

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

SC4000/CZ4041 - Machine Learning

Group Project Report

Project : ELO Merchant Category Recommendations

Name	Contributions
Cholakov Kristiyan Kamenov U2123543B	Exploratory Data Analysis, Model Selection
Mishra Pradyumn U2123912E	Data Preprocessing, Data Cleaning
Najah Ismail U2120555F	Feature Extraction
Denzyl David Peh U2122190F	Model Selection and Training
Kim Chae Yoon U1923512A	Hyperparameter tuning
All Members	Report and Video Presentation

Table of Contents

1. Problem Statement	2
2. Exploratory Data Analysis	2
2.1 Train & Test Data	3
2.2 Historical and New Merchant Transactions Data	5
2.3 Merchants Data	8
3. Methodology	10
3.1 Data Preprocessing	11
3.2 Feature Extraction	11
3.2 Model Training	14
3.2.1 Model Selection	14
3.2.2 Hyperparameter Tuning	14
3.2.2.1 Number of leaves	14
3.2.2.2 Feature fractions	15
3.2.2.3 Bagging fractions	16
4. Final Results	16
5. Conclusion	17
6. References	18

1. Problem Statement

In the growing financial service industry, fostering customer loyalty is pivotal for enterprises, particularly in the payment industry where numerous options vie for consumer attention as well as retention. ELO Merchant Category Recommendation Kaggle Competition urges the analysis and modeling of customer lifecycle data, including transaction history and demographic information, to accurately predict loyalty scores for cardholders. The primary goal is to reduce unwanted campaigns and create a more targeted, personalized and satisfying experience for customers. Challenges include managing extensive and heterogeneous customer data, requiring preprocessing and feature engineering techniques, as well as interpreting data post-anonymization to maintain privacy and regulatory compliance.

Customer Loyalty is quantified as customer loyalty score. Customer Loyalty is a metric that encapsulates the likelihood of a customer remaining engaged with a brand or service overtime.

The evaluation metric used for analysis is Root Mean Square Error (RMSE), defined as:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n \left(y_i - \hat{y}_i \right)^2}$$

Where \hat{y} is the predicted loyalty score for each card and y is the actual loyalty score for an individual card_id.

2. Exploratory Data Analysis

In the exploratory data analysis (EDA) section, the data is detailedly analyzed in order to get a better understanding of its distribution, structure and characteristics. The analysis aims to detect the patterns in the data and to check the correlation between the different features. In total, we have 5 main files (given in the Kaggle data section of the competition) which provide us with the raw data, later used for the preprocessing, feature extraction and training. The meaning of the various features are explained in tables 2.1, 2.2 and 2.3. For the sake of confidentiality, several columns are anonymous, their names are abstract and there is not any description provided for them. Popular and widely used libraries such as Pandas, Matplotlib and Seaborn were used for processing the data, calculating its characteristics and plotting the results.

Due to the length limitations of the report, we will present only the figures showing interesting insights about the patterns and the correlations between the features. All feature distributions will be displayed in the EDA section of our project video.

Table 2.1 Description of merchant data

Variable Name	Description
merchant_id	Unique merchant identifier
merchant_group_id	Merchant group (anonymized)
merchant_category_id	Unique identifier for merchant category (anonymized)
subsector_id	Merchant category group (anonymized)
numerical_1	Anonymized measure
numerical_2	Anonymized measure
category_1	Anonymized category
most_recent_sales_range	Range of revenue (monetary units) in last active month $\rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$
most_recent_purchases_range	Range of quantity of transactions in last active month $\rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$
avg_sales_lag3	Monthly average of revenue in last 3 months divided by revenue in last active month
avg_purchases_lag3	Monthly average of transactions in last 3 months divided by transactions in last active month
active_months_lag3	Quantity of active months within last 3 months
avg_sales_lag6	Monthly average of revenue in last 6 months divided by revenue in last active month
avg_purchases_lag6	Monthly average of transactions in last 6 months divided by transactions in last active month
active_months_lag6	Quantity of active months within last 6 months
avg_sales_lag12	Monthly average of revenue in last 12 months divided by revenue in last active month
avg_purchases_lag12	Monthly average of transactions in last 12 months divided by transactions in last active month
active_months_lag12	Quantity of active months within last 12 months
category_4	Anonymized category
city_id	City identifier (anonymized)
state_id	State identifier (anonymized)
category_2	Anonymized category

Table 2.2 Description of historical transactions and new merchant data

Variable Name	Description
card_id	Card identifier
month_lag	Month lag to reference date
purchase_date	Purchase date
authorized_flag	'Y' if approved, 'N' if denied
category_3	Anonymized category
installments	Number of installments of purchase
category_1	Anonymized category
merchant_category_id	Merchant category identifier (anonymized)
subsector_id	Merchant category group identifier (anonymized)
merchant_id	Merchant identifier (anonymized)
purchase_amount	Normalized purchase amount
city_id	City identifier (anonymized)
state_id	State identifier (anonymized)
category_2	Anonymized category

Table 2.3 Description of train and test data

Column Name	Description
card_id	Unique card identifier
first_active_month	'YYYY-MM', month of first purchase
feature_1	Anonymized card categorical feature
feature_2	Anonymized card categorical feature
feature_3	Anonymized card categorical feature
target	Loyalty numerical score calculated 2 months after historical and evaluation period

2.1 Train & Test Data

The files containing the train and test data: “train.csv” and “test.csv” follow the same structure, containing columns for 3 different features and the first active month for a credit card. The training data consists of 201 917 card records, while the test one holds 123 623 ones. Figure 2.1 presents the distribution of the 3 features. As we can see, the distribution of the test data follows the training one, meaning they both follow the same trends. Similarity can also be seen between the first active month distribution in Figure 2.2, the two time series share the same drops and peak, the only difference is the number of cards but this is expected as the test data has a smaller size.

Also, as seen in Figure 2.2, the earliest activated cards from the collections date back to 2012, while the most recently activated ones are from the end of 2018.

