Elo Merchant Category Recommendation

*

Shivani Mittal
Indraprastha Institute of Information Technology
Delhi, India
shivani19128@iiitd.ac.in

Harshita Pandey
Indraprastha Institute of Information Technology
Delhi, India
shivani19128@iiitd.ac.in

***Abstract*—The basic aim of this project is to build various machine learning models which predict the loyalty score of the customers with less error. This prediction helps the Elo brand to understand their customers and provide them discount or offers based on their loyalty score. In this project, we analyzed the data and performed feature engineering. We also built several machine learning algorithms and explained why this approach is better. The results show that the Light GBM model with hyper-parameter tuning gave better performance with 3.64631 Root Mean Squared Error. The results also show the dataset is sensitive to outliers. The model is trained without the outliers but there is no improvement in the performance of model.**

I. Introduction

Elo is one of the largest payment brands in Brazil. It offers financial services, including debit card and credit card. Elo has built partnership with merchants in order to provide offers or discounts to the card holders. Elo has built several machine learning models but none of the system provides personalized benefits to the customer. So, it organized a challenge on Kaggle named "Elo Merchant Category Recommendation". The main aim behind this challenge is to develop a machine learning model which can predict the loyalty score of the card holders using the customer transactions data. This model helps Elo to improve their customer experience by providing them discount or offers based on their loyalty score.
The problem statement is "How can machine learning models be used to best predict customer loyalty using Elo's dataset?".

II. Literature Review

Machine learning algorithms are used to find the patterns in the dataset which helps the companies to identify their customer needs and provide them benefit accordingly. Several researches have done in this field and various machine learning algorithms have built to find patterns from the different type of dataset. [AHC19] published a research work in which they applied feature engineering and various models for predicting the customer loyalty. They achieved 3.710 RMSE score on the test dataset. Another research work [MKS19] used Light GBM and XGB model for predicting the customer loyalty and achieved 3.676 score.

Identify applicable funding agency here. If none, delete this.

III. Data Analysis

The dataset included several data files namely 'train.csv', 'test.csv', 'historical_transactions.csv', 'new_merchant_transactions.csv', 'merchants.csv'. The dataset is simulated and fictious. It is not a real customer dataset. The dataset is taken from the Kaggle challenge.

A. Training and Testing Dataset

The training and testing dataset contains information about the card like 'card id', the first month the card was active and three anonymized categorical features. The training data has one target feature i.e., loyalty score. The figure 1 shows the distribution of 'feature_1', 'feature_2', 'feature_3' of the training and testing dataset. It is observed that all the features are equally distributed in training and testing dataset.

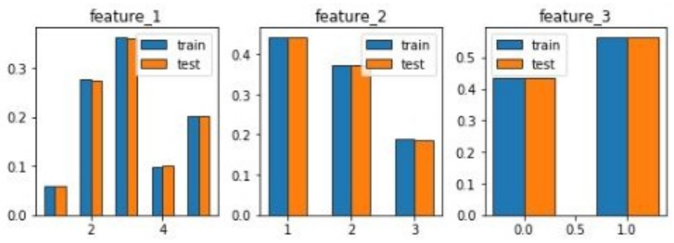


Figure 1. Distribution of features

The figure 2 shows distribution of target in the dataset. It is observed that outliers are present in the dataset and apart from outliers most of the loyalty score lies in the range of [-10, 10].

