TMALL Repeat Buyers prediction

Armando Fortes[1], David Pissarra[1], and Gabriele Oliaro[1]

[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
{ferreiracardos10, pissarrad10, oliarog10}@mails.tsinghua.edu.cn

Abstract

In this report, we describe a solution to the TMall Repeat Buyers Prediction challenge from the Tianchi portal on aliyun.com. We solved the machine learning problem as part of our final project for a Big Data Intelligence class at Tsinghua University. Our code performed well compared to the other submissions, placing us, at the time of our last submission (Dec 14, 2021), in the Top 40 among 6257 teams in total, or the top 0.6%.

1 Introduction

Promotions are a common way for vendors large and small to attract new buyers. It is especially common to offer discounts on particular shopping holidays such as on Nov.11 (Double 11 day) in China or on the Friday after Thanksgiving day (Black Friday) in the US. The idea is that some of the new buyers, initially visiting a store because of the lower prices, will then return as regular customers, thus allowing the store owners to earn back the money lost due the discounts, and make even more profits. Not all new customers, however, will be back when the undiscounted prices are restored, and so to maximize the return on investment (ROI), patrons need to design their promotion campaigns to target as exclusively as possible the people who are most likely to become loyal customers.

With the right amount of data, vendors can use machine learning tools to help solve the problem of optimizing the ROI of a promotion campaign. While this may not be feasible for small physical stores, putting together a large dataset with records on customer characteristics or the transaction details over a period of time, is feasible for large e-commerce platforms (such as TMALL), who are already likely to keep track of transactions and user info on a regular basis.

The competition (1) provided us with a dataset (described in section 2) offered in two formats. The dataset was small enough (~360MB) to allow training on commodity hardware. The dataset included a set of customer-vendor pairs without a label (indicating whether the customer had become a repeated buyer at the vendor's store). In order to obtain a position on the leaderboard, we had to submit a CSV file with our predictions (obtained through our machine learning model) for the pairs without a label. The competition website then computed the accuracy of our classifications, computed using a ROC AUC score, and ranked our submission by comparing our score with those obtained by the other teams.

After cleaning the data and performing normal preprocessing operations, we obtained our predictions in two main steps. First, we constructed the features that we were hoping would correlate most strongly with the output labels (section 3). After having obtained the relevant features, we used a combination of gradient boosting (section 4.1) and a ensemble method (section 4.2) to perform classification.

2 The Dataset

The TMALL competition provided us with a dataset containing two main types of information:

1. Customer demographic information, such as age and gender
2. Customer-merchant interaction data:
   - Label indicating whether the customer is a repeated buyer (training dataset)
   - Activity log: one record (with timestamp, category, brand and item number, plus the action type) for each item that was clicked, added to cart, purchased or added to favorite

To protect the buyers' and the vendors' privacy, the data was anonymized, and further is was also sampled in a biased way.

The data is offered in two formats. The first format (file `data_format1.zip`), divides the data in 4 tables, and it structured in a way that makes feature engineering easier. The second format (file `data_format2.zip`) is more compact, as it consists of a single table, and minimizes the redundancy of information. Because our goal was to extract features, we picked format 1 (section 2.1).

2.1 The feature-engineering-ready dataset format

The dataset from `data_format1.zip` is organized in the following 3 tables: the User Profile Logs (Table 1), the User Behaviour Logs (Table 2), and the Training and Testing Data (Table 3)

Data Field	Description	Data Type
`user_id`	The unique ID identifying each buyer	Integer
`age_range`	The user's age range, encoded as follows: 1 for $<$ 18; 2 for $[18, 24]$; 3 for $[25, 29]$; 4 for $[30, 34]$; 5 for $[35, 39]$; 6 for $[40, 49]$; 7 and 8 for $\geq$ 50, 0 or NULL for unknown age,	Non-negative Integer or NULL
`gender`	The buyer's gender, encoded with 0 for female, 1 for male, 2 and NULL for unknown.	Integer or NULL