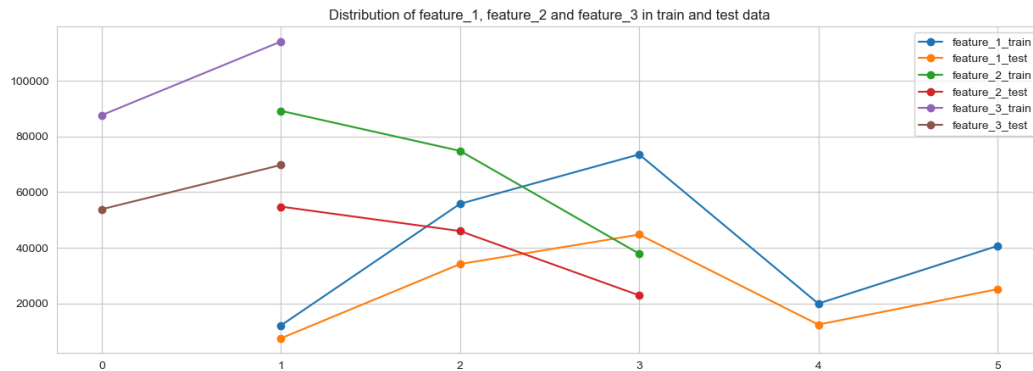


Figure 2.1 The distribution of feature_1, feature_2 & feature_3

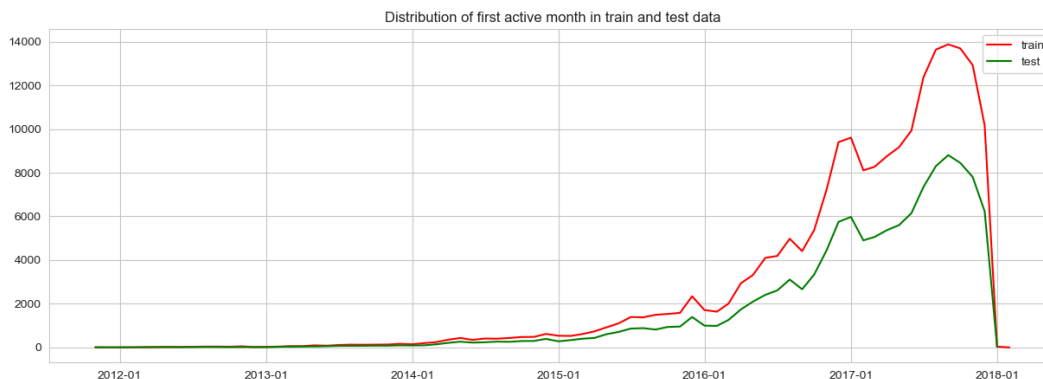


Figure 2.2 The distribution of the first active month for the train and test data

Lastly, we will analyze the correlation matrix (Figure 2.3) between the features. As we can see, the most significant correlation is the positive correlation between feature_1 and feature_3. However, because of the lack of description for the features and the medium values for their correlation, we cannot gain more detailed insights than the initial increase in their distribution which is also seen in Figure 2.1.

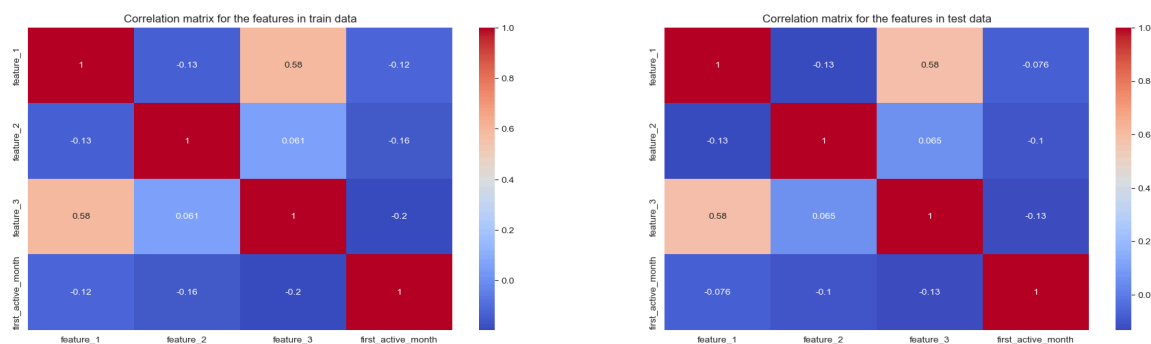


Figure 2.3 The correlation matrices for the features in the train and test data

2.2 Historical and New Merchant Transactions Data

The data for the previous (historical) and the new transactions is given in “new_merchant_transactions.csv” and “historical_transactions.csv”. We are analyzing them together because we spotted their similar column structure after the performed analysis. The columns, the number of unique values in them and their corresponding range can be found are presented in Table 2.4.

Table 2.4 “new_merchant_transactions.csv” and “historical_transactions.csv” features structure

Column Name	Description	Unique Values (Historical)	Value Range (Historical)	Column Name	Unique Values (New Merchant)	Value Range (New Merchant)
authorized_flag	Flag indicating authorization	2	['Y', 'N']	authorized_flag	1	['Y']
card_id	Card identifier	325540	id	card_id	290001	id
category_1	Categorical category	2	['N', 'Y']	category_1	2	['N', 'Y']
category_2	Categorical category	5	1.0 - 5.0	category_2	5	1.0 - 5.0
category_3	Categorical category	3	['A', 'B', 'C', 'nan']	category_3	3	['B', 'nan', 'C', 'A']
city_id	City identifier	308	id	city_id	308	id
installments	Number of installments	15	-1 - 999	installments	15	-1 - 999
merchant_category_id	Merchant category identifier	327	id	merchant_category_id	314	id
merchant_id	Merchant identifier	326311	id	merchant_id	226129	id
month_lag	Month lag to reference date	14	-13 - 0	month_lag	2	1 - 2
purchase_amount	Normalized purchase amount	215014	-0.7469078 - 6010603.9717525	purchase_amount	75190	-0.74689277 - 263.15749789
purchase_date	Purchase date	16395300	2017-01-01 00:00:08 - 2018-02-28 23:59:51	purchase_date	1667025	2017-03-01 03:24:51 - 2018-04-30 23:59:59
state_id	State identifier	25	id	state_id	25	id
subsector_id	Merchant category group identifier	41	id	subsector_id	41	id

The “month_lag” feature serves as a marker for the number of months that elapsed from the specified reference date. As seen in Figure 2.4, the new transactions appear to extend up to 2 months after the reference date, indicating the continuous activity for the period post to the designated date. On the other hand, the historical ones are recorded up to 13 months prior to it.

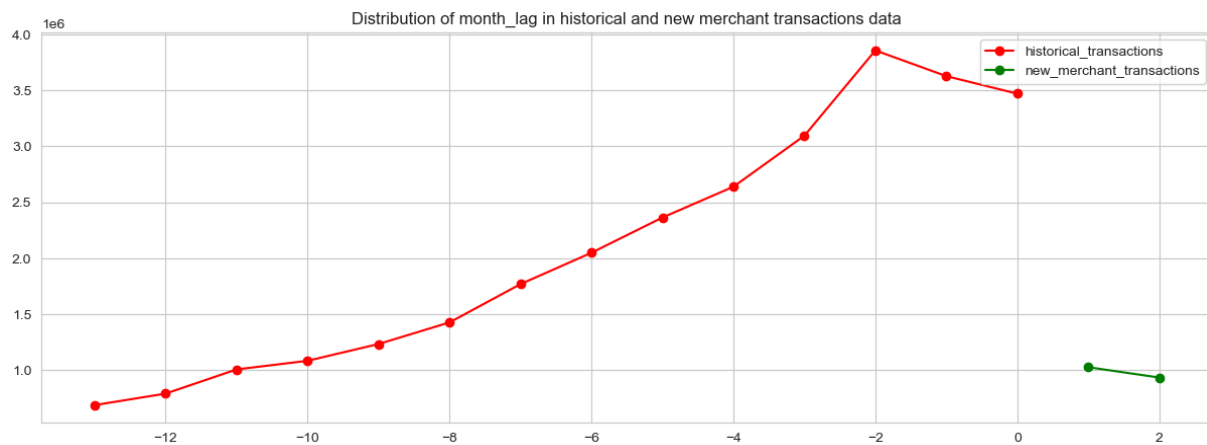


Figure 2.4 The distribution of month_lag for historical and new transactions

Figure 2.5 shows that both the new and the historical transactions follow the same distribution of the number of installments. This suggests that the installments-based behavior remains the same, irrespective of whether the transactions took place in the period before or after the reference data. Furthermore, both the historical and the new transactions show the peak in transactions with low number of installments. The values at the end: “999” and “-1” represent wrong values and refunded transactions, respectively.

