Customer Repurchase Prediction based on E-commerce data

Eliza Bai

September 8, 2017

Summary

The target of this project is to predict which buyer are most likely to use the voucher to make a purchase and which buyer will continue to purchase from the platform. I will describe my prediction solution, which consists of mainly on feature engineering and model training.

1 Problem description

E-commerce platforms normally using promotions to motivate those 'inactivate' buyers back to the website. However, most customers are one-time deal hunters, and promotions to them usually do not generate return on investment (ROI) as expected. Therefore, merchants need to identify potential 'loyal' ones from these customers, so as to conduct targeted promotions towards them based the promotion budget. E-commerce websites have the purchase history of all the customers at all the shops on their platforms. We can learn the preferences and habits of the 'inactivate' customers from their historical data, and then predict how likely a new customer will buy again from the platform.

This prediction problem can be formulated as a typical classification problem. Model training will be implemented to the classification tasks. Feature engineering, an integral part of data science, is often more important to a machine learning project. I will describe how to generate various types of features from user activity log data ($Transansactions.csv$) and study the importance of these features via performance study.

Four files which are $Train.csv$, $Test.csv$, $Transansactions.csv$ and $User-pofiles.csv$ are provided in this project. $Train.csv$ ('userid', 'promotionid received', 'voucher code received', 'voucher received time', $used?$, $repurchase?$) contains seven fields. 'userid' is unique for each buyer which ban be used to create userid based file later. 'promotionid received' and 'voucher code received' are same information which can not influence the buyer's decision, so I excluded these two in the feature variables. 'voucher received time' will be

contributed to creating features. *used*? and *repurchase*? have two class labels for model training. $Test.csv$ are same as $Train.csv$ except that *used*? and *repurchase*? are waiting to be predicted. $Transansactions.csv$ ('orderid', 'userid', 'shopid', 'total price', 'order time', 'voucher code used', 'promotionid used') gives information of users' purchasing history. To connect to the train data, for each 'userid', I grouped other fileds into lists or vectors ($df_history = df_transaction.groupby('userid', as_index = False).agg(lambda x : x.tolist())$). 'voucher code used' and 'promotionid used' are repeated again, so only one of them will be employed in the following sections. $Userpofiles.csv$ ('userid' 'phone verified', 'email verified', 'registration time', 'birthday' 'is seller', 'gender') illustrates personal information of each user. 'phone verifiled' only contain 1 which will be excluded in the features. 'registration time' will be transformed to the number of years. 'birthday' will be converted to age. Missing values and nonsense values of the age is simply filled with 0 and the age values are divided into four ranges (0, 20, 40, 60, 80). Missing values of registration years will be replaced by 0. Predicted gender will replace the real gender.

2 Feature engineering

In the rest of this section, I will give an overview of all the generated features. The features are joined with expanding training/testing data to transactions and userfiles. For example in python it could be

$df_T = df_train.merge(df_userfile, left_on =' userid', right_on =' userid', how =' left')$

$df_T = df_T.merge(df, left_on =' userid', right_on =' userid', how =' left')$

2.1 Count/ratio features

- Shop diversity: unique number of shopid over the whole data period

  The intuition behind shop diversity features is that if a user is interested in more shops of a platform, then the user is more likely to buy again from it.

- Day counts: the number of days made from first order to last order over the whole data period

  Day count features are mainly used to differentiate regular buyers from occasional buyers.

- Over all order number

  This feature is going to measure the order times of a user

- promotion used number

  This feature can tell the extent of a user's interest in promotions

  $(np.count_nonzero(np.isnan(row['promotionid_used'])))$

2.2 Aggregation features

- Max repeating order from same shop

  It usually indicates that the entity has a good reputation, so users are more likely to come back.

- Spent trend feature across time span

  It can tell the trend that a user would like to spend more or less on the platform. It's better measure by months or seasons. Due to time limit, I just measure by localizing the midpoint in the time period and split the total price of each user based on this midpoint. After that, the difference of average spent of two arrays are calculated. Related source code as below.