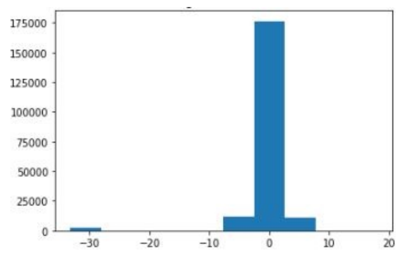


Figure 2. Distribution of target

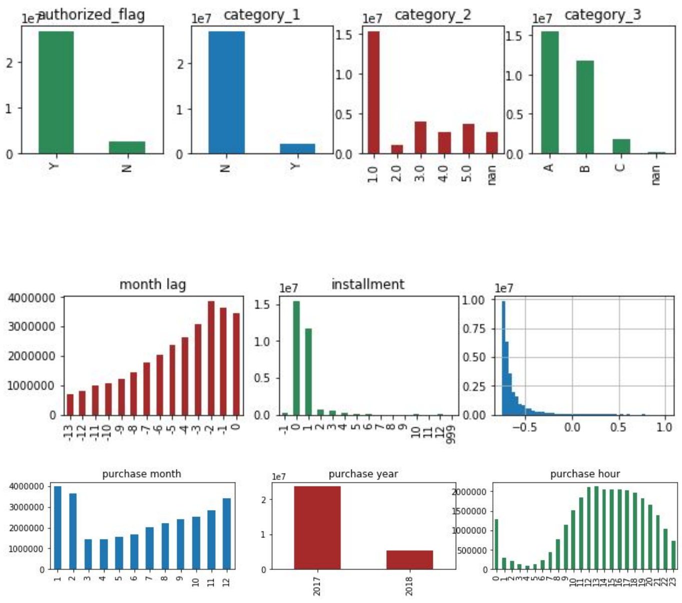


Figure 3. Historical transaction dataset overview

B. Historical Transactions Data

Historical transaction contains information about the transactions made by each card id in 3 months at any of the provided merchant id. This file contains data of 29112361 transactions made by 325540 unique card id. The dataset has 14 features out of which category_2, category_3, merchant_id contains null values. The observations made from figure 3 is listed below:

- In the dataset some non-authorized users are present.
- The installment feature is either normalized or contain false values because the column has -1 and 999 values.
- Month lag column shows that all the transactions were made before the reference date.
- The purchase amount column is highly normalised as it contains values in the range of -0.74 to 6010603.971. But 98.80% of the values lies in the range of [0, 1].

C. New Merchant Transaction Data

New merchant transaction contains transaction at new merchants over a period of two months. This file contains data of 1963031 transactions made by 290001 unique card id. The dataset has also 14 features out of which category_2, category_3, merchant_id contains null values. The observations made from figure 4 is listed below:

- Unlike historical data, all the users are authorized.
- The installment feature is either normalized or contain false values because the column has -1 and 999 values.
- Month lag column shows that all transactions were made after the two months of the reference date.
- The purchase amount column is highly normalised as it contains values in the range of -0.74 to 263.15. But 98.58% of the values lies in the range of -1 to 1.

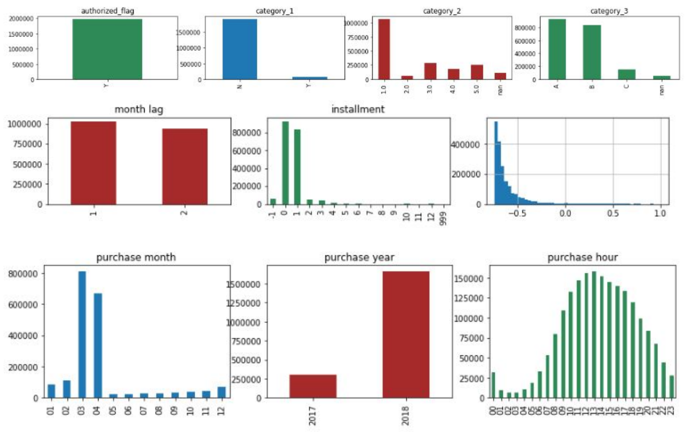


Figure 4. New Merchant Data Analysis

- In fig: 3 and III-D, it is observed that historical file contains most of the transaction in year 2017 while new merchant file contains transactions from the year 2018. Both graph shows that transactions are more prominent in the evening hours.

D. Merchant Data

Merchant data contains information about each merchant. It has the data of 334633 merchants. The observations made from figure III-D is listed below:

- category_1, category_2, category_4, most_recent_sales_range, most_recent_purchase_range are anonymized categorical attributes. category_2 attribute contains null values.
- numerical_1, numerical_2 are anonymized continuous attribute which contains approximately 98% values in the range of [0, 1].
- avg_sales_lag3, avg_sales_lag6, avg_sales_lag12, avg_purchases_lag3, avg_purchases_lag6, avg_purchases_lag12 are normalized continuous attribute. These attributes contains approximately 98.7% values in the range of [0, 10].

IV. Feature Engineering

This section deals with the inference made in the previous section and new features being added in the training and testing data to increase more information and authenticity of the data. We generate more features using the historical, new merchant transactions and merchant data for better predictions of the loyalty score.