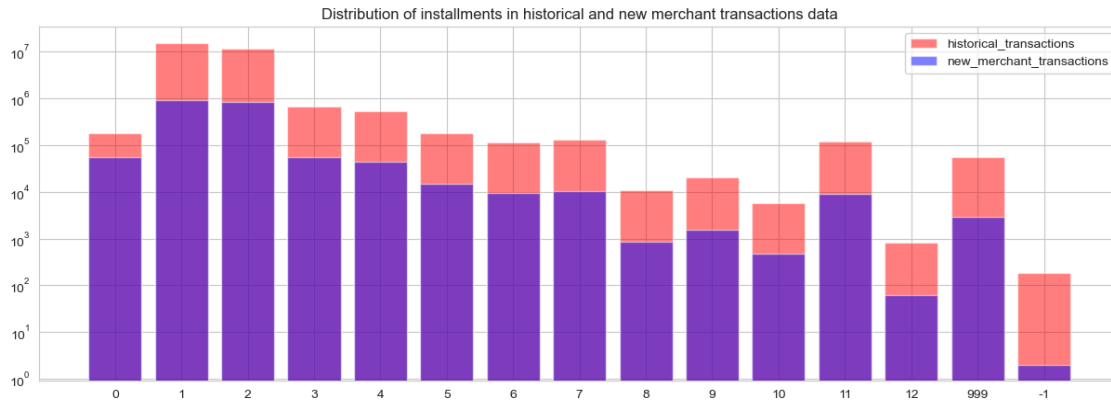


Figure 2.5 The distribution of installments for historical and new transactions

The 2 month lag difference in Figure 2.4 is spotted in Figure 2.6 too, as we see the transactions follow the same trends but the new transactions range start with a 2 month delay. Additionally, despite some cards being activated as early as 2012, as shown in Figure 2.2, the earliest transactions recorded within the collections only trace back to the beginning of 2017.

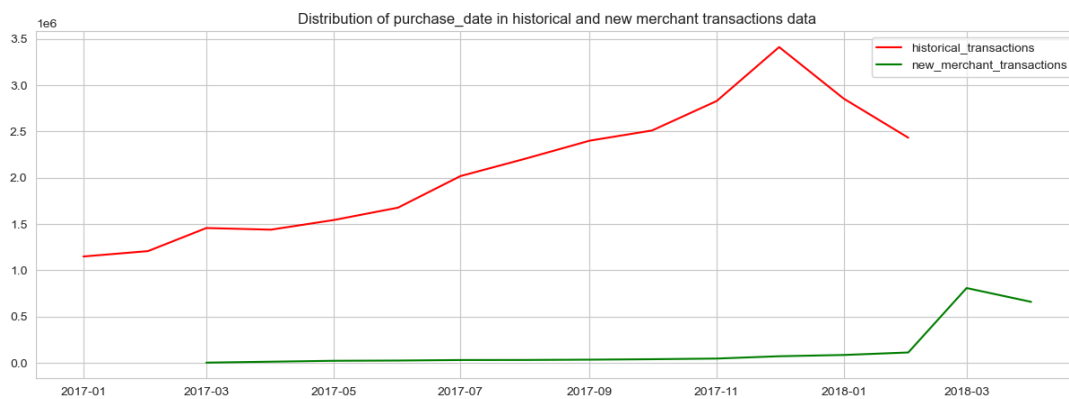


Figure 2.6 The distribution of purchase_date (month) for historical and new transactions

As for the authorized flag distribution, from Table 2.4 and Figure 2.7, we observe that the new dataset consists of only authorized transactions.

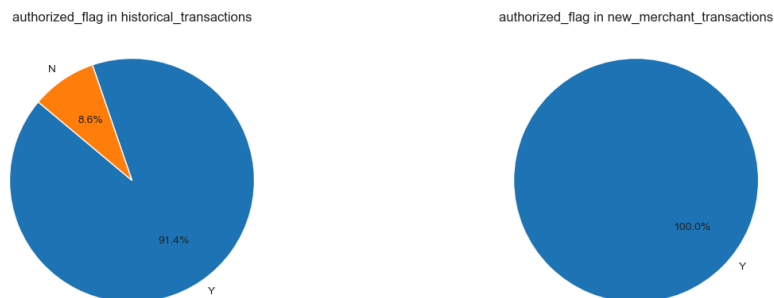


Figure 2.7 The distribution of authorization_flag for historical and new transactions

The biggest difference between the two collections can be seen in the distribution of the `purchase_amount` feature (seen in Figure 2.8), where the historical transactions have a much wider range of values compared to the new ones.

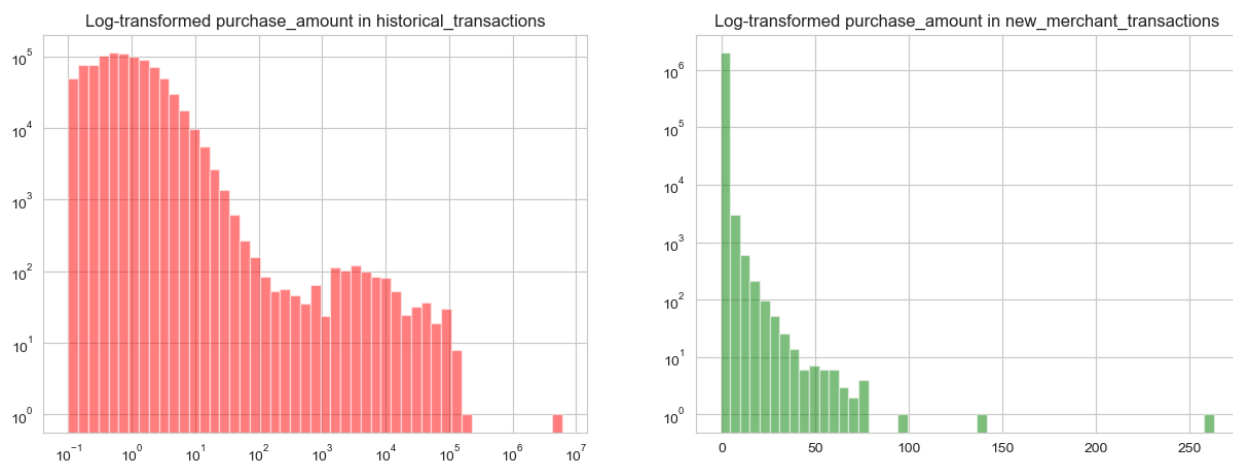


Figure 2.8 `purchase_amount` - `historical_transaction` vs `new_merchant_transactions`

The correlation matrices for the two transactions' collections are presented in Figure 2.9. As we can see, the matrix for the `new_merchant_transactions` is missing the row and the column for the `authorization_flag` feature. This is a result of the collection consisting of only authorized transactions. As for the correlation results, the most correlated features in `historical_transactions` are `purchase_date` and `month_lag`, while the ones in the new transactions collection are `installments` and `category_3`. The correlation between `month_lag` and `purchase_date` is expected, as the two features are directly connected, `month_lag` is the number of elapsed months since the purchase date. However, this correlation is not present in the new transactions collection. This may suggest that `month_lag`'s definition deviates from the `purchase_date` value in the `new_merchant_transactions` data.

