2015 IEEE 15th International Conference on Data Mining Workshops

Multi-classes Feature Engineering with Sliding Window for Purchase Prediction in Mobile Commerce

Qiang Li*, Maojie Gu†, Keren Zhou‡ and Xiaoming Sun§
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
Email: liqiang01@ict.ac.cn*, gumaojie@ict.ac.cn†, zhoukeren@ncic.ac.cn‡, sunxiaoming@ict.ac.cn§

***Abstract*—Mobile devices become more and more prevalent in recent years, especially in young groups. The rapid progress of mobile devices promotes the development of M-Commerce business. The purchase on mobile terminals accounts for a considerable percentage in the total trading volume of E-Commerce and begins to draw the attention of E-Commerce corporation. Alibaba held a Mobile Recommendation Algorithm Competition aiming to recommend appropriate items for mobile users at the right time and place. The dataset provided by Alibaba consists of about 6 billion operation logs made by 5 million Taobao users towards over 150 million items spanning a period of one month. Compared with traditional scenarios in purchase predicting, the competition raised three challenges: (1)The dataset is too large to be processed in personal computers; (2)Some days with great discounts provided by Taobao Marketplace are within the period of dataset; (3)Positive samples are too few compared to the dimension of features.**

In this paper we study the problem of predicting the purchase behaviour of M-Commerce users, by exploring the solution for Alibaba's Mobile Recommendation Algorithm Competition. We first deeply study the habit of customers and filter many outliers. After that we adopt the method of "sliding window" to supply positive samples of training dataset and smooth the burst of sales near Dec 12th. We design a feature engineering framework to extract 6 categories of features that aim to capture the buying potential of user-item pairs. Our features exploit the interaction of user-item pair, user's shopping habit and item' attraction for users. Then we apply Gradient Boost Decision Trees (GBDT) as the training model. In the end, we combine outputs of individual GBDT together by Logistic Regression to get the final predictions. Our solution achieves 8.66% F1 score, and ranks the third place in the final round.

I. Introduction

With the proliferation of intelligent mobile devices, the mobile commerce is growing at a staggering rate. While traditional user recommendation systems are applicable for pc users, developing feasible model for mobile recommendation is still an uncharted territory. The Alibaba Mobile Recommendation Algorithm Competition focuses on solving the problem. The whole dataset is provided by Alibaba's M-Commerce platforms, which consists of two parts–5 million user behaviors and 156 million items in a month. Different from the traditional competition, the organizers presented us the geological hash string as a description of users' location. Our task is to use the dataset as training examples to predict the user behaviors in the next day.

Clearly, we could view this problem as a binary-classification task, of which we have to predict whether a user will buy a certain product or not. The competition uses F1 scores as the standard of final evaluation: $F1 = 2pr/(p+r)$, where p is the precision and r is the recall. For instance, suppose the prediction set consists of 5 items, out of which 3 items are in the real dataset. Then it renders $p = 3/5$. Likewise, suppose the real dataset has 8 records, then we have $r = 3/8$. Therefore the final F1 score should be 0.46.

There are three main challenges in the competition:

1) Because the given dataset is larger than the previous competitions, analysis softwares on personal computers are not capable to cope with the problem. On the contrary, we are supposed to use Alibaba's Yushanfang distributed platform.
2) We have to extract features, and build available user and item profiles from original noisy dataset with few attributes. For instance, records in December twelfth, which is referred to as double-12, is very different from other days, since Alibaba has a special offer at the time each year.
3) There is an extreme unbalance between positive and negative examples, which indicates that a myriad of people view items but do not buy them. Such a distribution often leads to a biased predicting model, which achieves low F1 score.

To solve these challenges, in other words, to get a high F1 score, we come up with bunch of innovative solutions. In the preprocessing stage, we filter out noisy data, which could be generated by web crawlers or unpredictable behaviours. In addition, we smooth dataset in double-12 by normalization. Since the positive instances are much more fewer than negative ones, we adopt sliding window method to make a balance. After tackling the raw data, we extract 6 categories of features. Then we use those features as inputs of individual GBDTs, and blend the outputs by Logistic Regression to get the final result.

The paper is organized as follows. Section 2 presents the overview of our problem solving framework. Section 3 presents approaches to filter the given dataset. Section 4

978-1-4673-8493-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDMW.2015.172

describes a framework for extracting available features. We introduce our models in section 4. Section 5 shows historical result to illustrate how we optimize our models in the competition.

