Data Mining Project: Loyal Customer Prediction

1. Data exploration and preprocessing

Have an overview of the dataset will help us to model the task easier. So we do data exploration first. The dataset contains anonymized customers' shopping logs in the past 6 months before and on the "Double 11" day, and the information indicating whether they are loyal customers for given merchants (for training data only). The given dataset contains five files, with some basic statistics below:

Table 1

	train_label.csv	test_label.csv	User_log.csv	User_info.csv
# of brand	-	-	8161	-
# of categories	-	-	1581	-
# of users	106031	106031	212062	212062
# of merchants	1992	1991	4995	-
# of items	-	-	916773	-

We could observe that

(1) There much more merchants than users;

(2) The size of training set and testing set is similar;

(3) Not all merchants in user_log.csv are included in either train_label.csv or test_label.csv.

After further checking, we find that users in train_label.csv and test_label.csv have no overlapping part while merchants in train_label.csv and test_label.csv have 1990 in common.\

In real world, we may need to further preprocess the data, like filtering anomaly users and merchants. A user with huge number of actions in a single day could be a crawler, which definitely should be removed from training set. Such samples will have a bad effect on classifiers. For example, we assume one feature could be based on action counts, and the intuition of this feature is that if a user have more actions, the user will be more likely to be a loyal customer. However, if there is a crawler which has huge number of actions every day but will never buy a product from any merchant, the assumption obviously does not hold. Therefore, if we use count based features, the crawlers actually should be classified as positive which is not true.

However, due to the limited time and first submission result, I do not have time to implement this anomaly removing part.

2. Feature Engineering

For this project, I begin with some simple features which could be used to build more complex features in the future. For each entities, I build a profile for each of them. An entity here could be a user, a merchant, a user-merchant pair, a brand and so on. But currently, I only build profiles on users, merchants and user-merchant pairs. In the following part, I refer entity to either a user, a merchant or a user-merchant pair.

2.1 Monthly action counts

For all the entities, I extract the monthly action counts features for them. For example, if a user click on only one item each month, then the feature vector should be [1,1,1,1,1,1,7] which is a 1x7 matrix. The last element is the sum of the previous actions. For a merchant or user-merchant pair, we could also extract their monthly action counts features. For different action type, I build different monthly action counts feature respectively.

The intuition or assumption behind this feature is:

(1) if a user is more active in each month, the user will be more likely to be a loyal customer;

(2) if a merchant has more actions from users, the merchant will be more likely to have loyal customers;

(3) If a user-merchant pair is more active in each month, the user will be more likely to be loyal to corresponding merchant.

2.2 Monthly action ratio

For all the entities, I extract the monthly action ratio features for them. Given an action type of an entity, the action ratio at certain month is the ratio of the number of that type action to the overall actions in that month. We also include the ratio of certain type of action to the overall actions during the whole period.

The intuition or assumption behind this feature is similar to monthly action counts. However, this feature cares about the ratio rather than specific numbers.

2.3 Monthly Day Counts

For all the entities, I extract the monthly day counts features for them. Given an entity, a particular action type, the day counts of a month is defined as the number of days have that action type. We also include the number of days that have the particular action type during the whole period.

The intuition or assumption behind this feature is similar to monthly action counts. However, this features cares about the visiting frequency.

2.4 Diversity

For all the entities, I extract the diversity features for them. There are three types of diversity features:

(1) Item diversity: the number of unique items associated with an entity in each month or during the whole period;

(2) Brand diversity: the number of unique brands associated with an entity in each month or during the whole period;

(3) Category diversity: the number of unique categories associated with an entity in each month or during the whole period;

The intuition or assumption behind this feature is:

(1) if a user is interested in more different items, brands or categories, the user will be more likely to be a loyal customer;

(2) if a merchant has more different items, brands or categories which are interested by other users, the merchant will be more likely to have loyal customers;

(3) If a user-merchant pair has records about more different items, brands or categories, the user will be more likely to be loyal to corresponding merchant.

3. Models

In this project, I only tried one model: extreme gradient boosting (XGBoost). Here is the link for official documentation: https://xgboost.readthedocs.io/en/latest/.

There are many options for hyper-parameters. Due to the limited time, I only did the grid search on

(1) learning_rate: 0.01, 0.1, 1

(2) scale_pos_weight: 0.6, 1, 2

(3) max_depth : 6,7,8

(4) min_child_weight: 5, 100,200

Every time when I did grid search for hyper-parameters, 10-fold cross validation was used.

Finally I choose the following parameters for the best submission (private board):

(1) learning_rate: 0.1

(2) scale_pos_weight: 1

(3) max_depth: 7

(4) min_child_weight: 200

4. Results

Fortunately, I achieved 1st place on both private board and public board with log-loss 0.21773 and 0.22303 respectively.

Actually, the final results suffer from the imbalanced data, since there are 7885 positive samples and 122470 negative samples. The maximum probability of a user being a loyal customer of a merchant is only about 0.47. More sampling techniques could be used to improve the results.

5. Lesson learned

5.1 Feature management

Since the user log has a large size, it could be very time-consuming to extract the features. An efficient way to extract the features and manage features is necessary.

To effectively extract the features, we first need to design the features and find the dependencies among them before writing any code. In this project, I visualize such dependency relations by drawing feature dependency graph as shown in the following figure:

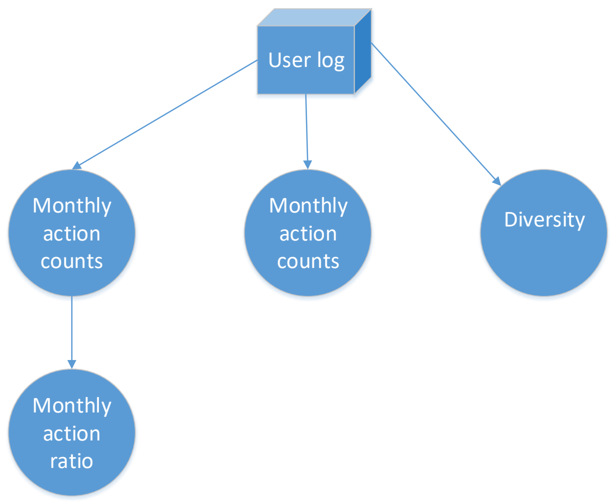

Figure 1 feature dependency graph

The feature dependency graph could help me to write the codes to extract those features by scanning the minimal times of the user log or parent features.
To incrementally store the feature, each time when I extract one feature, I write the feature into the corresponding entity profile in json format.

5.2 Hyper-parameters tuning

For a given training set and a given classifier, hyper-parameters tuning could have a huge impact on the final results. It is also a time-consuming tasks. Since I started the project very late, it was on the final day that I realized that I didn't have enough time to finish a larger grid searching since the estimated time could be 37 hours. Therefore, estimating time of finishing hyper-parameters choosing at the beginning is very important, or you may not be able to finish that before the deadline.