A. Train and Test Data

The training and testing data contains three categorical features – feature_1, feature_2, and feature_3. We generate new features by taking sum, mean, maximum, minimum, variance of each value of these features. We have converted first active month into date time format and generate new

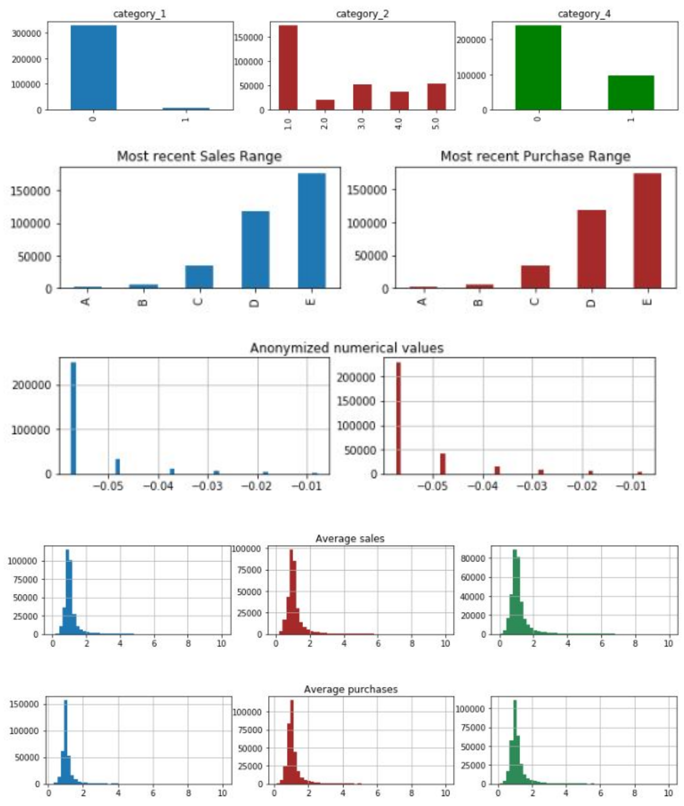
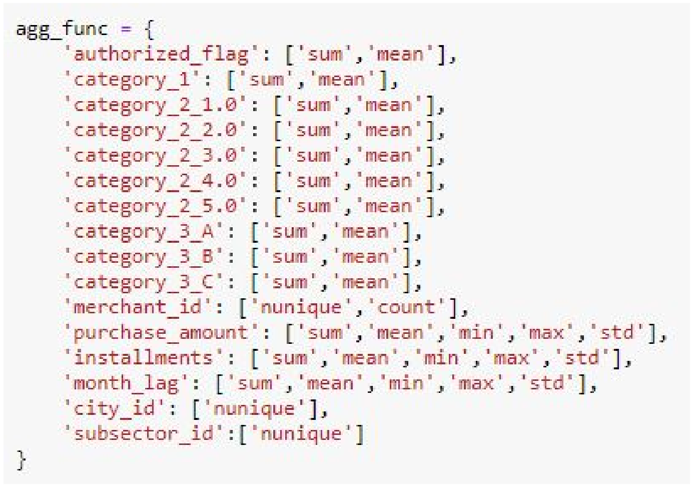


Figure 5. Merchant data Analysis

features by taking quarter of date, number of days from max date to first active month.

B. Historical Transaction data

For the historical data, authorized flag and category_1 contains values in alphabetical format as shown in Fig: III-B, we encode their values into their binary equivalent. Category_2, category_3 and merchant_id contains null values. So, we replace their null values with their respective mode values.
The data contains two categorical attributes - category_2 and category_3. These features are converted into one-hot encoding scheme. New features are generated for each category of a feature.
In the dataset each 'card_id' has multiple transaction data. We can generate new features by grouping dataset corresponding to each 'card_id' and take mean, sum of the values of feature corresponding to each unique card id. Different metric can be used for different features. Features like city_id, state_id, which contain ids, we can take the number of unique ids present and count of the ids. Features which contain numeric values we can take their mean, sum, standard deviation, minimum value, maximum value. Fig 6 shows the aggregate function used for feature engineering.



```
agg_func = {
    'authorized_flag': ['sum','mean'],
    'category_1': ['sum','mean'],
    'category_2_1.0': ['sum','mean'],
    'category_2_2.0': ['sum','mean'],
    'category_2_3.0': ['sum','mean'],
    'category_2_4.0': ['sum','mean'],
    'category_2_5.0': ['sum','mean'],
    'category_3_A': ['sum','mean'],
    'category_3_B': ['sum','mean'],
    'category_3_C': ['sum','mean'],
    'merchant_id': ['nunique','count'],
    'purchase_amount': ['sum','mean','min','max','std'],
    'installments': ['sum','mean','min','max','std'],
    'month_lag': ['sum','mean','min','max','std'],
    'city_id': ['nunique'],
    'subsector_id':['nunique']
}
```

Figure 6. Aggregate Function

C. New Merchant Transaction data

Same feature engineering can be done for new merchant transaction dataset. The dataset is grouped based on the 'card_id' and calculate the aggregate values of each column using the aggregate function mentioned in fig 6.

