**Progress Report**

Fang Feng, Qirui He, Xichen Tan

04/09/2019

# 1 Problem Description

Our project comes from a featured prediction competition on Kaggle[[1]](#footnote-1). The background is that Elo[[2]](#footnote-2) has built partnerships with merchants in order to offer promotions or discounts to cardholders. However, this kind of recommendation system is not specifically tailored for an individual or profile, and Elo wants to know whether their promotions work for consumers and that’s why we need to do this project. In order to develop such an algorithm to identify and serve the most relevant opportunities to individuals, we need to firstly uncover signal in customer loyalty. Basically, the predicted loyalty is the evaluation criteria of this competition and it can reflect how satisfied consumers are with the recommendation system. So, what exactly our goal in this project is to use the datasets provided by Elo and apply the features in them to predict customers’ loyalty. Basically, we assume that all data is simulated and fictitious. Though is not real customer data, it can reflect a real-life scene.

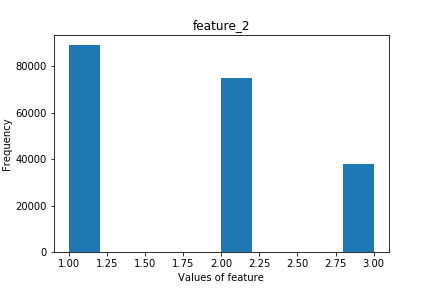
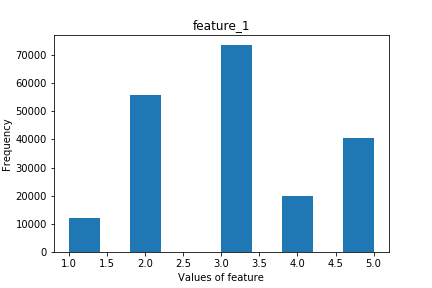
# 2 Data Description

Basically, we have 6 datasets which are all in “.csv” format named “historical\_transactions.csv”, “new\_merchant\_transactions.csv”, “merchants.csv”, “sample\_submission.csv”, “train.csv”, and “test.csv”. They have already been provided by Kaggle. Next, we will introduce these 6 datasets separately.

* “historical\_transactions.csv”: contains up to 3 months’ worth of historical transactions for each *card\_id*. And historical transactions are made before recommendations. It has 29112361 entries and 14 columns (14 features).
* “new\_merchant\_transactions”: contains the transactions at new merchants (*merchant\_ids* that this particular *card\_id* has not yet visited) over a period of two months. These transactions are made after recommendations. It has 1963031 entries and 14 columns (its features are the same as historical\_transactions.csv dataset).
* “merchants.csv”: contains aggregate information for each *merchant\_id* represented in the data set. It has 334696 entries and 22 columns(features).
* “train.csv” and “test.csv”: contain *card\_id* and information about the card itself. “train.csv” has 201917 entries and 6 columns (5 features + 1 target). “test.csv” has 123623 entries and 5 columns (except target column, which is the loyalty we need to predict). Also, since we don’t really join this competition, there is no correct prediction data for us to test, so we can only judge the performance of our model on training dataset.
* “sample\_submission.csv”: a sample submission file in the correct format, so it doesn’t matter here. We just ignore it.

# 3 Preprocessing

## 3.1 Deleting Outliers



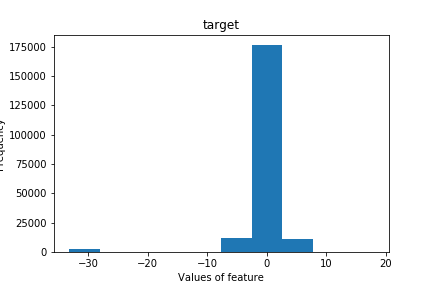
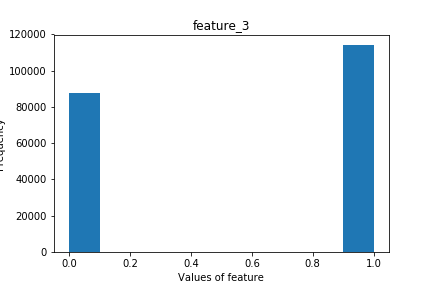


Figure 3.1 Visualization of training data

Visualizing the data from training dataset, and we find that in ‘target’ data, some values of data are far away from mainly distributed area. Thus, we consider deleting these outliers to make better prediction.

## 3.2 Handling missing data

We first drop NA data, which is the data not recorded. Then, filling this empty space with some random features corresponding to its original classification.

## 3.3 Feature Extraction

Basically, our main idea is to make as many features as possible and then fit them all into our training model. After doing this, we can perform feature reduction using feature importance from the model or other techniques such as PCA.