Table 1: The User Profile Table

Data Field	Description	Data Type
`user_id`	The unique ID identifying each buyer	Integer
`item_id`	The unique ID identifying each possible item that can be bought	Integer
`cat_id`	The unique ID identifying each possible category that an item can belong to	Integer
`merchant_id`	The unique ID identifying each vendor	Integer
`brand_id`	The unique ID identifying each brand an item can belong to	Integer
`time_stamp`	The date (`mm`: month and `dd`: day) when an action took place	String in `mmdd` format
`action_type`	The action taken by the buyer with respect to a vendor and an intem. Encoded as follows: 0 for a click, 1 for add-to-cart, 2 for purchase and 3 for add-to-favourite.	Integer

Table 2: The User Behavior Logs Table

Data Field	**Description**	**Data Type**
`user_id`	The unique ID identifying each buyer	Integer
`merchant_id`	The unique ID identifying each vendor	Integer
`label`	A binary value indicating whether `user_id` became a repeated buyer at `merchant_id`. Encoded as: 1 for repeat buyer, 0 for non-repeat buyer. The label is only available for the training portion of the data	Binary number (training), or empty (testing)

Table 3: The Training and Testing Data Table

3 Feature Engineering

The given datasets, user_log (user interation log) and user_info (information about users) do not provide any structured features that can be directly embedded in some model. It turns out that these datasets need to be analyzed in order to create valuable features that can correlate users and merchants. We created some kinds of features which are going to be explained in detail throughout this section.

3.1 Counting Features

There are four types of actions between users and merchants: purchases, add-to-favourites, add-to-carts, and clicks. Each of these actions can be counted, creating an interesting set of features. For example, a user with a large amount of purchases on the last few months is very likely to buy anything in the future. In addition, we also count the actions of one user w.r.t. a specific merchant, or even the actions of all users with the merchant. Moreover, we calculate unique values existing on the dataset: categories, brands, dates, and others. Therefore, we are able to understand how many different brands were sold or bought from a merchant or user respectively. Over the previous counters, we could calculate some ratios, for instance the ratio between clicks and total number of actions.

User ID	**Items**	**Categories**	**Brands**	**Days**	**Months**
263947	36	26	20	22	6
338674	68	14	34	23	5
...	...	...	...	...	...
61119	153	46	72	34	4

Table 4: Example of implemented counting features.

In Table 4, it is denoted some examples of unique counted values. In this example the user with ID 263947 has bought 36 items from 26 different categories, 20 brands, on 22 different days, over 6 different months.

3.2 Statistical Analysis Features

Over the counting features which were previously mentioned, we calculate simple statistical analysis as follows:

- Max – calculate the maximum value for a specific action regarding a user/merchant
- Mean – calculate the mean among action values for a given user/merchant
- Std – calculate the standard deviation over the action types for a user/merchant
- Median - calculate the median among action values for a given user/merchant

Despite the fact that these features seem quite simple, in fact 32 features were added to our dataframes, which really boosted our performance in the competition.

3.3 Time Span Features

The given dataset has information about interactions between users and merchants along 186 days (from 11 May to 12 November). Therefore, we could divide this big dataset into 6 smaller periods of time, each of them with exactly 31 days, as shown in Figure 1.

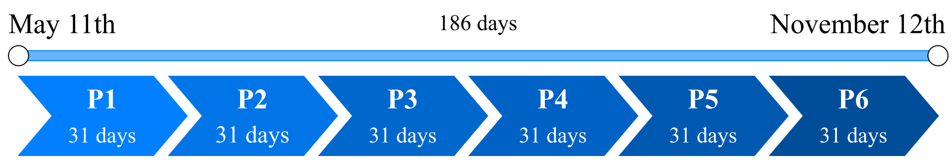

Figure 1: Dataset time span.

Each of these periods was respectively analyzed by fetching counting features for each period over all kinds of actions. Additionally, we calculated for each of 4 actions the slope of the linear regression function regarding the time periods (Equation 1). Therefore, we can calculate if some user is buying more products over the time (Figure 2), which may represent that this user is likely to buy more products in the meantime. On the other hand, if there is, for example, a merchant which is getting less clicks period by period, it might point out that this merchant will lose some costumers.

$$\frac{n\sum xy - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2} \tag{1}$$

