Repeat Buyers Prediction after Sales Promotion for Tmall Platform

Bowei He[1], **Zhiqiang Zhang**[1], **Jian Liu**[1], **Fuzhen Zhuang**[2†], **Chuan Shi**[1†]
[1]Beijing University of Posts and Telecommunications, Beijing, China.
[2]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.
[†]Corresponding Author, zhuangfz@ics.ict.ac.cn, shichuan@bupt.edu.cn.

Abstract

The task of IJCAI 2015 competition is to predict those new buyers who will purchase items from the same merchants again within six months. This paper describes the solution of the winning team "DMGroup", from Beijing University of Posts and Telecommunications and Institute of Computing Technology, CAS. We analyze the task and transform it into a classification problem. Our solution mainly includes four steps. First, we analyze the characteristics of data, and correspondingly design some basic analysis strategies. Second, we extract and select features from three aspects: user-related features, merchant-related features and user-merchant interactive features. Third, we train four diverse classifiers through self-split training set. Finally, we perform the hybrid ensemble on both models and features to further improve the performance of our solution. Our solution obtains 0.699647 AUC score in Stage 1 and 0.711373 AUC score with ranking top one in Stage 2.

1 Task Description

IJCAI 2015 competition sponsored by Alibaba is a repeat-buys prediction task, and the data set is provided by the largest B2C platform *Tmall.com*[1] in China. In order to attract a large number of new buyers, merchants on Tmall.com sometimes hold big promotions (e.g., discounts and cash coupons.) on particular dates (e.g., Boxing-day Sales, "Black Friday" and "Double 11 (Nov. 11th)".). However, most of them are one-time deal hunters, and these promotions may have little long-term impact on sales. If we can identify the potential loyal customers, who can be converted into repeated buyers, the merchants can greatly reduce the promotion cost and enhance the return on investment (ROI).

In this challenge, Alibaba provides a set of merchants and their corresponding new buyers acquired during the promotion on "Double 11" day. The task is to predict which new buyers for given merchants will become loyal customers in the future. Specifically, participants need to predict the probability that these new buyers would purchase items from the same merchants again within six months. The AUC (area under curve) score is used as the evaluation measure.

The competition consists of two stages:

- In Stage 1, the training set contains 212,062 users and 4,995 merchants, while the test set contains another 212,108 users and shares the same set of merchants. Note that, there are no overlapping users between training set and test set. Participants may extract any features, employ any models, and only submit the prediction results for evaluation.
- In Stage 2, the top 50 teams from Stage 1 will have the opportunity to run their models on a much larger data set on Alibaba's cloud platform. Moreover, participates only need to submit the code in JAVA, then the distributed computation will be handled by the cloud platform.

Let's review more details of the training and test data in Stage 1. **train_format2.csv** contains anonymous users' shopping logs in the past six months before and on the "Double 11" day, and the label information '1' or '0' indicating whether they are repeated buyers or not for some certain merchants. Due to privacy issue, the organizers sampled the data in a biased way. Thus, the statistical results on this data set may have a little deviation from the actual case of Tmall.com, but they will not affect the applicability of the solution. There are six data fields in the training set, and the details of data format are depicted in Table 1. **test_format2.csv** has the same data format, while there are not labels.

Stage 2 includes a much larger data set, and all the players can only build their models based on the given code framework and submit their codes to the organizer's distributed cloud platform for evaluation. So it is very important to pay more attention to the algorithm efficiency.

2 Framework Overview

For the repeat buyers prediction problem, we can consider it as a binary classification problem. Specifically, for a given user-merchant pair, we predict whether the user will buy items again from the merchant. As shown in Fig. 1, our solution framework has four steps, including data processing, feature extraction and selection, model training, and ensemble.