D. Merchant Data

In the merchant data, average_sales_lag3, average_sales_lag6, average_sales_lag12 contain null values which are replaced with 0. category_2 column also consists of null values, which is replaced with its mode value i.e., 1.
We have noticed that merchant data do not contain 'card_id' but train data has 'card_id' as key value. So, to combine merchant features with train data we have to map 'merchant ids' to the 'card-id' using either historical transaction data or new merchant transaction data. 99.99% of unique merchant ids from historical transaction are present in merchant data but the historical data is large, and it requires a lot computation to merge them. In new merchant data 99.99% of unique merchant ids are also present in merchant data. So, we merge merchant data with new merchant transaction data based on card_id. After merging them, 35 new features are generated.

All the generated features from historical transaction and merging of merchant and new merchant transactions are then combined with training and testing data based on the 'card_id'. Now, in both training and testing dataset 144 features are present. Training data has one target column that represents the loyalty score.

V. Methodology

A. Experimental Scenario: Selecting Baseline Model

The main aim of this project is to build a machine learning model which can predict the loyalty score of the testing data with the lowest root mean squared error. Initially, the models were built on the default parameters and without adding any other features. We have used Linear Regression, Random

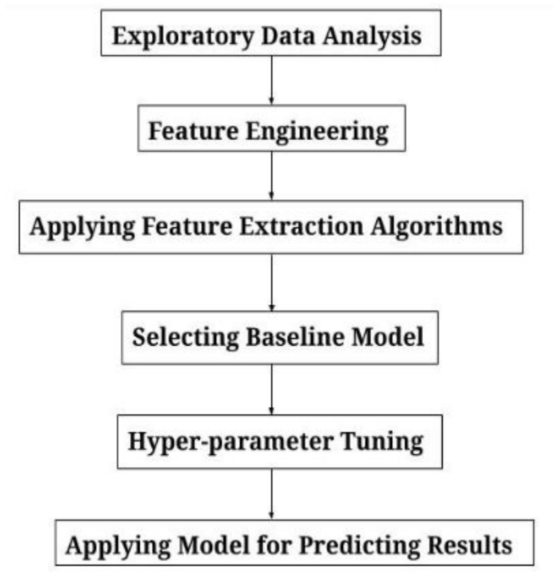

Exploratory Data Analysis
Feature Engineering
Applying Feature Extraction Algorithms
Selecting Baseline Model
Hyper-parameter Tuning
Applying Model for Predicting Results

Figure 7. Methodology

Forest models provided by the sklearn library as the baseline models. We also used various ensemble methods to check the performance of data towards boosting. The dataset is quite large, so it is beneficial to use LightGbm and XGBoost algorithms. These algorithms are fast, efficient, accurate and compatible with large dataset. The table I shows the training and testing root mean squared error on different regression models. These results will be used as baseline for the further models.

Table I
RMSE SCORE ON BASELINE MODELS

Model	Training RMSE	Testing RMSE
Linear Regression	3.8499	3.81445
Random Forest Regressor	3.8488	3.81300
Light Gbm	3.8494	3.81386
XGB Boosting Algorithm	3.7186	3.81305

B. Experimental Scenario: Feature Importance

After setting the baseline model, the model is built over the 144 features generated in section IV. We trained linear regression model and random forest algorithm for all the newly added features for testing the separability of the model. The model performed poorly by stating a RMSE of 3.76127 and 3.93097 respectively. We also tried to run Support Vector Machine, but the notebook was not able to allocate that much memory because the dataset and number of features are large. Now, we calculated the k-best features using the chi2 and random forest method. All the models are trained on the k-best selected features. The figure 9 shows the RMSE score for each models using both feature selection method. It is observed that features selected using the random forest method are performing better than the Chi2 method. The bar plot in fig: 8 shows the 20 best features in order of their importance calculated using the random forest algorithm. From the figure 9 it is also observed that models trained using feature selection method were performing poor than the models trained on all the features.