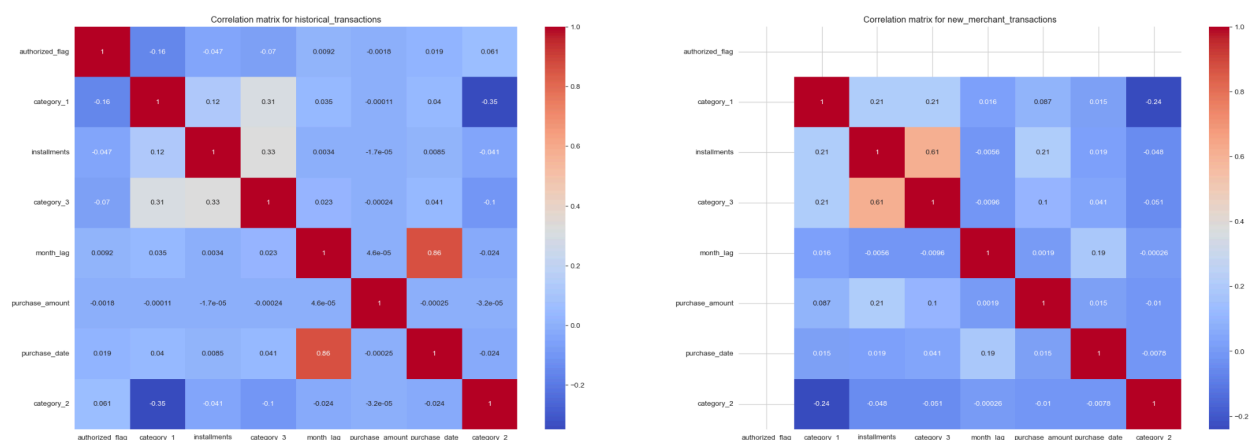


Figure 2.9 The correlation matrices for the transactions datasets' features

2.3 Merchants Data

The merchants' data is stored in the “merchants.csv” file and represents the additional data for all merchants. The characteristics of the different features are shown in Table 2.5.

Table 2.5 “merchants.csv” features structure

Merchants	Merchants_num_unique	Merchants_values
0	merchant_id	334633, id
1	merchant_group_id	109391, id
2	merchant_category_id	324, id
3	subsector_id	41, id
4	numerical_1	954, -0.05747065 - 183.73511137
5	numerical_2	947, -0.05747065 - 182.07932234
6	category_1	2, ['N', 'Y']
7	most_recent_sales_range	5, ['E', 'C', 'D', 'A', 'B']
8	most_recent_purchases_range	5, ['E', 'D', 'C', 'A', 'B']
9	avg_sales_lag3	3372, -82.13 - 851844.64
10	avg_purchases_lag3	100003, 0.33349533 - inf
11	active_months_lag3	3, 1 - 3
12	avg_sales_lag6	4507, -82.13 - 1513959.0
13	avg_purchases_lag6	135202, 0.16704466 - inf
14	active_months_lag6	6, 1 - 6
15	avg_sales_lag12	5009, -82.13 - 2567408.0
16	avg_purchases_lag12	172917, 0.09832954 - inf
17	active_months_lag12	12, 1 - 12
18	category_4	2, ['N', 'Y']
19	city_id	271, id
20	state_id	25, id
21	category_2	5, 1.0 - 5.0

During the analysis of the features, we found that the features “numerical_1” and “numerical_2” have almost the same distribution as can be seen in Figure 2.10. The majority of the difference between the records' values for the two features spreads in the range between 0 and 1, showing that the features are extremely similar.

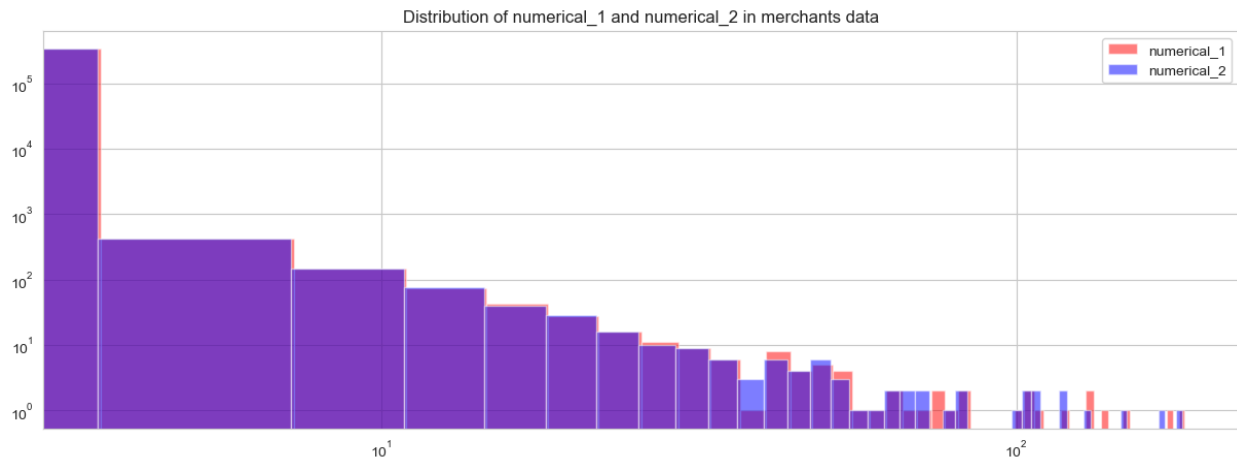


Figure 2.10 numerical_1 vs numerical_2 distribution

Furthermore, “avg_purchase_lag3”, “avg_purchase_lag6” and “avg_purchase_lag12” also show very similar distributions of their values; they are almost completely overlapping as seen in Figure 2.11.



Figure 2.11 “avg_purchase_lag3”, “avg_purchase_lag6” and “avg_purchase_lag12” distribution

Finally, we can see that “avg_months_lag12” holds the values from “avg_months_lag3” and “avg_months_lag6” as they overlap (shown in Figure 2.12).