• **Data Processing**. We split the training set into two parts (i.e., 50% and 50% in our solution.), one is local training set and the other is for validation. Furthermore, we analyze the

[1]www.tmall.com

Table 1: Details of Data Format

Data Fields	Definition
user_id	A unique id for the shopper.
age_range	User's age range: 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 ≥50.
gender	User' s gender: 0 for female, 1 for male, 2 and NULL for unknown.
merchant	A unique id for the merchant.
label	Value from {0, 1, -1, NULL}. '1' denotes 'user_id' is a repeat buyer for 'merchant_id', while '0' is the opposite. '-1' represents that 'user_id' is not a new customer of the given merchant, thus out of our prediction. However, such records may provide additional information. 'NULL' occurs only in the testing data, indicating it is a pair to predict.
activity_log	Set of interaction records between {user_id, merchant_id}, where each record is an action represented as 'item_id:category_id:brand_id:time_stamp:action_type'. '#' is used to separate two neighbouring elements. Records are not sorted in any particular order.

data and find some useful properties which can help to design more meaningful features.

• **Feature Extraction and Selection**. First we extract the features from three aspects, including user-related features (e.g., the total number of clicks and purchases for each user.), merchant-related features (e.g., the total number of clicks and purchases for each merchant.), and user-merchant interactive features (e.g., the number of clicks and purchases for each user to a merchant.). To improve the efficiency, we carefully evaluate and select final 83 useful features.

• **Model Training**. To guarantee the effectiveness and diversity of the selected models, we try our best to evaluate many classification models according to the local training data and validation data. Finally, four kinds of classification models are used in our solution, Gradient Boosting Decision Tree, Random Forests, AdaBoost and Logistic Regression.

• **Ensemble**. To further improve the performance, we conduct model ensemble as well as feature ensemble based on the local training data and validation data. The final results on test data show that the ensemble framework can effectively improved the AUC score compared to each single model.

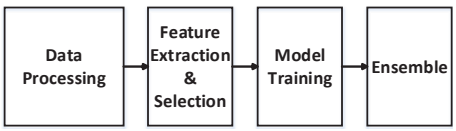

Figure 1: The Architecture of Developed Framework.

3 Data Processing

To evaluate the proposed solution and avoid over-fitting, a validation is needed. Generally, 70% or 60% of the data are used for training and the rest are for validation. Since we find that the given training and test data are with ratio 1:1, so we also equally split the training data into local training set and validation set. In this way, we find that the results of AUC scores on the validation set off-line have the same variation trend with the ones on test data online.

Through data analysis, we also find that the numbers of click and purchasing behaviours of users grow explosively on 'Double 11' day, which are visualized in Fig. 2. Therefore, we consider the feature extraction from two sides, before 'Double 11' day and on 'Double 11' day.

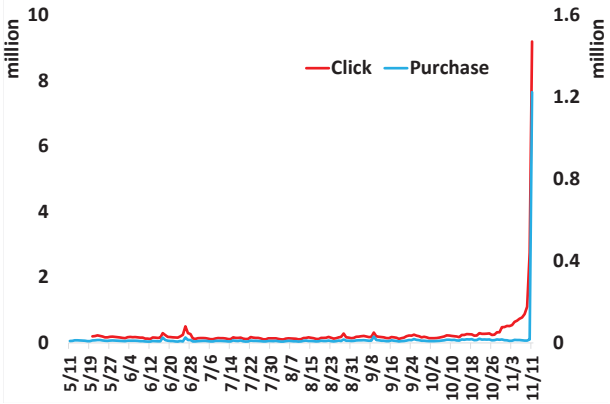

Figure 2: Statistics of click and purchase behaviour

Moreover, we find that almost 10% of purchasing behaviours happened without any click behaviours before. In other word, a user purchases an item in a merchant without clicking the item. It is strange and we try to fill these missing click records. Unfortunately, it does not work and we find that all behaviour records except the purchase in our hands are sampled in a certain proportion from the original Tmall.com data.

4 Feature Extraction and Selection

Feature extraction and selection is a very important process in data mining. We extract the features from three aspects: user-related features, merchant-related features, and interactive features between users and merchants. We summarize some important features in Table 2, and list all extracted features in the Appendix.

4.1 User-Related Feature

This kind of features describe the users' properties. Furthermore, we group the user-related features into three subcategories: user profile feature, user count feature and user ratio feature.

User Profile Feature: From the provided user profile table, we can find user's personal information, e.g., age and gender. These information are very useful, since users with different ages and genders have different probability to be repeat buyers. So we generate the age_range and gender features.

