two different folds (the number of distinct groups has to be at least equal to the number of folds). The folds are approximately balanced in the sense that the number of distinct groups is approximately the same in each fold. Group Kfold will help in training our dataset so that the Model will train fro K-1 folds and use Kth fold as validation dataset, and this process will be repeated K times keeping different fold as Kth fold

B. Light GBM

Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise.

C. XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve problems in a fast and accurate way.

D. Catboost

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy. It is especially powerful in two ways: It yields state-of-the-art results without extensive data training typically required by other machine learning methods. !provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

E. RandomForest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks operate by constructing a multitude decision trees at training time and outputting the class that is the mode of the classes.

VII. Hyperparameter Tuning

Hyperparameters are important for machine learning algorithms as they control the behaviors of training models and have a significant impact on the performance of machine learning models.There are different techniques used for the tuning of hyperparameters. Some used techniques are.

A. Grid Search

This is a time consuming process as it tries all the possible configurations of the parameters. In the end it returns the configuration which provides the best score. It is best to use it if the dimensions are less.

B. Bayesian Optimization

Bayesian Optimization provides a principle technique based on Bayes Theorem to search a solution of a problem that is efficient and effective. It works by building a probabilistic model of the objective function, called the surrogate function, that is then searched efficiently with an acquisition function before the samples are chosen for evaluation on the real objective function.

VIII. Training the model

Our early stage models were brute force trying to go through different types of regression models and implementing them in our problem statement with out much of parameter tuning because of which models generated with algorithms like RandomForest, GradientBoosting,and XGB were giving very less accuracy. Later with the help of Grid Search over these classifiers one by one we were able to get good accuracy. After feature engineering, our number of useful features reduced.Reducing the features also decreased the complexity of the model as well.We trained our data on the following algorithms:

- Logistic + Linear Regression
- Random Forest
- Light GBM
- XGBoost
- CatBoost
- KFold + Light GBM
- 0.5 * LGBM + 0.5 * CatBoost
- 0.7 * LGBM + 0.3 * CatBoost

S.No.	Algorithm	RMSE Score
1.	Logistic + Linear Regression	3.201
2.	Random Forest	2.652
3.	LightGBM	1.872
4.	XGBoost	1.750
5.	CatBoost	1.872
6.	KFold + LightGBM	1.870
7.	0.5 * LGBM + 0.5 * CatBoost	1.87139
8.	0.7 * LGBM + 0.3 * CatBoost	1.87165
9.	Dual Catboost + LGBM	1.853

```
GroupKFold + LGBM
Starting LightGBM. Train shape: (632000, 36), test shape: (271653, 36)

 Fold :  1
Training until validation scores don't improve for 500 rounds.
Did not meet early stopping. Best iteration is:
[100]   training's rmse: 1.80941        valid_1's rmse: 1.85489
Fold  1 RMSE : 3.440606

 Fold :  2
Training until validation scores don't improve for 500 rounds.
Did not meet early stopping. Best iteration is:
[100]   training's rmse: 1.81477        valid_1's rmse: 1.82357
Fold  2 RMSE : 3.325412

 Fold :  3
Training until validation scores don't improve for 500 rounds.
Did not meet early stopping. Best iteration is:
[100]   training's rmse: 1.81304        valid_1's rmse: 1.83241
Fold  3 RMSE : 3.357741

 Fold :  4
Training until validation scores don't improve for 500 rounds.
Did not meet early stopping. Best iteration is:
[100]   training's rmse: 1.81578        valid_1's rmse: 1.80649
Fold  4 RMSE : 3.263393

 Fold :  5
Training until validation scores don't improve for 500 rounds.
Did not meet early stopping. Best iteration is:
[100]   training's rmse: 1.81695        valid_1's rmse: 1.81297
Fold  5 RMSE : 3.286865

 Fold :  6
Training until validation scores don't improve for 500 rounds.
Did not meet early stopping. Best iteration is:
[100]   training's rmse: 1.81937        valid_1's rmse: 1.79556
```

Fig. 14. Kfold LGBM Mdoel

```
0:      learn: 1.9654443        test: 2.1440416 best: 2.1440416 (0)     total: 368ms
300:    learn: 1.6092644        test: 1.7463372 best: 1.7463372 (300)   total: 1m 25s
600:    learn: 1.5758627        test: 1.7279531 best: 1.7279531 (600)   total: 2m 50s
900:    learn: 1.5483731        test: 1.7200129 best: 1.7199896 (897)   total: 4m 12s
1200:   learn: 1.5192658        test: 1.7150194 best: 1.7150057 (1199)  total: 5m 34s
1500:   learn: 1.4929405        test: 1.7113449 best: 1.7113036 (1497)  total: 6m 57s
1800:   learn: 1.4689451        test: 1.7093822 best: 1.7093461 (1794)  total: 8m 19s
2100:   learn: 1.4477298        test: 1.7084073 best: 1.7083615 (2092)  total: 9m 40s
Stopped by overfitting detector  (150 iterations wait)

bestTest = 1.708286068
bestIteration = 2220

Shrink model to first 2221 iterations.
CPU times: user 21min 20s, sys: 19.7 s, total: 21min 40s
Wall time: 11min 1s
```