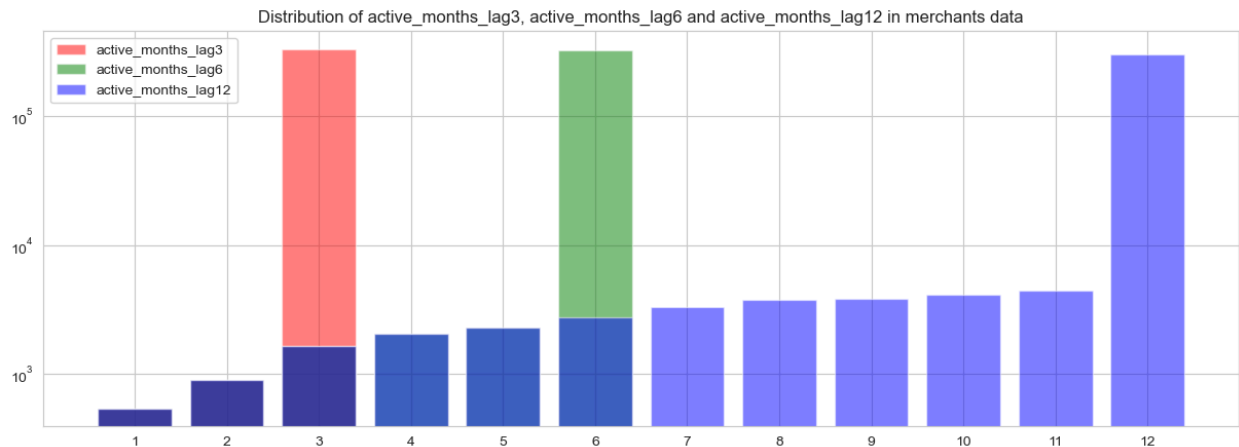


Figure 2.12 “active_months_lag3”, “active_months_lag6” and “active_months_lag12” distribution

The above analysis is confirmed in Figure 2.13. The overlapping of the distributions of numerical_1 and numerical_2 has resulted in the red 2x2 square at the top left corner of the matrix. The correlation value of 1 for all the cells in it proves that the two features follow the same pattern. Also, similar 2x2 squares are formed for the most recent purchases and sales ranges. Furthermore, the correlation between the active months lag, the average purchase lag and the average sales lag can be spotted in the orange cells in the submatrix for the lag features. The high number of cells in the “warm” color range proposes that all these features are highly correlated and follow the same trends. The red cells show the almost-perfect similarity in the distribution between the average purchase lag features. To summarize, the merchants' dataset consists of many closely related features, mostly because of their definition and the way of their calculation.

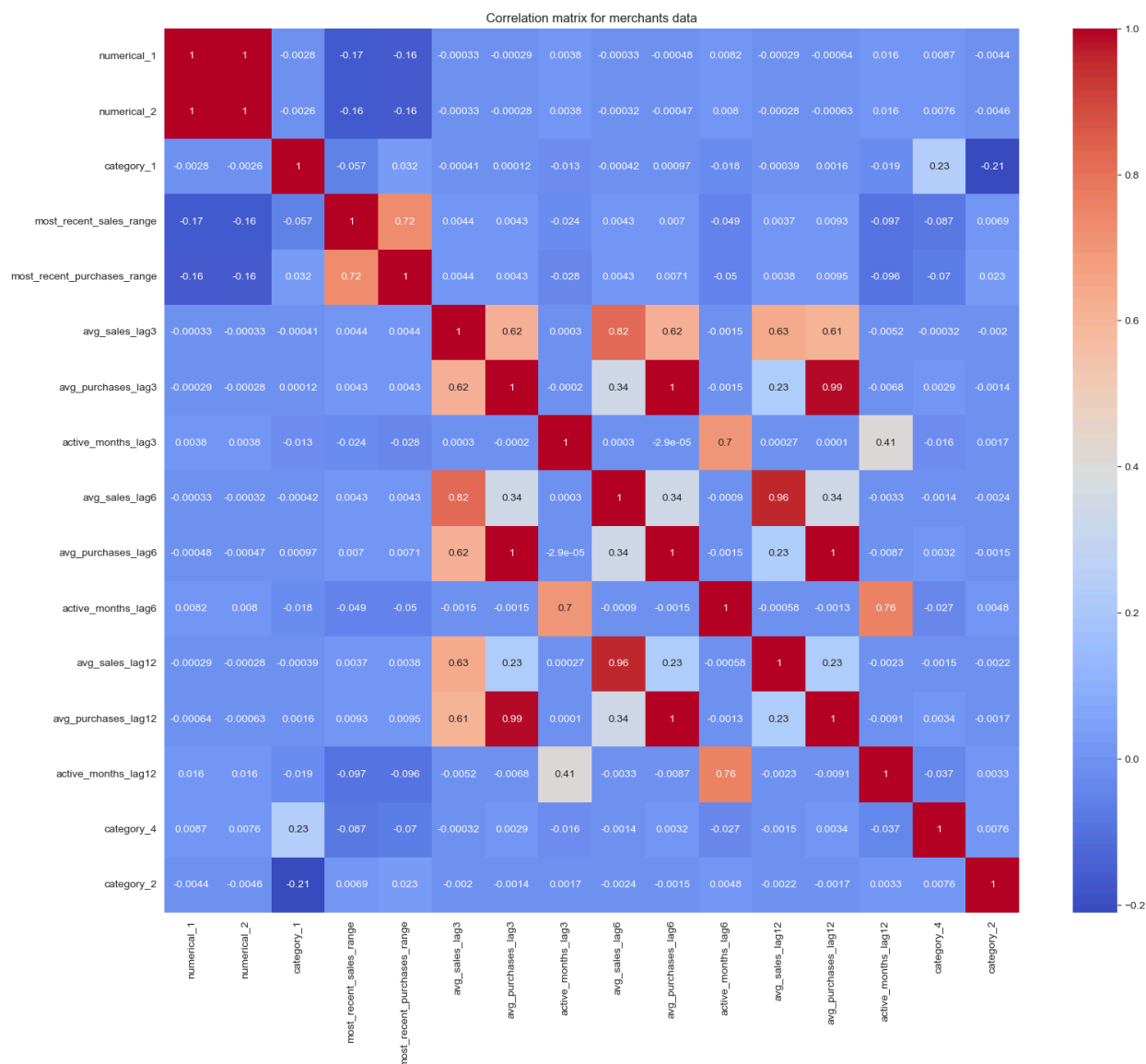


Figure 2.13 The correlation matrix for the merchants dataset's features

3. Methodology

We used a novel training approach to predict the customer loyalty from the raw data provided. The approach was divided into 4 pivotal phases: Data Pre-Processing, Feature Extraction, Model Training, and Results. The overall structure of the proposed solution is presented in Figure 3.1. Because of the highly competitive nature and the Elo Merchant Category Recommendation competition, we were pursuing an approach with very high accuracy. Existing approaches rely on the combination of multiple models (more than 10), making them extremely time and resource inefficient. Thus, we focused our solution on the careful selection and engineering of features. As you can see, the proposed methodology consists of only 3 models which are trained on only the carefully selected features with the highest importance.