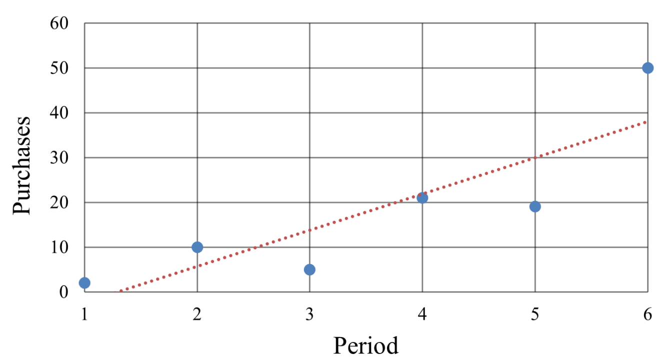

Figure 2: User uptrend example.

3.4 Double 11 Features

Double 11 is a very popular shopping festival in China, leading to huge volumes of transactions. As shown on Figure 3, there is a big amount of interactions when compared to the remaining 185 days. Hence, we counted values and calculated ratios over the number of actions for each user, merchant, and aggregation user-merchant. People that purchase products on days like this may only take advantage of big promotions, being one-time deal hunters.

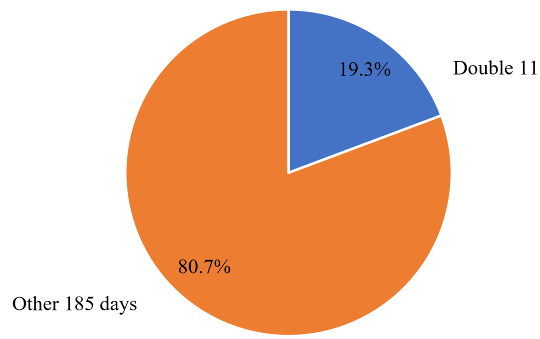

Figure 3: Double 11 impact on the dataset.

3.5 Principal Component Analysis Features

With the previous described features, we created 285 features in total. However, we summarize our dataframe by using Principal Component Analysis (PCA) in a smaller number of features (5 components, which can be seen as an hyperparameter). These 5 PCA features will be added to the entire dataset giving more useful data to the inference. By using PCA we will create 5 new features that increase the interpretability by maximizing the variance, so the model will better distinguish patterns of users or merchants. To conclude, we end up creating 290 features that will be received by our model.

4 Method

4.1 Gradient Boosting

Gradient Boosting is a very famous technique, which of course comes from the concept of Boosting, where we try to improve single weak models (in this case, decision trees) by combining them together, in order to generate a collectively strong model. Moreover, it actually extends the Boosting concept, since the addition of the generated weak models is formalized as a Gradient Descent algorithm, over an objective function. Also, each of the decision trees is added one at a time, and trained using the residual errors of their predecessors as labels, therefore, at each iteration, the focus is directed to the samples which have not yet been accurately predicted.
After thorough research for implementations of the Gradient Boosting technique, we found that XGBoost (2), LightGBM (3) and CatBoost (4) were some of the best ones for this type of problem. They are usually able to achieve great results, specially XGBoost which is the most used Gradient Boosting implementation for tabular data, regarding Kaggle competition winning solutions. Consequently, we decided integrate them and see how they would perform on the features we had developed.

4.2 Ensemble

At First, we relied on XGBoost (2) to make our predictions in the competition, since it was the model with the best results in our 10-Fold Cross Validation (section 5.2). However, we decided to build an Emsemble Model which would take into account the predictions from every individual implementation, because, even though XGBoost had the best performance so far, perhaps the other models were also extracting useful signals from the data that would complement the ones of XGBoost. Accordingly, we used Equation 2, where we assign a weight to each model and calculate the corresponding weighted average for our predictions.

$$P(u,m) = \sum_{K} w_k \cdot P_k(u,m), \quad \text{where} \sum_{K} w_k = 1 \tag{2}$$

We started by assigning weights according to the performance of each model during cross validation, so XGBoost would have the largest weight, and so on. After manually tuning the weights, we were able to find an optimal combination and achieve improved results, which are presented in the following section.