Table 2: Extracted Features

Feature Category	Feature Sub-category	Feature examples
User-Related Feature	User Profile Feature	age_range / gender
	User Count Feature	user_click_cnt / user_cart_cntuser_buy_cnt / user_fav_cnt / user_active_clickday_cnt · · ·
	User Ratio Feature	user_click_11_item_ratio / user_cart_11_item_ratio / user_buy_11_item_raito · · ·
Merchant-Related Feature	Merchant Count Feature	shop_click_cnt / shop_cart_cnt / shop_buy_cnt/shop_fav_cnt / shop_click_cnt_before_10 · · ·
	Merchant Ratio Feature	shop_11_old_user_ratio / shop_11_new_buy_ratio / shop_dup_user_before_11_ratio · · ·
	Merchant Data Leak Feature	shop_appear_in_train / shop_label_1_in_train / shop_label_1_in_train_ratio
Interactive Feature	Interactive Count Feature	user_shop_click_cnt / user_shop_cart_cnt / user_shop_buy_cnt / user_shop_fav_cnt · · ·
	Interactive Cross Feature	user_shop_brand_cross_score / user_shop_cat_cross_score / user_shop_max_dup_item_before_11_ratio · · ·
	Interactive Data Leak Feature	user_shop_cat_label_1_in_train / user_shop_cat_appear_in_train

User Count Feature: This kind of features consider the numbers of four behaviours (i.e., click, add_to_cart, purchase and add_to_favorite) during days before "Double 11" and on "Double 11". Therefore, we generate eight features by counting users' behaviours, e.g., user_click_cnt, before and on "Double 11".

Moreover, users may have different behaviours on different items, categories and brands. When a user has a lot of interaction with different items, categories and brands, he may be an important user. So we count the number of items, categories and brands a user has ever interactions on.

Furthermore, a user who has ever been a repeat buyers may have higher probability to be a repeat buyer, so we calculate the total times that a user becomes a repeat buyer. Simultaneously, we count the total days on which a user has purchase records.

User Ratio Feature: Through analyzing data, we find that some users' behaviours in November is very different from the ones before November. Some users have lots of behaviour only in the November. This is reasonable, since Tmall.com has lots of sale promotions in November until Double 11. So we consider the differences of users' behaviours on different items, categories and brands in November and these behaviours at other time.

4.2 Merchant-related Feature

There are 1994 merchants, which appear in both training and test data. In Stage 2, we have to use the Alibaba's cloud platform. In that platform, the feature extraction is different from that of Stage 1. In Stage 2, there are one master node and many slave nodes, as show in Fig. 3. Data are divided and distributed to different slave nodes. The platform only ensure that all of one user's log are distributed to one slave node, but one merchant's log may be distributed to many slave nodes. So we cannot accurately obtain merchant-related features in Stage 2. Fortunately, the merchants in both stages are the same, so we directly adopt the extracted merchant-related feature in Stage 1 and use it as an auxiliary file for Stage 2. We also extract three types of merchant-related features: merchant count feature and merchant ratio feature and merchant data leak feature.

Merchant Count Feature: We count the times of behaviours on merchants as features, such as the total number of four behaviours (click, add_to_cart, purchase and add_to_favorite) and the total number of different items, categories and brands in a given merchant. Similar to the user count feature, This kind of features can reflect the property of these merchants.

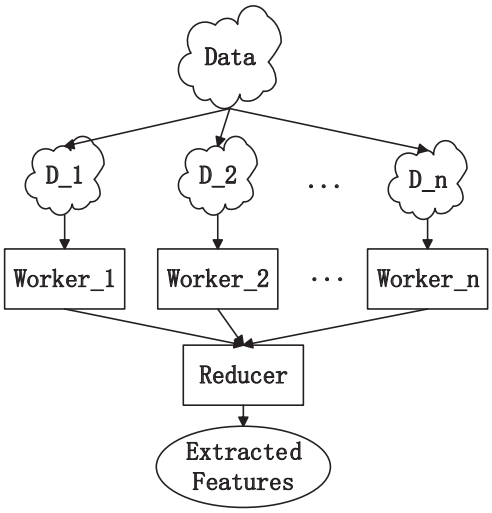

Figure 3: Feature extraction in distributed cloud platform

Merchant Ratio Feature: There are some merchant-related features, totally different from user-related features, while they contribute a lot to the final results. One feature is the repeat buyer ratio of a merchant. The task is to predict the probability that these new buyers would purchase from the same merchants again within six months. If a merchant has a lot of repeat buyers, it implies that the merchant has very good reputation or items. It will attract users to buy more than once. In addition, the ratio features are different from count features. The count feature can reflect whether a merchant is popular, while the ratio feature can reflect a merchant's quality.