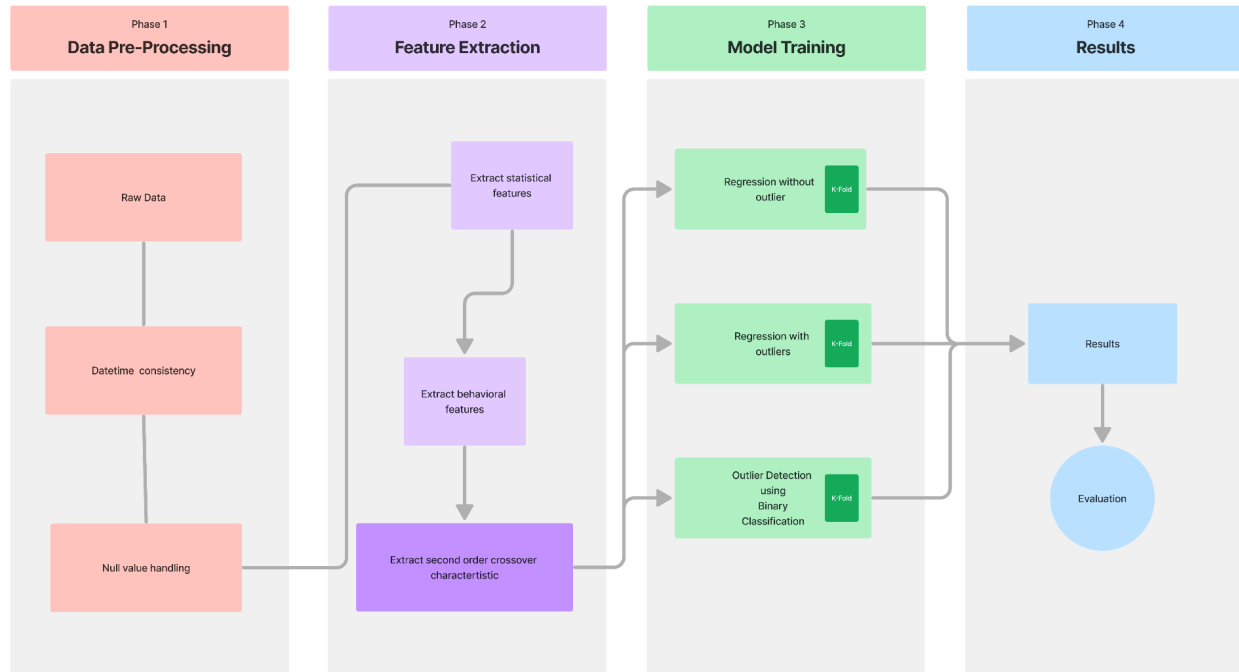


Figure 3.1 Overview of methodology

3.1 Data Preprocessing

After importing the datasets, the data needs to be cleaned by filling in values that do not fall in the data range for specific features. An example of this is the ‘installments’ feature which is defined as the number of installments paid for a particular purchase. However this column includes values like -1 and 999, which are not valid installments, hence they were replaced by NaN. Date features are also split into their respective day, week, month, and year. Other features that have null values, such as Category 2, 3 and merchant ID, can be filled by placeholder values so as to not lose the data point.

3.2 Feature Extraction

We define an aggregate function for extracting basic statistical features of various transaction-related attributes like purchase amount, purchase date, and installments, such as the count, mean, sum, standard deviation, median, maximum, and minimum values. These basic statistical features provide general descriptions of the data.

After that, we construct behavioral features for each `card_id` based on the `historical_transaction` data over the entire time frame, and again for only the last 2 months. This is important because user behavioral features are known to be very effective for improving prediction accuracy [1], especially more recent features. We compute features such as the time difference between the first and most recent transaction, average time interval between transactions, and the 2.5%, 25%, 75% and 97.5% quantiles for purchase amounts.

We then compute second-order features using the given features and behavioral features obtained earlier, by calculating aggregates such as number of unique `card_id` or sum of purchase amount for each pair of specified columns. For each of these

aggregates, we compute the mean, max and standard deviation. Second-order crossover characteristics allow the model to capture interactions between different features, even if they might not seem to be important on their own. This enhances the feature representation and thus improves model performance.

We also noticed that the distribution of target values has a distinct second peak, as seen in Figure 3.2 below. This contains 2207 data points, which is significant and should not be ignored. To handle this, we create another 'outlier' column, assigning a value of 1 if the target value is less than -30, and otherwise assigning a value of 0. This will be used when training our outlier detection model.

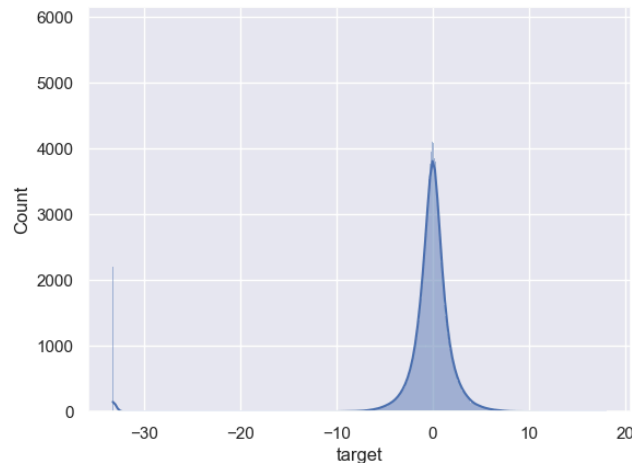


Figure 3.2 Number of data points vs target in train

At this point, we have a total of 797 features. We use the FeatureSelector tool [2] to identify the proportion of missing values for each feature, and the zero-importance features. Zero-importance features will likely not improve prediction power, but rather increase the amount of computation required and might also introduce noise. Removing these features will reduce the amount of time taken for training, and might reduce overfitting.