  $if(max(row['order_time'])-min(row['order_time']))/86400 > 30 and len(s[ind:])! = 0 : df.loc[index,'spent_trend'] = round(np.mean(s[ind:])-np.mean(s[:ind]))$

2.3 Recent activity features

- The time difference between last order until now

  This feature can measure the 'inactive' extent of each user.

- The time difference from last order and promotion sent time

  The user is more likely to be a one-time deal hunter only for promotion or not.

2.4 Age and gender related features

- Age

  It tells which age range is more likely to using online shopping

- registered year

  Different years of experience may have different behaviours

- Gender

  Different user groups may favour different types of products and shopping ways.

- email verified

  Users with email verified may more likely using the platform in long term.

- is seller

  Is seller or not also may have different influence on the decision.

3 Model training

I chose Python and scikit-learning machine learning library in this mini project to implement data training.

3.1 Models tested

- Decision tree

  It generates overfitting easily (reduce training set error at the cost of an increased test set error).

- SVM

  It is extremely slow to wait the result. Kernelized SVMs require the computation of a distance function between each point in the dataset, which is the dominating cost of $O(n_{features} \times n^2_{observations})$.

- Adaboost

  The results from Adaboost is slightly worse than Gradient boosting. Due to its similarity to Gradient boosting, I didn't explore too much on it.

3.2 Models adopted

- Logistics regression and the parameters I used

  $LogisticRegression(C = 1.0, class_weight = None, dual = False, fit_intercept = True, intercept_scaling = 1, max_iter = 100, multi_class =' ovr', n_jobs = 4, penalty =' l2', random_state = None, solver =' liblinear', tol = 0.0001, verbose = 0, warm_start = False)$

  I evaluate feature importance from the magnitude of coefficients of the model.

- Bagging random Forest

  $RandomForestClassifier(bootstrap = True, class_weight = None, criterion =' entropy', max_depth = 7, max_features = 4, max_leaf_nodes = None, min_impurity_decrease = 0.0, min_impurity_split = None, min_samples_leaf = 5, min_samples_split = 25, min_weight_fraction_leaf = 0.0, n_estimators = 20, n_jobs = 4, oob_score = False, random_state = None, verbose = 0, warm_start = False)$

  Feature importances can be output directly from this model.

- Gradient boosting

  $GradientBoostingClassifier(criterion =' friedman_mse', init = None, learning_rate = 1, loss =' deviance', max_depth = 1, max_features = None, max_leaf_nodes = None, min_impurity_decrease = 0.0, min_impurity_split = None, min_samples_leaf = 1, min_samples_split = 2, min_weight_fraction_leaf = 0.0, n_estimators = 100, presort =' auto', random_state = 0, subsample = 1.0, verbose = 0, warm_start = False)$

  Gradient Boosting model also have feature importances attribute.

- ensemble technique

  Since random forest and gradient boosting reduce variance and bias respectively, I ensemble both in VotingClassifier with soft Voting.

  A performance study of these models is demonstrated in the slides.

4 Imbalanced data

From train.csv I noticed that

Used? 0 *72891* 1 *2738*

Repurchase? 1 *62107* 0 *13522*

the class labels from both variables have imbalanced phenomenon which refers to a problem with classification problems where the classes are not represented equally. So even you predict Used labels to be all 0, the accuracy will still greater than 95%.

I resampled (undersampling) the data set to alleviate it.

5 Evaluation methods

I adopt accuracy ($metrics.accuracy_score$), AUC ($roc_auc_score()$) and 10 fold cross validation to assess the prediction quality.

6 Future work

Due to time issue, there are a lot of works could be explore further to improve the classification.

- Expand features

  For example, similarity scores, trends, PCA, LDA features, monthly features, trend features and different entities combination (e.g. shopid profile interact with userid profile).

- Tune model parameters

  Gridsearch and other methods can be employed.

- Explore other models

  XGBoost model and Feature ranking function of XGBoost adjust imbalancing data.

- Discretization of consecutive values

  Wrapper mehtod (fayyad1993)

- Other ways to improve imbalanced data