Merchant Data Leak Feature: Except the features mentioned above, there is a feature playing a very important role in our solution and we call it the data leak feature. In the competition, all merchants appear in both training and test set. From the training set, we can calculate the ratio of a merchant that be bought again. Higher ratio implies the high probability that a user would purchase items from the same merchants again.

4.3 User-Merchant Interactive Feature

Except the two kinds of features, we also extract many effective user-merchant interactive features, and they can be roughly classified into three categories: interactive count feature, interactive cross feature and interactive data leak feature.

Interactive Count Feature: One type of these features is the

total number of four behaviours (click, add_to_cart, purchase and add_to_favorite) which is similar to user-related features and merchant-related features. Another type of features is the total numbers of different items, categories and brands that a user has interactions on the merchant. For a user-merchant pair, the time span from the first visit to "Double 11" is also useful.

Interactive Crossover Feature: An item is associated with one category and one brand, and we find that different items, categories and brands will influence whether a user becomes a repeat one. We calculate the repeat buy ratio of each category, brand and item, and choose the max ratio as the crossover feature.

In addition, we find the user will repeatedly buy items in a merchant if the merchant sell some categories or brands that the user ever buy or click. For each user, we can calculate the set of categories and brands that he has ever clicked or bought. Similarly, we can also calculate the set of categories and brands clicked or bought by a merchant. Then we can calculate the intersection set of categories and brands between a user and a merchant. We use the size of the intersection set as the interactive crossover feature.

Interactive Data Leak Feature: Furthermore, from the training set, we can find that category is also an important factor that affects the user to become a repeat buyer. We calculate the occurrence number of each category appearing in the training set, and its occurrence number among training samples with label 1. We use these two numbers as two interactive features too.

4.4 Feature Selection

There are 150 features are extracted, however, some of them are redundant. To improve the efficiency, we design a feature selection strategy to further select the features. Specifically, we design three different validation sets and make sure that each feature has performance improvement on all of three validation sets, and then we choose it as an effective feature. This selection process is very important, since in Stage 2 we have to run our program in the cloud platform, and the maximal running time is limited to 4 hours. Finally, 83 features out of 150 are obtained.

5 Model Training

Based on selected features, we test various kinds of classification models. Finally, we find the following four classifiers can achieve satisfying performance on these features: Gradient Boosting Decision Tree, Random Forests, AdaBoost (decision tree as week learner) and Logistic Regression. These four classifiers are adopted in Stage 1, and only GBDT is used in Stage 2 due to the limitation of the Alibaba's cloud platform, we only apply GBDT classifier in Stage 2. According to the characteristics of data, we carefully tune the these models and find some tricks to achieve good performance.

5.1 Gradient Boosting Decision Tree

Gradient Boosting Decision Tree (GBDT) [Friedman, 2001; 2002] uses decision trees as base learners and combines them into a single strong learner. The final prediction of GBDT is the weighted sum of outputs from each tree. In Stage 1, we adopt the implementation from scikit-learn package [Pedregosa *et al.*, 2011], and the loss function is "deviance", which is the same as logistic regression. In Stage 2, We employ GBDT classifier provided by Alibaba's cloud platform.

There are some specific parameters in GBDT: the number of trees (iterations), learning rate, the maximum depth of each tree and the minimum number of samples in a leaf. The last two parameters control the size of each tree. Empirical results shows that small values of learning rate favor better test error [Hastie *et al.*, 2009], so we set it as 0.03 in both Stage 1 and Stage 2. In Stage 1, we train 500 trees with no less than 50 samples in each leaf. Since the data set is much larger in Stage 2, so 2200 trees, of which the maximum depth is 5, are trained to achieve the best AUC score 0.711373.