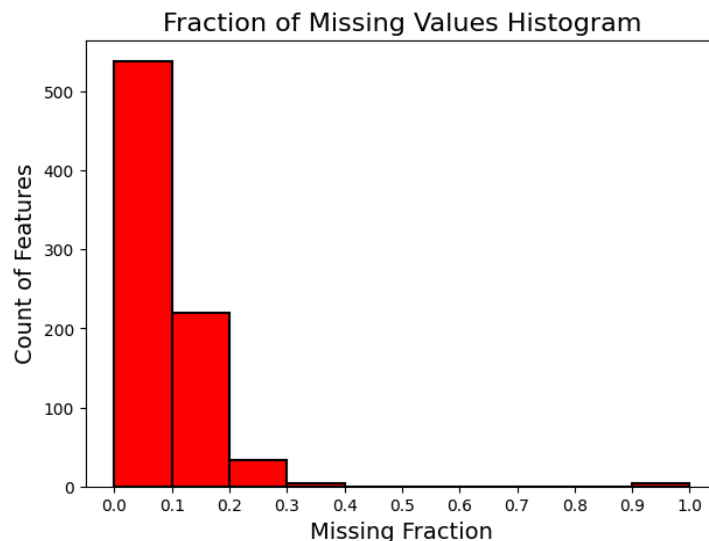


Figure 3.3 Number of features vs. Missing fraction

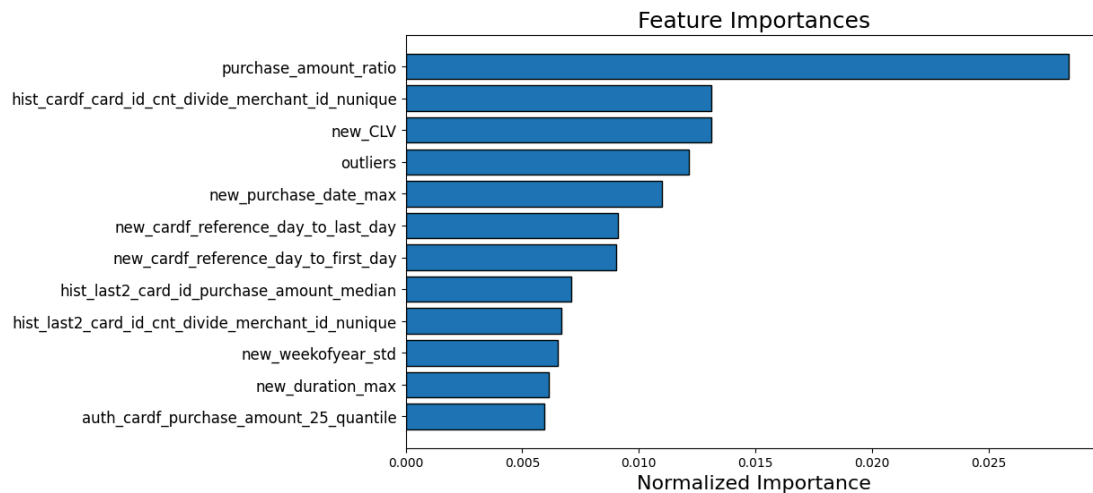


Figure 3.4 Top 12 Feature importances extracted from LGBM

We filter out any features with more than 60% missing values (Figure 3.3), as well as the zero-importance ones. After dropping 4 features with almost entirely missing values, as well as 145 zero-importance features, we have only 648 features remaining. According to FeatureSelector, 651 features are required for 99% cumulative importance. This is shown in Figure 3.5 below.

Among the zero-importance features, all of the given features except for feature 1 and feature 2 were dropped. This is consistent with the EDA performed earlier, however it should be noted that most second-order features derived from the dropped features are still kept.

Moreover, the high importance of the outlier feature leads us to use different approaches towards instances classified as outlier and those not.

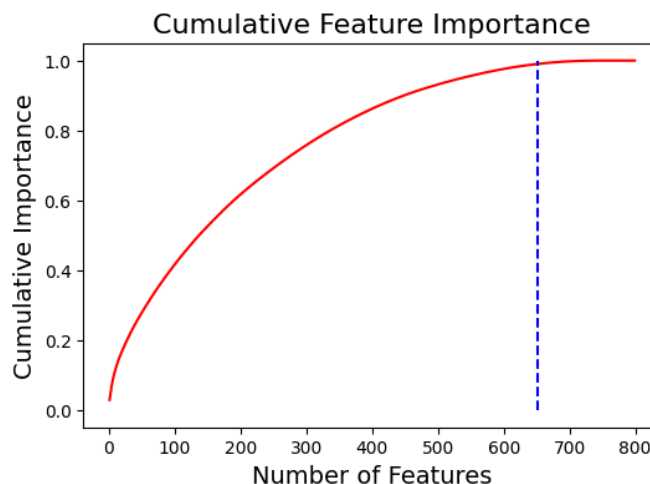


Figure 3.5 Graph of Cumulative Feature Importance

3.2 Model Training

3.2.1 Model Selection

For our methodology, we require 2 models: One to perform regression to predict the customer loyalty as well as a binary classifier to predict the outliers. During our experimentation and EDA, we observed the Light Gradient Boosting Machine (LGBM) provides better results. Not to mention with the high dimensionality of our data, LGBM can better handle this data. LGBM is a gradient boosting framework that uses tree-based learning algorithms to solve both classification and regression, making it suitable for our determined methodology. The high dimensionality and large number of data points makes this dataset suited for LGBM as it is very efficient compared to other models for large datasets.

Our approach involves training 2 regression models over 2 datasets. The first dataset includes all of the data included in the train data provided. The second regression model is trained over a dataset that does not contain the outliers as discussed in section 3.1. This makes this model suited for predicting data that can be identified as not an outlier. The first model, which is trained on outliers as well as non-outliers, would be better at predicting the value for an outlier than the other model. If we are able to classify a data point as an outlier, we can use the model that is trained over all data points to provide a more accurate result than the model that is trained without outliers. Similarly, for the rest of the data, which are non outliers, we can use the model trained on only the outliers to extract a better performance.

To classify if a data point is an outlier or not, we are using a binary classifier. This provides a probability of a data point being an outlier or not. We used a threshold of 0.8 to classify a datapoint as an outlier. For all data points in the test dataset, using these predicted outlier values, we can use a specific model to predict more accurate values.

All models are trained for a maximum number of training rounds of 20000. The stopping rounds are set to 500 to prevent overfitting if the model continues training without improvement. RMSE is calculated for the regression tasks for evaluation. Log loss is used for binary classification which measures the performance of the model in terms of its predicted probabilities for the positive class.

3.2.2 Hyperparameter Tuning

The following hyperparameters of the LGBModel were tuned using K-fold cross-validation, where $k = 5$.

3.2.2.1 Number of leaves

The leaves of the tree are the final partitions of the feature space. Each data point is assigned to one of the leaves based on its feature values, and the predicted output is the mean of the target values of all the training samples assigned to that particular leaf.

The number of leaves refers to the maximum number of leaves in one tree. It provides control over the complexity of the tree. While a higher number of leaves ensures better capture of the complexity, it also leads to overfitting. We tested this with the following values: 16, 32, 64 and 128. The final value of num-leaves used is 64, in accordance with the lowest RMSE.

