# Predicting Customer Loyalty

Vo Thuc Khanh Huyen*
*Viettel Digital Talent 2021*
(Dated: June 29, 2021)

In this project, we will focus on resolving the problem in the Kaggle challenge named *"Elo Merchant Category Recommendation"*. We will analyze, process data, and use machine learning to improve the understanding of customer loyalty for the Elo payment brand. The results show that LightGBM is the best performing algorithm on the given dataset.

## CONTENTS



## I. INTRODUCTION

Elo is one of the biggest and most reliable payment brands in Brazil. *Elo Merchant Category Recommendation* problem talks about the loyalty score of credit cards for their users in the Elo brand. The loyalty score is calculated in 2 months after the historical and evaluation period. This is a regression task and we use RMSE for evaluation. In this project, we aimed to:

- Analyze, process data to generate many suitable features from our dataset.
- Apply various machine learning models to our dataset and do the assessment on achieved results.

## II. DATA ANALYSING

The Elo dataset contains 5 data tables:

1. **Train:** shape (201917, 6), contains 6 features, which is *card_id, target, first_active_month,* 3 anonymous categorical features are *feature_1, feature_2 and feature_3*. The *target* feature is the loyalty score.
2. **Test:** shape (123623, 5), contains the same feature as train data but the target feature is not present in this dataset.
3. **Historical transaction:** shape (29112361, 14), contains transactions information made up to the reference month.
4. **New transaction:** shape (1963031, 14), contains transactions information made in 2 months after the reference month.
5. **Merchant:** shape (334696, 22), contains additional information about all merchants (*merchant_id*) in the dataset.

We will go into detail for each table data.

1. **Train and test data:** It is good to see that the distribution of *feature_1, feature_2, feature_3,* and *first_active_month* in train and test data are approximately the same (see figure 1, 2). This is vital because the feature *year* and *month* of *first_active_month* have high importance in LightGBM model. Besides, the *target* feature seems to be dominated by a normal distribution (see figure 3), but approximately one percent of the train data is outliers (lie at exactly $-33.21928095$). These outliers cause the model's error to rise to between 3 and 4. These outliers are also present in the test data because when we remove these outliers, it results in a significantly worse test RMSE. We tried various models classify these outliers, but the results are bad. So we continue to train the model with these outliers.