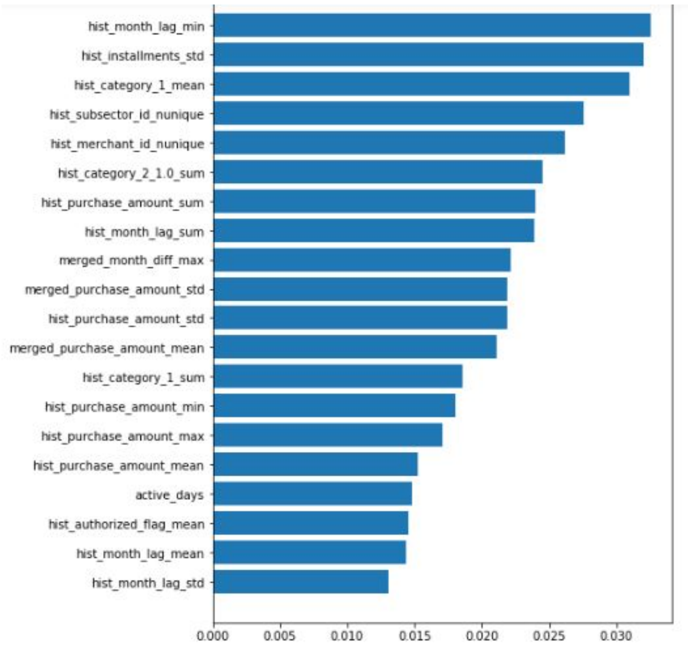
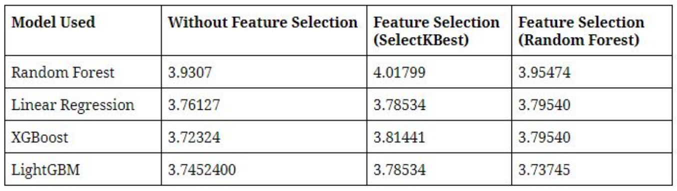


Figure 8. Feature Importance



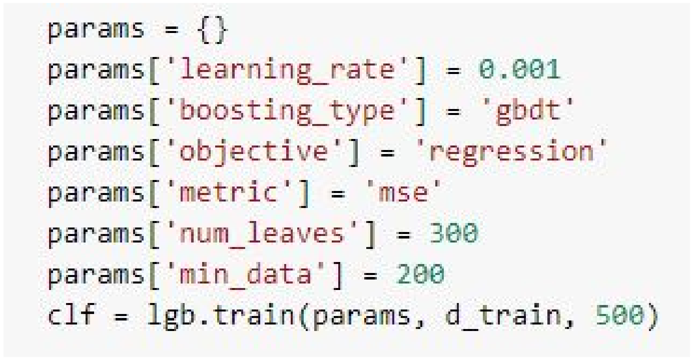
Model Used	Without Feature Selection	Feature Selection (SelectKBest)	Feature Selection (Random Forest)
Random Forest	3.9307	4.01799	3.95474
Linear Regression	3.76127	3.78534	3.79540
XGBoost	3.72324	3.81441	3.79540
LightGBM	3.7452400	3.78534	3.73745

Figure 9. RMSE score with and without feature selection methods

C. Experimental Scenario: Gradient Boosting Algorithms

Since, the basic models performed poorly after training on the k-best features. So, we trained the random forest and Linear regression model on all the features but it is observed that they are computationally expensive algorithm and taking a lot of time to run. So, we used Light GBM and XGB model which are gradient boosting based algorithm and can handle large amount of data and takes lower memory to run. Light GBM also supports GPU learning. The fig: 10 shows the parameters used to train the Light GBM model.