Fig. 15. CatBoost Model

IX. Conclusion

In order to get the best results, we used a stacking of three models and calculated the weighted average score for these. Firstly, we used a groupKFold working on LGBM model having a learning rate of 0.005 and having a total of 6 folds. Secondly, we used two catboost models differentiating on the value of hyperparameters mainly the learning rate, number of leafs and threads. Both the catboost model were running on 6 folds. Finally, we used weighted averaging where LGBM contributed to 50 % and catboost to 25 % each. Finally the score was obtained using these which was the best score i.e., **1.85334**.

Below is the plot of feature importance that depicts the features affecting the model accuracy.

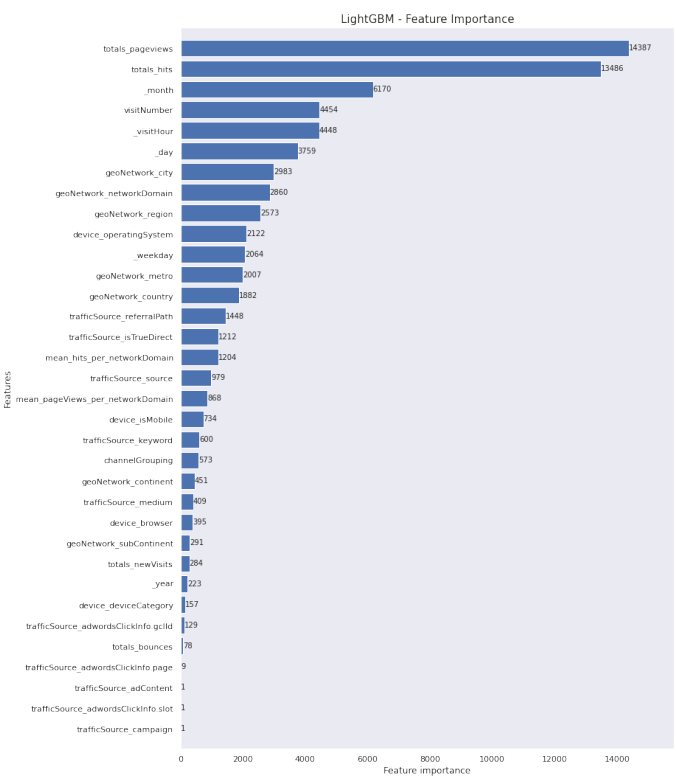

Fig. 16. Feature Importance

X. Acknowledgement

The success and outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of the project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We would like to thank Professor G. Srinivas Raghavan and our Machine Learning teaching assistant Shreyas Gopta,for giving us the opportunity to work on the project and helping us out in the initial stages in numerous occasions. His highly detailed lectures on various topics helped us understand what we were actually doing.

Leader-board was great motivation to work on the project and the competition forced us to read up various articles and papers which gave us ideas and enthusiasm for the project.

XI. PROJECT FILE LINK

Our project folder including the python code,script file and submission csv file,is available at the following GDrive link: https://rb.gy/o4rqj9

XII. REFERENCES

[1] Jovan Sardinha. An introduction to model ensembling. [Online]. Available: https://medium.com/weightsandbiases/an-introduction-to-model-ensembling-63effc2ca4b3

[2] Philip Hyunsu Cho, Nan Zhu et.al., XGBoost. [Version: 1.2.0]. [Online]. Available:https://xgboost.readthedocs.io/

[3] Jérémie du Boisberranger, Joris Van den Bossche et.al., sckit-learn: Open source scientific tools for Machine Learning. [Latest] [Online]. Available: https://scikit-learn.org/

[4] Microsoft Corporation Revision 9597326e. LightGBM. [Latest]. [Online]. Available: https://lightgbm.readthedocs.io/

[5] Andrei (Andrey) Khropov, annaveronika et.al., catboost.ai. [version: 0.24.3]. [Online] https://github.com/catboost/catboost

[6] E. Jones, T. Oliphant, P. Peterson et al., "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed ¡today¿]. [Online]. Available: http://www.scipy.org/

[7] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.

[8]

[9] S. Raschka, "Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack," The Journal of Open Source Software, vol. 3, no. 24, Apr. 2018. [Online]. Available: http://joss.theoj.org/papers/10.21105/joss.00638

[10] A. S. Hendrik Jacob van Veen, Le Nguyen The Dat, "Kaggle ensembling guide," 2015, [Online; posted 6-February-2018]. [Online]. Available: https://mlwave.com/kaggle-ensembling-guide/

[11] Upasana, "How to handle imbalanced classification problems in machine learning?" 2017, [Online; posted 7-March-2017]. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/ 03/imbalanced-classification-problem/