5.2 Random Forest

Random Forest (RF) [Breiman, 2001; Ho, 1998] is another tree-based ensemble model, in which each tree built from a sampled subset drawn with replacement (i.e., a bootstrap sample) from the training set. We also adopt the implementation from scikit-learn package, which combines classifiers by averaging their probabilistic prediction. We train 4000 trees with no less than 100 samples in each leaf.

5.3 AdaBoost

AdaBoost [Freund and Schapire, 1995] is a boosting method which fits a sequence of weak learners on repeatedly modified versions of the data. At each step, the sample weights are modified according to the prediction error made by the previous step, and then the learning algorithm is applied to the re-weighted data. We also adopt the implementation from scikit-learn package and use decision tree as week learning. We trained 150 decision trees for Adaboost.

5.4 Logistic Regression

We use the implementation of regularized logistic regression from scikit-learn package, with liblinear library [Fan *et al.*, 2008] solver, to train classifier. L2 regularization is chosen to avoid overfitting and the regularization strength is set to 50. Due to serious class imbalance, the weight of positive samples is set to 100, while the weight of negative samples is 1. In addition, the values of all features are normalized to the range of [0,1] with the minimum-maximum scaler.

6 Ensemble

In order to further boost performance, we propose a hybrid ensemble method on both diverse classification models and feature combinations. Specifically, we not only ensemble four classifiers mentioned above but also train these classifiers on different feature combinations. In addition, we combine the results of these models with the stacking technique [Wolpert, 1992; Zhou, 2012]. Fig. 4 shows the hybrid ensemble framework. For each feature combination, we employ 4 classifiers, and then combine the results of all 28 classifiers with a linear regression.

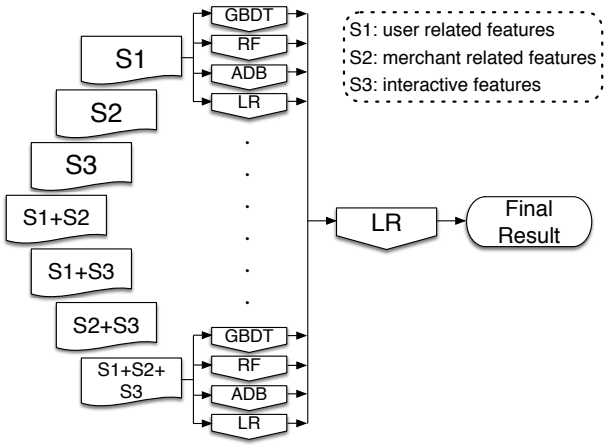

Figure 4: The hybrid ensemble framework. (GBDT: Gradient Boosting Decision Tree, RF: Random Forest, ADB: AdaBoost, LR: Logistic Regression)

6.1 Model Ensemble

In the model ensemble framework, we assign normalized weights to the four classifiers and then use the weighted average probability as final prediction. Since there are only four models, we do a grid search of weights to find the optimal weight setting.

To evaluate the performance under a specific weight setting, we check the results on different data set. Concretely, we randomly select 50% users from the local training set to form a new training set and use the rest users as the corresponding test set. We train different classifiers on this training set and predict on the corresponding test set. We combine these models with different weights, and then check whether the combination will improve performance. Similarity, we do model training on local training set and predict on validation set. Then we check whether the performance improvement still holds on this data set. After several times of experiments, we find the optimal weights are as follows: 1 for both GBDT and Random Forest, 16 for Adaboost and 0.05 for Logistic Regression.

6.2 Feature Ensemble

In order to make full use of different types of features to improve performance, we consider all combinations of the three disjoint subset of features: user-related features, merchant-related features and user-merchant interactive features (called S1, S2 and S3, respectively). There are seven combinations, i.e., S1, S2, S3, S1+S2, S1+S3, S2+S3 and S1+S2+S3. After that, four classifiers (i.e., GBDT, RF, ADB and LR) are trained on them independently. Then, 28 results (4 classifiers on 7 feature combinations) are generated. Finally, we apply the logistic regression classifier to make final prediction.

6.3 Experiment Comparison

Table 3 shows the AUC score of all methods, including four individual classifiers, model ensemble and model+feature ensemble method, on the local training and validation set. We can observe that the model ensemble method outperforms any individual classifier. Moreover, the hybrid ensemble on both models and features achieves the best performance among all methods.