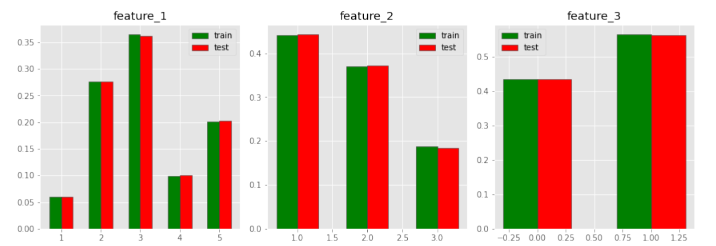


Figure 1: The distribution of features in train and test data

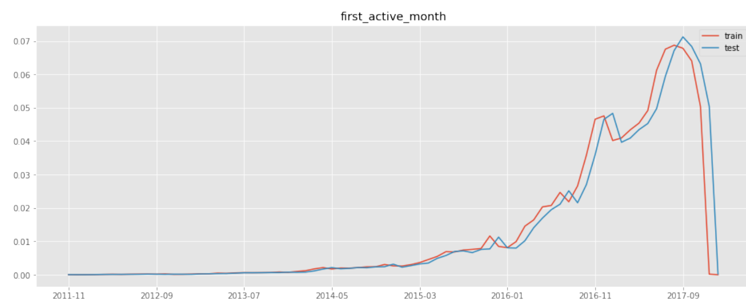


Figure 2: Train - Test - *first_active_month*

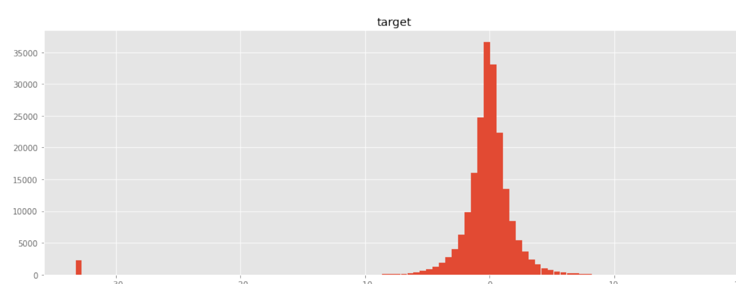


Figure 3: Train - *target*

2. **Historical transaction:** The table stores several transaction records, each of these belongs to one customer (*card_id*). This contains 14 features (see table 5). There are 6 ID type of features and 3 anonymous categorical features. The *purchase_date* is in only year 2017 and 2018, and this only contains the record made from 0 to 13 months before the reference month (*month_lag*). From the *purchase_date*, we can see that transactions are mainly made in afternoon and weekend, so we can generate many valuable features from it. The *authorized_flag* has 2 values 1 and 0, this checks whether a transaction is authorized or not. The *installments* has values from -1 to 12, and 999, we can see that -1 is a illegal values and for value 999, mainly transactions are not authorized. As a result, we will remove the data point that have installment's value -1 or 999, this action improve our model extremely (see figure 4). The *purchase_amount* have a lot of negative values, so we will deal with it in the next section.
3. **New transaction:** The table is highly the same as the historical transaction table. The different is that the record made 1 and 2 month after the reference month. All records are authorized. We also remove the data point that have installment's value -1 or 999 (see figure 5).
4. **Merchants:** We will not use this table in training model, so we don't discuss about it.

* vothuckhanhhuyenvn@gmail.com


Figure 4: Historical transaction features

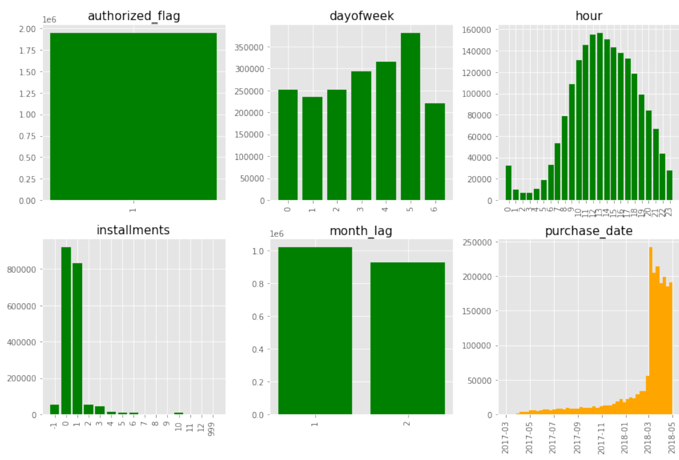


Figure 5: New transaction features

## III. DATA PREPROCESSING

1. **Train and test data:** We don't encode the categorical features (*feature_1, 2, 3*) because we will use LightGBM and CatBoost algorithm, which allow data to have categorical features. We tried to encode them, but the results are not good.

2. **Historical transaction:**

   - Because the file historical_transactions.csv is so large, we use the function *reduce_mem_usage()* in here [1] to reduce about half of memory size.
   - Encode categorical features like *month_lag*, *category_2*, *category_3* using *LabelEncoder* and *get_dummies* from sklearn and pandas.
   - Replace infinite values with nan values. Replace installments which values are -1 or 999 with nan values.
   - Replace nan values from categorical feature with mode and numerical data with the median.
   - Transfer type of *purchase_date* into datetime type using *pandas.to_datetime*.
   - Because the *purchase_amount* have negative values, we will transfer it by the way like here [2] [3].

     $$pa = pa/0.00150265118 + 497.06$$

3. **New transaction:** Doing the same things as the historical transaction table.

## IV. FEATURE ENGINEERING

1. **Train and test data:** We extract *month* and *year* of *first_active_month* feature, and then encode them.