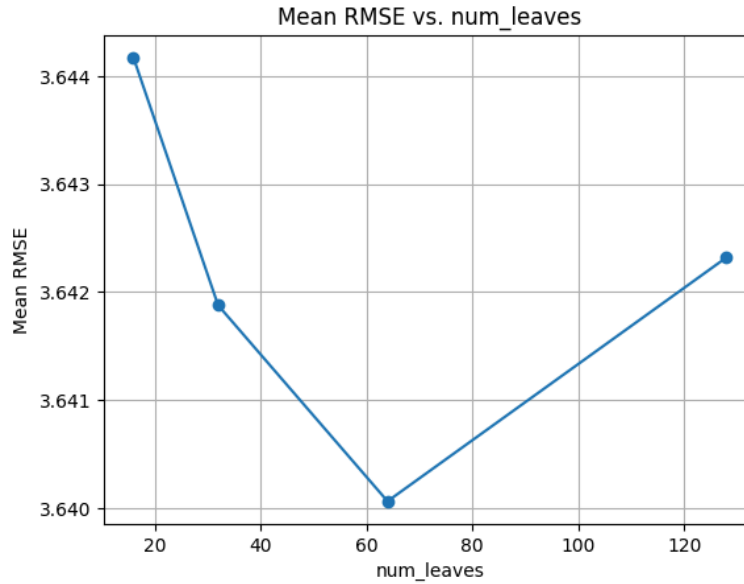


Figure 3.6 Graph of Mean RMSE against num_leaves

3.2.2.2 Feature fractions

Feature fractions refer to the proportion of features randomly selected to build a decision tree in each iteration of the LGB Model algorithm. This technique aims to reduce overfitting by introducing diversity among the trees in the ensemble. Lower feature fractions indicate that fewer features are considered for splitting at each node of the decision tree. This can help prevent the model from memorizing noise in the training data and shift focus to more informative features. However, it may also lead to important patterns in the data being underutilized and may lower the model's predictive performance. We tested this with the following values: 0.2, 0.5, 0.7 and 0.9. The feature fraction value used is 0.5 related to the lowest RMSE.

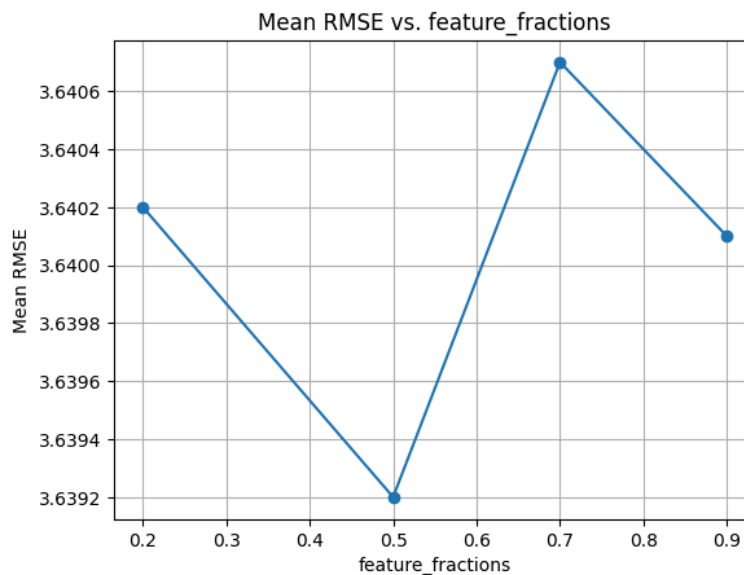


Figure 3.7 Graph of Mean RMSE against feature_fractions

3.2.2.3 Bagging fractions

Bagging fraction or subsample parameter controls the fraction of training data used in each iteration by randomly sampling a subset of the training data for building each tree. This technique aims to reduce variance and overfitting by including some diversity. We tested this with the following values: 0.2, 0.5, 0.7 and 0.9. These values were tested to find an optimal balance between reducing overfitting and capturing variability in the training data. A bagging fraction of 0.9 is used. This means that only 90% of the training data is randomly sampled with replacement for building each tree. This means that each tree is trained on a slightly different subset of training data.

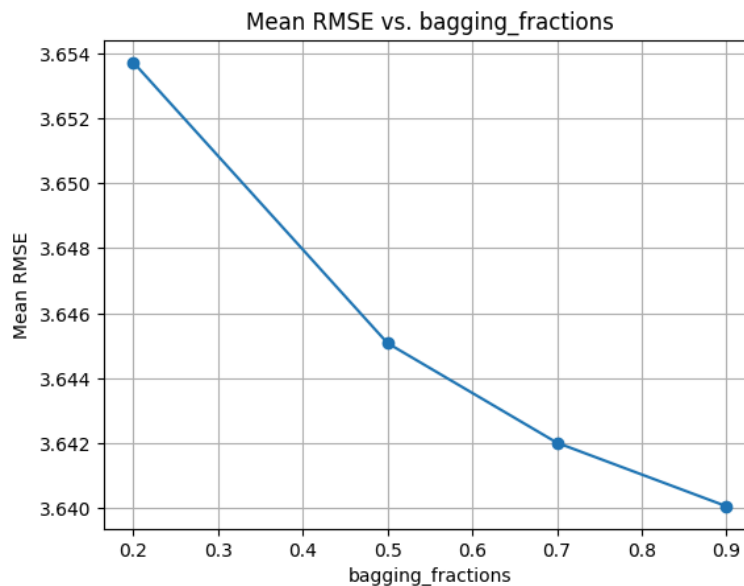


Figure 3.8 Graph of Mean RMSE against bagging_fractions

The final hyperparameters are shown in Table 3.1 below. We used these parameters to train both the binary classifier and regression models.

Table 3.1 Results of Hyperparameter tuning

Hyperparameter	Value
num_leaves	64
feature_fraction	0.5
bagging_fraction	0.9

4. Final Results

After submitting the output csv file to Kaggle, we obtained a public score of 3.68232. This would have been 274th place on the leaderboard, out of 4111 submissions. This is the top 6.67%. Below are the screenshots of our score, along with the leaderboard of public scores.


Submission and Description	Private Score ⓘ	Public Score ⓘ	Selected
 predictions_lgb.csv Complete (after deadline) · 19h ago	3.61171	3.68232	<input type="checkbox"/>

Figure 4.1 Score obtained after submitting our predictions









272	Excuses			3.68220	55	5y
273	乌龟晒太阳			3.68225	91	5y
274	Thor			3.68233	4	5y
275	HY			3.68236	15	5y

Figure 4.2 Leaderboard scores above and below our score

5. Conclusion

Our model predicts the customer loyalty score of ELO's customers with low RMSE. The method of dissecting the approach towards outliers and non-outliers is a novel approach to handling customer transactional data.

Moreover, through the feature importance diagram, we see that the behavioral characteristics such as aggregate values of purchase amount, date-to-first-purchase, etc. as extracted in our select features proves to be important determinants of loyalty score. The time variables in customer transaction history also stand out. For instance, the week-of-year variable allows you to factor in the increased sales during holiday seasons.

Overall we believe that the time and behavior oriented feature extraction as well as the customized handling of outliers through a separate model is pivotal in the success of predicting customer loyalty.

6. References

- [1] F. Castanedo, 'Improving Machine Learning Models by using Behavioral Data | Snowplow', Improving Machine Learning Models by using Behavioral Data. Accessed: Apr. 25, 2024. [Online]. Available: <https://snowplow.io/blog/improving-machine-learning-models-by-using-behavioral-data/>
- [2] W. Koehrsen, 'WillKoehrsen/feature-selector: Feature selector is a tool for dimensionality reduction of machine learning datasets', feature-selector. Accessed: Apr. 25, 2024. [Online]. Available: <https://github.com/WillKoehrsen/feature-selector?tab=readme-ov-file>