Table 3: The AUC score of different methods

Methods	AUC in local evaluation
GBDT	0.688379
Random Forest	0.688377
AdaBoost	0.683360
Logistic Regression	0.681488
Model Ensemble	0.691793
Model+Feature Ensemble	**0.692564**

7 Acknowledgments

This work is supported by the National Basic Research Program of China (2013CB329603), the National Science Foundation of China (Nos. 61375058, 61473273, 61203297), National High-tech R&D Program of China (863 Program) (Nos. 2015AA050203, 2013AA01A606), and the CCF-Tencent Open Fund.

References

[Breiman, 2001] L. Breiman. Random forests. *Machine Learning*, pages 5–32, 2001.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, pages 1871–1874, 2008.

[Freund and Schapire, 1995] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, pages 119–139, 1995.

[Friedman, 2001] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, pages 1189–1232, 2001.

[Friedman, 2002] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, pages 367–378, 2002.

[Hastie *et al.*, 2009] T. Hastie, R. Tibshirani, and J. Friedman. *Elements of Statistical Learning Ed. 2*. Springer, 2009.

[Ho, 1998] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans Pattern Analysis and Machine Intelligence*, pages 832–844, 1998.

[Pedregosa *et al.*, 2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, pages 2825–2830, 2011.

[Wolpert, 1992] David H. Wolpert. Stacked generalization. *Neural Networks*, 1992.

[Zhou, 2012] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman and Hall/CRC, 2012.

Appendices

A User-Related Features

A.1 User Profile Feature

1. age_range: user's age range
2. gender: user's gender

A.2 User Count Feature

3. user_click_cnt: user's total click times except "Double 11"
4. user_cart_cnt: user's total add to cart times except "Double 11"
5. user_buy_cnt: user's total buy times except "Double 11"
6. user_fav_cnt: user's total favorite times except "Double 11"
7. user_active_clickday_cnt: the number of different days that a user have click behaviour
8. user_active_buyday_cnt: the number of different days that a user have buy behaviour
9. user_click_itemcnt: the number of different items that a user click
10. user_cart_itemcnt: the number of different items that a user add to cart
11. user_buy_itemcnt: the number of different items that a user buy
12. user_fav_itemcnt: the number of different items that a user add to favorite
13. user_click_catcnt: the number of different category that a user click
14. user_cart_catcnt: the number of different category that a user add to cart
15. user_buy_catcnt: the number of different category that a user buy
16. user_fav_catcnt: the number of different category that a user add to favorite
17. user_click_brdcnt: the number of different brand that a user click
18. user_cart_brdcnt: the number of different brand that a user add to cart
19. user_buy_brdcnt: the number of different brand that a user buy
20. user_fav_brdcnt: the number of different brand that a user add to favorite
21. user_dup_buycnt: the number of merchants that a user have repeat buy behaviour in them

A.3 User Ratio Feature

22. user_click_11_item_ratio: the ratio of different item that a user click in November
23. user_cart_11_item_ratio: the ratio of different item that a user add to cart in November
24. user_buy_11_item_ratio: the ratio of different item that a user buy in November
25. user_fav_11_item_ratio: the ratio of different item that a user add to favorite in November
26. user_click_11_cat_ratio: the ratio of different category that a user click in November
27. user_cart_11_cat_ratio: the ratio of different category that a user add to cart in November
28. user_buy_11_cat_ratio: the ratio of different category that a user buy in November
29. user_fav_11_cat_ratio: the ratio of different category that a user add to favorite in November
30. user_click_11_brd_ratio: the ratio of different brand that a user click in November
31. user_cart_11_brd_ratio: the ratio of different brand that a user add to cart in November
32. user_buy_11_brd_ratio: the ratio of different brand that a user buy in November
33. user_fav_11_brd_ratio: the ratio of different brand that a user add to favorite in November
34. user_11buy_ratio: the portion of the user buy in "Double 11" to all buy behaviour
35. user_buy2click: the conversion rate of a user from click to buy