2. **Historical transaction:**

   - ***purchase_date* related features:**
     - Extract several elements of date like *year*, *month*, *weekofyear*, *dayofyear*, *weekofyear*, *day*, *hour*.
     - Check whether the purchase_date is on the weekend or not. We also pay attention to special days in a year, we count the countdown days from the purchase_date and these special days. These are highly important features (see figure 7).
     - We count days difference between two consecutive purchase_date of each customer. We also count months difference between the purchase_date and the day that this contest released.
     - And much more features related to month_lag, ... After this, we *group* historical transaction table to form hist_feats table by *card_id* and use aggregation functions like sum, mean, mode, max, min, std, skew, ...
   - ***purchase_amount* related features:**
     - We *group* historical transaction table by *card_id* using simple aggregation functions on *purchase_amount*. Also, we try to group by 2 features which are *card_id* and one of these features: *category_1, 2, 3*, *installments*, and ID related features.
     - We also consider the ratio of total purchase_amount between 2 month_lag.
   - ***merchant_id* related features:**
     - We define intimate merchant for each customer which is the place that the customer visited at least 2 times. In this situation, the customer and transaction are called an intimate customer and transaction for this merchant.
     - For each merchant, we count the ratio between total intimate customers and total customers, total intimate transactions and total transactions.
   - **Basic information related features:** We consider authorized and unauthorized transactions, count unique values of ID related features, ...

   After *group* table by *card_id*, we will get hist_feats table which have *card_id* as index.

3. **New transaction:** We do the same things as in the historical transaction table, but we don't consider authorized and unauthorized transactions.

## V. MODELLING - ASSESSMENT

1. **Prepare data for training:** We do the same things for train and test table, example for train table:

   - Merge train, hist_feats and new_feats table into one table by using *card_id*.
   - Create features related to both hist_feats and new_feats table by taking ratio, sum, ... (see table 3).

2. **Training model:** We tried 2 models LightGBM and CatBoost. These two models allows to pass categorical features into train data. For each model, we tried to pass encoded and unencoded categorical features (*feature_1, feature_2, fetures_3*). We use 5 folds cross-validation for training, we also use property *is_outlier* to split train and validation data. The result are shown in table 1. LightGBM - unencoded model has the best score in CV, private and public. Besides, LightGBM models run much faster than CatBoost models. As a result, we will choose it to be our final model.

| | CV score | Private score | Public score |
|---|---|---|---|
| LightGBM - unencoded | 3.645296 | 3.60952 | 3.69481 |
| CatBoost - unencoded | 3.665821 | 3.62592 | 3.72180 |
| LightGBM - encoded | 3.66342 | 3.61376 | 3.70235 |
| CatBoost - encoded | 3.67428 | 3.62856 | 3.72337 |

Table 1: Model comparison

3. **Final model:** The parameters of the model are found in table 2. The final result we achieve score as rank 108 on the private leaderboard (see figure 6). The importance

score of features are found in figure 7, from this we can see that *purchase_date* is an important features. Source code of this project is in *eloRepository*.

| | |
|---|---|
| objective="regression" | max_depth= 9 |
| boosting="gbdt" | learning_rate=0.005 |
| metric="rmse" | bagging_fraction=1 |
| reg_alpha=0.1 | bagging_freq=1 |
| reg_lambda=20 | feature_fraction=0.2 |
| num_leaves=120 | verbosity=-1 |
| min_data_in_leaf=70 | num_boost_round=10000 |
| min_gain_to_split=0.05 | early_stopping_rounds=200 |
| max_bin=350 | verbose_eval=100 |