5 Evaluation and Experiments

5.1 The ROC AUC score

The ROC AUC score is an evaluation metric used to benchmark all submissions to the TMALL Repeat Buyers Prediction. ROC AUC stands for Area Under the Curve (AUC) of the Receiver operating characteristic (ROC) graph.
The ROC graph is a curve with the specificity (or probability of obtaining false positives in the classification) as the independent variable and the sensitivity (or probability of obtaining true positives in the classification) as the dependent variable. To obtain the curve, we compute the FPR (false positive rate) and the TPR (true positive rate) at several threshold settings, then plot the data points obtained.
Given the ROC plot, the AUC is simply the area between the ROC curve and the x-axis (i.e. the specificity axis). It can be computed simply by taking the integral of the ROC curve. Higher values of the AUC indicate better classifiers, because larger areas under the ROC curve are obtained if the curve is made up of points whose y-axis value are larger (i.e. the TPRs are larger).

5.2 K-Fold Cross Validation

After selecting the three Gradient Boosting implementations, we decided to run K-Fold Cross Validation on each of the models. This method consists in dividing our dataset into K splits and using one of these splits as a validation set, while the others serve as training. This is performed for every possible combination of splits. Accordingly, we are able to estimate the skill of the machine learning models, since we are exposing them to various different combinations of splits for training and validation. We used 10 folds and achieved the results presented in Table 5.

Model	ROC-AUC Score
XGBoost	0.6916
LightGBM	0.6773
CatBoost	0.6871
Ensemble Model	**0.6924**

Table 5: 10-fold cross validation average results of the different implementations.

In terms of individual implementations, XGBoost (2) had the best results, followed by CatBoost (4), and finally LightGBM (3). However, as previously mentioned, the Ensemble Model managed to outperform all the others, by combining their predictions.

5.3 Best Feature Fetching

One further experiment we added to our study was to use the XGBoost (2) algorithm to calculate how important each feature we implemented actually is. For this purpose, XGBoost analyzed the improvement in accuracy brought by each feature to the branches it is on. Afterwards, we were able to sort the features by importance and select only the most important ones for training. Some of the most important features developed, according to XGBoost, are presented in Table 6.
Upon filtering the most important features, we went from 290 to 150, which corresponds to almost a 50% cut. With this procedure we intended to analyze which of the developed features had the greatest influence in our results, and potentially decrease the complexity of the learning process, without affecting performance.

Feature	Importance
items_user_merchant	5.3590
purchases_user_merchant_period_5	5.2504
purchases_user_merchant	5.1866
categories_user_merchant	4.0474
periods_user_merchant	4.0189
double11_periods_user_merchant_ratio	2.9512
...	...

Table 6: Features sorted by importance.

Model	**Every Feature** (290)	**Best Features** (150)
XGBoost	0.6913	0.6916
LightGBM	0.6773	0.6757
CatBoost	0.6871	0.6882
Ensemble Model	0.6924	**0.6925**

Table 7: ROC-AUC scores of the different implementations before and after best feature fetching.

As one may observe in Table 7, fetching the best features was not a huge improvement on performance, however, it successfully reduces the complexity of learning process as well as the time taken by each model to be trained, while still maintaining top results. In the end, our best model is again the Ensemble model, trained using only the 150 best features.

6 Conclusion

In this project, we successfully tackled the TMALL Repeat Buyers prediction challenge by Aliyun.com, placing ourself among the Top 40 teams on the Leaderboard, at the time of submission. As of today (Jan 4, 2021), after more than 250 additional teams submitted their solutions, our Dec. 14 submission still ranks #45 out of a total of 6524 teams, or in the top 0.68% of all submissions. Our team name was `davidpissarra` and affiliation: 清华大学.

To achieve this result, first, we spent a significant amount of time performing feature engineering, in order to organize the information in the data in a way that allowed our classifier models to be most effective. We then designed a classifier architecture based on boosting, and computed our final results using an ensemble model based on XGBoost (2), LightGBM (3) and CatBoost (4).

References

[1] TMALL.com, "Repeat buyers prediction." [Online]. Available: https://tianchi.aliyun.com/competition/entrance/231576

[2] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[3] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[4] A. V. Dorogush, A. Gulin, G. Gusev, N. Kazeev, L. O. Prokhorenkova, and A. Vorobev, "Fighting biases with dynamic boosting," *CoRR*, vol. abs/1706.09516, 2017. [Online]. Available: http://arxiv.org/abs/1706.09516