We first convert ‘purchase\_date’ in “historical\_transactions.csv” and “new\_merchant\_transactions” from string to numerical datetime. And then extracting day, month, year and other time-related features out of it. Also, we did some simple normalization, like mapping class ‘Y’ to 1, class ‘N’ to 0.

## 3.4 Feature Aggregation

We find that ‘card\_id’ for each cardholder is unique and we can use it for users’ profile and our feature aggregation. Thus, we can group our data based on ‘card\_id’ and perform calculations on other features. For example, calculating the min, max, mean, std (standard deviation) of the grouped data. Finally, we merge all the extracted and aggregated feature together with our train and test dataset.

# 4 Method

The problem we are trying to solve essentially is a supervised regression problem. Instead of using single model, such as Lasso and Logistic Regression, a weak learner, boosting algorithm is introduced in this project for converting weak learner into strong one, so that improving performance. The mothed we use so far is LightGBM (Light Gradient Boosting Machine), which make improvement from popular algorithm GBDT (Gradient Boosting Decision Tree) based on gradient boosting algorithm, whose mathematic form shown as following:

Where stands for function of new tree in round t. represents complexity of new tree.

Compared with another GBDT-family algorithm, XGBoost, LightGBM is much better in terms of faster training speed by using GOSS (Gradient-based One-Side Sampling), lower memory usage because of using Histogram based algorithm, and better accuracy thanks to leaf-wise growth strategy. Additionally, LightGBM is able to do with factor features without converting them into one-hot encoding. We have millions of sample points and hundreds of features in this project with most of them are factor features, LightGBM will be balanced in great performance and high efficiency. Downside of this algorithm is overfitting due to high complexity of tree. Appropriate maximum depth of tree is essential to solve this drawback.

Parameters we have to decide are mainly depth of tree, size of child sample, learning rate etc. They will be decided in terms of performance on train data by cross validation and grid search approach and referring others’ job on Kaggle kernel.

Last thing to mention is that our cross validation is scored with respect to RMSE (Root Mean Squared Error). Which is defined as:

Where is the predicted loyalty score for each *card\_id,* and is the actual loyalty score assigned to a *card\_id.*

# 5 Results

At current stage, we build a baseline LightGBM model with initial selection of parameter, mainly in order to evaluate feature importance. With this baseline model, we got mean RMSE around 3.65 on validation data, implementing by 5-fold cross validation. The result has been close to golden competent on Kaggle. What is most important is we have the information of importance of feature which is shown on appendix. As it indicates, all of features with high importance score are created during data preprocessing stage, such as mean of history month difference, mean of history authorized flag, maximum of new purchase amount, meaning our job is great so far. However, looking at RMSE after each round of boosting, this model makes slightly enhance from around 3.70 to 3.65 on validation data, while RMSE change from around 3.67 to 3.35. We shall try to make changes on parameter selection and feature selection to do with this problem.

# 6 Current Conclusion, Unexpected Challenges, Next Steps

For now, we only try out the LGBM (Light Gradient Boosting Machine) model, though finding out the importance of different features and getting a not bad model performance based on RMSE, the parameters we used is the default values from other people’s kernel on Kaggle, and we still need to apply some other methods like Bayesian optimization to adjust them. Also, except the LGBM, we should try other single model like Neural Network, Xgboost, Catboost, and KNN, Random Forest which we have learned on the class. Furthermore, we could consider assembling them together using Bayesian regression to get a higher CV (cross-validation) score.

On the other hand, for feature engineering, we can do more feature aggregation and try to figure out more new features that have a strong correlation with final target. Besides, we need to do more data visualization to take a closer look at our features and their relationship with the target value. On the contrast, instead of increasing number of features, we can drop some useless feature such as features with a high percentage of missing values, collinear features, features with less than 0 importance according to our training model, etc.

Next, we just try train, test, historical\_transactions, new\_merchant\_transactions datasets yet, leaving the merchants dataset behind. We may consider how to do feature engineering on this data set in the future and how to merge them with our previous cleaned dataset. Since features in merchants dataset is grouped by merchant\_id rather than card\_id, there are still some works to be done.

# 7 task distribution

This project can be divided into three different parts: data preprocessing, feature engineering and selection and model construction and comparisons.

Qirui He is responsible for data pre-processing, deleting outliers and handling missing data.

Fang Feng takes the charge of feature extraction and feature aggregation. LightGBM model is built by her to figure out feature importance and later model comparison.

Xichen Tan takes the responsibility for building a random forest model using the same data set and will compare the performance of LightGBM model and random forest.

1. An online community of data scientists and machine learners, allowing users to explore and build model to solve data science challenges. [↑](#footnote-ref-1)
2. One of the largest payment brands in Brazil. [↑](#footnote-ref-2)