B Merchant-Related Features

B.1 Merchant Count Feature

36. merchant_click_cnt: merchant's total click times except "Double 11"
37. merchant_cart_cnt: merchant's total add to cart times except "Double 11"
38. merchant_buy_cnt: merchant's total buy times except "Double 11"
39. merchant_fav_cnt: merchant's total favorite times except "Double 11"
40. merchant_click_cnt_before_10: merchant's total click times before October
41. merchant_cart_cnt_before_10: merchant's total add to cart times before October
42. merchant_buy_cnt_before_10: merchant's total buy times except October
43. merchant_fav_cnt_before_10: merchant's total favorite times except October
44. merchant_click_itemcnt: the number of different items to be clicked in a merchant
45. merchant_cart_itemcnt: the number of different items to be added to cart in a merchant
46. merchant_buy_itemcnt: the number of different items to be bought in a merchant
47. merchant_fav_itemcnt: the number of different items to be added to favorite int a merchant
48. merchant_click_catcnt: the number of different category to be clicked in a merchant
49. merchant_cart_catcnt: the number of different category to be added to cart in a merchant
50. merchant_buy_catcnt: the number of different category to be bought in a merchant

51. merchant_fav_catcnt: the number of different category to be added to favorite in a merchant
52. merchant_click_brdcnt: the number of different brand to be clicked in a merchant
53. merchant_cart_brdcnt: the number of different brand to be added to cart in a merchant
54. merchant_buy_brdcnt: the number of different brand to be bought in a merchant
55. merchant_fav_brdcnt: the number of different brand to be added to favorite a merchant
56. merchant_dup_buy: the number of repeat buy times in the merchant
57. merchant_dup_buy_user: the number of repeat buy user in the merchant

B.2 Merchant Ratio Feature

58. merchant_11_old_user_ratio: the ratio of regular user have buy behaviour in "Double 11" in a merchant
59. merchant_11_new_buy_ratio. the ratio of the number of buy behaviour from new user in "Double 11" in a merchant
60. merchant_dup_user_before_11_ratio: the ratio of repeat buy user before "Double 11" in a merchant
61. merchant_normal_user_before_11_ratio: the ratio of normal user before "Double 11" in a merchant (normal user means the user isn't a repeat buy user or a new user in "Double 11")
62. merchant_normal_user_ratio: the ratio of normal user in a merchant (normal user means the user isn't a repeat buy user or a new user in "Double 11")
63. merchant_11buy_portion: the portion of a merchant's buy behaviour in "Double 11" to all buy behaviour
64. merchant_buy2click: the conversion rate of a merchant from click to buy
65. merchant_buy2click_before_10: the conversion rate of a merchant from click to buy before October
66. merchant_cat_dup_user_ratio: the ratio of category repeat buy user in a merchant (category repeat buy user means a user buy the same category in the merchant more than once)

B.3 Merchant Data Leak Feature

67. merchant_appear_in_train: the occurrence number of a merchant in training set
68. merchant_label_1_in_train: the occurrence number of a merchant with label 1 in training set
69. merchant_label_1_in_train_ratio: the ratio of a merchant with label 1 in training set

C Interactive Features

C.1 Interactive Count Feature

70. user_merchant_click_cnt: a user's total click times on a merchant
71. user_merchant_cart_cnt: a user's total add to cart times on a merchant
72. user_merchant_buy_cnt: a user's total buy times on a merchant
73. user_merchant_fav_cnt: a user's total favorite times on a merchant
74. user_merchant_firstact_daysto1111: the life span of a user who has behaviour to a merchant
75. user_merchant_itemcnt: the number of different items that a user has behaviour to a merchant
76. user_merchant_catcnt: the number of different categories that a user has behaviour to a merchant

C.2 Interactive Cross Feature

77. user_merchant_brand_cross_score: the interactive number of brand that a user has ever bought and a merchant has ever sold
78. user_merchant_cat_cross_score: the interactive number of category that a user has ever bought and a merchant has ever sold
79. user_merchant_max_dup_item_before_11_ratio: see 5.4
80. user_merchant_max_dup_cat_before_11_ratio: see 5.4
81. user_merchant_max_dup_brand_before_11_ratio: see 5.4

C.3 Interactive Data Leak Feature

82. user_merchant_cat_label_1_in_train: see 5.4
83. user_merchant_cat_appear_in_train: see 5.4