II. System Overview

Figure 1 provides an overview of our system during the Ali Mobile Recommendation Competition. There are four main stages in our system:

- Preprocessing
- Generating features
- Training individual models
- Blending outputs of each individual model to get final result

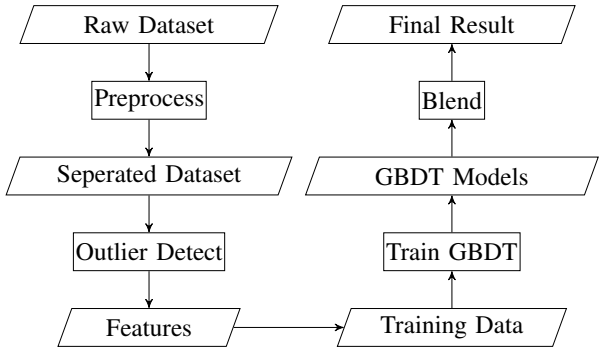

Fig. 1. Overview of the wholesale procedures with different dataset

In the first stage, we use some effective rules to filter raw data and detect outliers. Furthermore, we also discard some users whose behavior looks like the web crawler. In the second stage, we generate various features by different approaches according to our analysis of the given dataset. In the third state we focus on generating models including Random Forests and Gradient Boosting Decision Trees, to identify whether a user will buy a particular item. In the end, we ensemble all the outputs generated by each individual model to get final results.

III. preprocessing

This section introduces the procedure of data preprocessing and how we build the training and testing dataset with "sliding window" method. Besides, we also present the decaying function of four behaviours as well as the techniques of handling Double-12.

A. User Behaviour Analysis and filtering

The original data is a table of 5,822,532,780 records, consisting of the behaviour logs of 5 million user towards 156 million items from 11/18/2014 to 12/18/2014. Each record is a user's operation log upon an item, containing the behaviour types(click, collect, add to cart, buy), operating time and the user's current location. Each item belongs to a category. Besides, we are given a subset of items I^o, which belongs to service, catering or photograph. Our target is to predict the purchase behaviour of users towards the items subset on 12/19/2014.

After carefully observing the operation logs, we find some users' behaviour strange. We conjecture that they might be web crawlers, which click the main page of an item and quickly jump to other items' main page through out-links. In this way they click items for thousands of times to download web pages, but never buy any item. Apart from web crawlers, there are other types of outliers. These users buy some items and disguise as customers. But in fact they aim at promoting the page views of some specific items. Outliers seldom buy items, but they produce huge amount of page views, which cause severe bias to the features of users. So finding strange users and filtering out them from original records is a important step before feature engineering.

Outliers tend to click many items with few to buy. In this competition, we propose 4 rules to filter users.

1) never collect, add to cart or buy items;
2) click for more than 500 times but never buy items;
3) click on I^o for more than 200 times but never collect, add to cart or buy items of I^o;
4) click for more than 5000 times and $\frac{\#\text{click}}{\#\text{buy}} > 500$.

We remove users that satisfy the 4 rules listed above. 376900 users are removed from original 5 million users but only 2.4% positive samples are removed. Experimental results show that the user filtering effectively improves the performance.

B. Supplement Positive Samples with Sliding Window

The target of the competition is to predict the purchase in the next day with historical operation logs. How to construct samples and training dataset? A straightforward idea to construct training dataset is extracting user-item pairs that are active before 12/18/2014. A user-item pair is a sample in training dataset. Here "active" means the user and the item in a user-item pair are once interacted with. After defining the user-item pairs, we can extract features with the logs from 11/18/2014 to 12/17/2014. We label each bought user-item pair on 12/18/2014 as positive sample. Otherwise, we label the corresponding sample as negative. Till now, we obtain a training dataset transformed from operation logs and the predicting problem becomes a binary classification problem. Similarly we can obtain online test samples by filling the features of active user-item pairs between 11/18/2014 and 12/18/2014. With machine learning methods, we can train a model with training dataset and predict which items users will buy in 12/19/2014. However, there are three problems in this method. (1) We could only predict items once interacted with users before the buying day. (2) The features of training dataset and testing dataset are not independent identically distributed. (3) There are only few positive samples with respect to feature dimensions, and they are also far less than the negative samples.