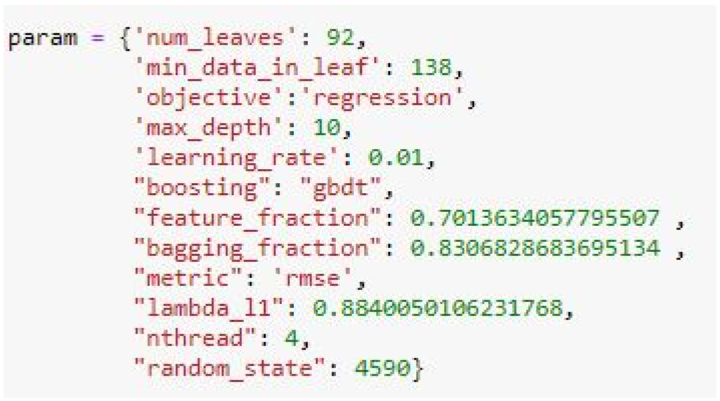
Now, to improve the performance of the Light GBM model, we tuned the hyper-parameters of the Light GBM model using the Bayseian Optimization. We did not use Grid search method because it requires long run time and it tries all the possible hyper-parameters for tuning the parameters. The fig: 11 shows the tuned parameters of the Light GBM Model. After hyper-parameter tuning, the Light GBM model trained on all the 144



```
params = {}
params['learning_rate'] = 0.001
params['boosting_type'] = 'gbdt'
params['objective'] = 'regression'
params['metric'] = 'mse'
params['num_leaves'] = 300
params['min_data'] = 200
clf = lgb.train(params, d_train, 500)
```

Figure 10. Hyper-parameters of Light GBM Model

features with five fold cross validation. It is observed that the performance of the Light GBM is improved significantly. The table II shows the RMSE score of XGB and Light GBM using various methodologies.



```
param = {'num_leaves': 92,
         'min_data_in_leaf': 138,
         'objective':'regression',
         'max_depth': 10,
         'learning_rate': 0.01,
         "boosting": "gbdt",
         "feature_fraction": 0.7013634057795507 ,
         "bagging_fraction": 0.8306828683695134 ,
         "metric": 'rmse',
         "lambda_l1": 0.8840050106231768,
         "nthread": 4,
         "random_state": 4590}
```

Figure 11. Hyper-parameters of Light GBM Model after Bayesian Optimization

Table II
RMSE SCORE USING VARIOUS METHODOLOGIES

Model	RMSE
XGB Model	3.74524
Light GBM model	3.72324
Light GBM Model with hyper-parameter tuning	3.64631
XGB Model without Outliers	3.82386
Light GBM Model without Outliers	3.76828

D. *Experimental scenario: Remove Outliers*

From figure III-B it is observed that the outliers are present in the training dataset. Until this point, we trained the model without removing the outliers. In this section we trained the Light GBM and XGB model after removing the outliers. The Light GBM model is trained on the tuned hyper-parameters. The table II shows the performance of the models after removing the outliers.

CONCLUSION

The paradigm discussed in the above report shows the ensemble models like Light GBM and XGB performs well on the large dataset. It was also observed that performance of the Light-GBM improved after tuning the hyper-parameters. It shows that hyper-parameters play an important role in the performance of the model.

It is also observed that models trained after removing the outliers reduced the performance of the models. The reason can be that the data is fictious. All the features are strongly normalized. So, may be the data points which we considering as outliers are not outliers in the real dataset.
Feature engineering plays an important role in the performance of the model. But the data is fictious and most of the features are anonymized, so this make it difficult for us to interpret the data.

CONTRIBUTION

- **Contribution made by Shivani:**
  - The analysis of historical_transaction.csv and new_merchant_transaction.csv data files.
  - Trained various models on Train.csv for building the baseline for further models.
  - Feature engineering on historical and new merchant data and generate new features using them.
  - Trained Linear regression model on all the generated features.
  - Selected k-best features using the Chi2 method and trained various model on these k-best features.
  - Built Light GBM model and tuned its hyper-parameters using the Bayesian Optimization.
  - Trained XGB model and Light GBM model without outliers.
- **Contribution made by Harshita:**
  - The analysis of merchant.csv and train.csv data files.
  - Feature engineering on merchant and train data and generate new features using them.
  - Trained Random forest and XGB Boost algorithm on all the generated features.
  - Trained Random Forest algorithm for selecting the k-best features and trained various models on these k-best features.

REFERENCES

[AHC19] Suhaib Aslam, Guido Heijden, and Harry Collins. "Predicting Customer Loyalty Using Various Regression Models". In: (2019).

[MKS19] M R Machado, S Karray, and I T de Sousa. "LightGBM: an Effective Decision Tree Gradient Boosting Method to Predict Customer Loyalty in the Finance Industry". In: *2019 14th International Conference on Computer Science Education (ICCSE)*. Aug. 2019, pp. 1111–1116. DOI: 10.1109/ICCSE.2019.8845529.