Table 2: LightGBM parameters

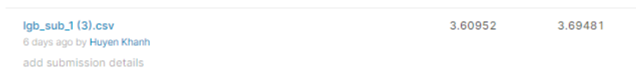


Figure 6: Final result

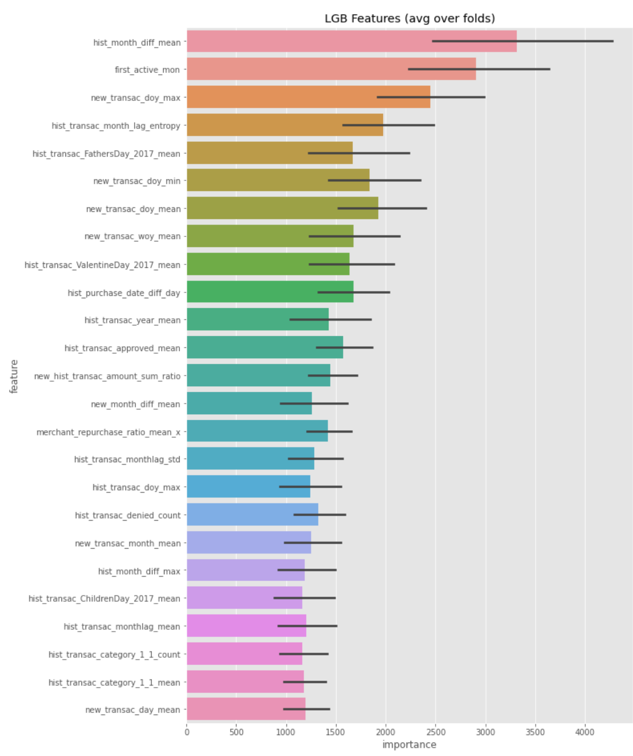


Figure 7: Feature importance

## VI. FUTURE WORK

- We have to be very careful about the preprocessing of data and feature engineering of data.
- Implement more models such as XGBoost, Neural Network, ... We want to try stacking method on this problem.
- We will try to implement a better classification model to classify outliers. The first solution authors tried this way and get 0.015 boosts in local cv [4].

## VII. APPENDIX

| |
|---|
| new_hist_transac_amount_max_ratio |
| new_hist_transac_amount_sum_log_ratio |
| new_hist_transac_amount_max_log_ratio |
| installment_total_sum |
| new_hist_purchase_amount_ratio_1_1 |
| new_hist_purchase_amount_log_ratio_1_1 |
| new_hist_purchase_amount_ratio_1_2 |
| new_hist_purchase_amount_log_ratio_1_2 |
| new_hist_purchase_amount_ratio_2_1 |
| new_hist_purchase_amount_log_ratio_2_1 |
| new_hist_purchase_amount_ratio_2_2 |
| new_hist_purchase_amount_log_ratio_2_2 |
| new_hist_purchase_amount_ratio_3_1 |
| new_hist_purchase_amount_log_ratio_3_1 |
| new_hist_purchase_amount_ratio_3_2 |
| new_hist_purchase_amount_log_ratio_3_2 |
| new_hist_purchase_amount_ratio_4_1 |
| new_hist_purchase_amount_log_ratio_4_1 |
| new_hist_purchase_amount_ratio_4_2 |
| new_hist_purchase_amount_log_ratio_4_2 |
| new_hist_purchase_amount_ratio_5_1 |
| new_hist_purchase_amount_log_ratio_5_1 |
| new_hist_purchase_amount_ratio_5_2 |
| new_hist_purchase_amount_log_ratio_5_2 |
| new_hist_purchase_amount_ratio_6_1 |
| new_hist_purchase_amount_log_ratio_6_1 |
| new_hist_purchase_amount_ratio_6_2 |
| new_hist_purchase_amount_log_ratio_6_2 |

Table 3: Historical and new transactions

| | |
|---|---|
| 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 |

Table 4: train.csv

| | |
|---|---|
| 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 5: historical_transactions.csv

[1] G. Martin, *Reduce Memory Usage* (https://www.kaggle.com/gemartin/load-data-reduce-memory-usage).
[2] raddar, *Purchase amount* (https://www.kaggle.com/raddar/towards-de-anonymizing-the-data-some-insights).
[3] CPMP, *Purchase amount* (https://www.kaggle.com/cpmpml/raddar-magic-explained-a-bit).
[4] 30CrMnSiA, *First solution* (https://www.kaggle.com/c/elo-merchant-category-recommendation/discussion/82036).