The first problems is similar as the cold-start problem, which is a classical problem in recommend system, therefore we did not spend much effort on it. To solve the second

and third problem, we propose the "sliding window" method. The features of training dataset and testing dataset constructed with the method mentioned above are extracted from logs of different length of time spans. So the features are slightly different for training and testing dataset. Firstly, we fixed the length of time span to extract features, which we call "window". We use the logs of day followed the window to label the samples obtained from window. The day we used to label the samples is called "label-day". Positive samples obtained by original method is the active buying user-item pairs on label-day 12/18/2014 . The positive user-item pairs of one day are not enough to utilize the potential of features. We want to supply positive samples by adding more "label-days" in training dataset. Here is the framework of sliding window method. The original dataset consists of 31 days, we set the length of window as 10. Each window is followed by a label-day. Samples of one window are the active user-item pairs within the window. We extract features from logs within the window and label the samples with the buying behaviour of label-day. We select 12/18,12/17,12/16 . . . , 11/28 as label-days and construct samples from each corresponding window and label-day. We then gather the samples with different label-days. With sliding window, the positive samples are much more than that of traditional method. Besides, each sample's features is extracted from fixed length of window, so the features of training and testing dataset are similar.

C. Smoothing towards Double-12 festival

Taobao Marketplace, under the Alibaba Group Holding Ltd, will offer a variety of discounts to shoppers on Dec 12, which is so-called Double-12 festival. So the users are more active near Dec 12, especially the sales volume is even more than 400% of daily amount. If the tendency of users' behaviour on each day is stable with the time, the samples of different label-days are nearly independent identically distributed. Unfortunately, the burst of behaviours near Dec 12 will cause severe bias to the samples of training dataset. Here are the figures of count of behaviours on each day.

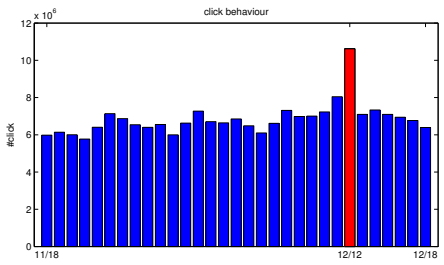

Fig. 2. Distribution of click behaviour

The figures demonstrate that all behaviours are abnormal on Double-12. To normalize the extreme bias of behaviour logs, we manually assign weights for behaviours on Double-12 while accumulating the count of behaviours for features. The Double-12 smoothing factor of click, collect, cart and buy behaviours are 0.63, 0.75, 0.45, 0.24. Besides, we will not select days from 12/11/2014 to 12/14/2014 as label-days, because buying behaviours are abnormal caused by the discounts on Double-12 festival.

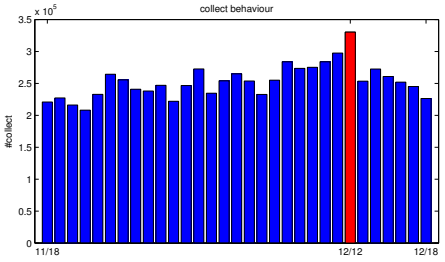

Fig. 3. Distribution of collect behaviour

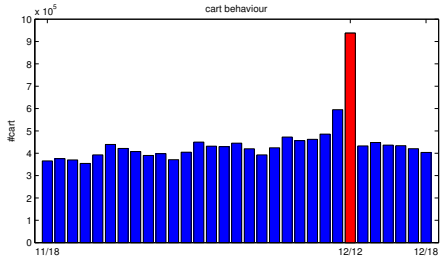

Fig. 4. Distribution of cart behaviour

D. The Calculation of Decaying Function

With sliding window method, we can extract features from logs within the window and label the samples using buying logs in the next day. Each log contain the operating time, so we can mine more information from operation with time stamp. Our team study the weight of each operation and build the relationship between operation weight and the gap of operating time and label-day(in short "operating gap"). The purpose of the competition is to predict purchase, so we focus our attention on the influence that "operating gap" has on buying potential.

It's not difficult to find that users' interest on an item will decline with time. We want to fit user's interest decaying function with time, then the weight of each operation can be seen

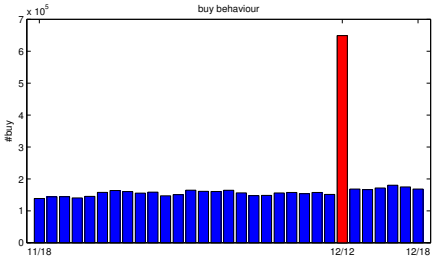

Fig. 5. Distribution of buy behaviour

as the corresponding decaying function value. So we fix one day as observing day and analyse the behaviour of last 20 days. For convenience, we discuss the click behaviour here. For each buying user-item pair on observing day, we can calculate the count of click of each day within the intervals of 20 days. After accumulate the count of different user-item pairs' behaviour, we estimate the distribution of click behaviour over the last 20 days before the observing buying day. The distributed function of click over time before buy behaviour are treated as the decaying function of click behaviour. We then use simple power function to fit the tendency of behaviours' distribution over time. Here are our fitting results:

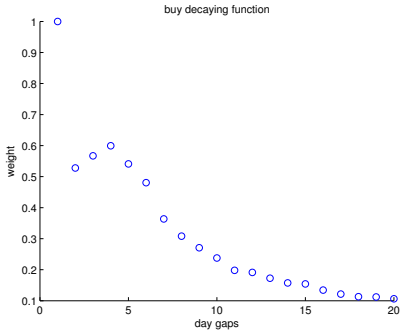

Fig. 6. Users' buying behavior decay function

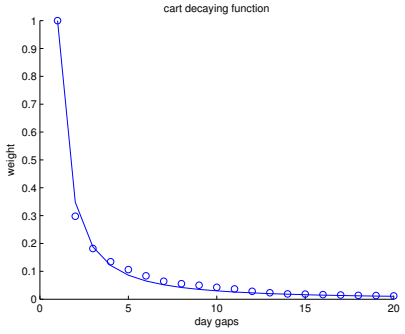

Fig. 7. Users' carting behavior decay function

The series of points are the real data and the line is the figure of our fitted function. From the figures we can find that the fitting is reasonable. In addition, we find that users' interest will decay quickly after click, collect and cart behaviour, while users' interest will increase in the first few days after buy behaviour. This meets our common knowledge that people usually do not buy the same items frequently.

IV. Feature Engineering

A. Feature Framework

The original data is a table of about 6 billion records, consisting of the behaviour logs of 5 million user and 156 million items from 11/18/2014 to 12/18/2014. Each record is a user's operation upon an item, containing the behaviour

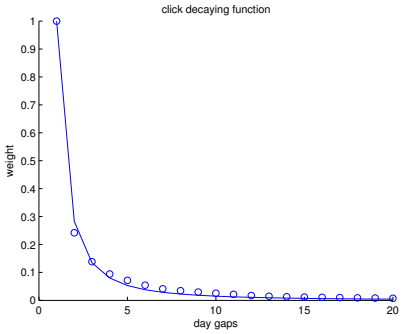

Fig. 8. Users' clicking behavior decay function

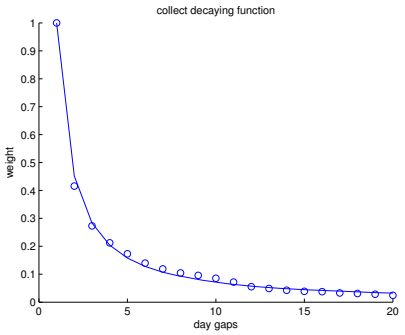

Fig. 9. Users' collecting behavior decay function

types(click, collect, add to cart, buy), operation time and the user's current location. Each item belongs to a category, catering or photograph. Based on the interactions between users and items, we build our features from the following 6 aspects:

1) user-item-features
2) user-category-features
3) user-features
4) item-features
5) category-features
6) cross-features

User-item-features and user-category-features capture the interactions between users and items or corresponding category, for example : "count total times of a user clicked an item". User-features reveal users' shopping habits, preference and purchasing power. Features of items and categories describe the inherent property of each item and category. Items in one category are similar to each other, so a category can be seen as the clustering result of items. Cross-features are the product or division of the previous features, generated from the features calculated above.

B. User-Item-Features

The goal of the competition is to predict the purchase behaviour of the users in the subset of items in 12/19/2014. Our team focus on the interactive user-item pair – the user once interacted with the items before the buying day. There

are two necessary conditions for the users to buy the item: 1) the user's behaviours count 2) the user's behaviours time. Then we can divide user-item-features into four groups: count features, time features, rank features, rule features.

1) Count features: For each user-item pair, we can calculate the counts of four behaviours. The binary features that denote whether the a behaviour happens are also useful. In order to reveal the tendency of a user-item pair, we calculate the counts of four behaviours within different time spans.

Examples for one user-item pair:

- count four behaviours with decay function and double-12 smoothing within window;
- whether four behaviours happened within window;
- count four behaviours in last **2** hours;
- count four behaviours in last **1** days;
- count four behaviours in last **3** days;

2) Time features: The last click time of one user upon one item is closely related to the probability whether user will buy this item. Besides, the days and hours of each behaviour indicate the user's interest towards an item. Examples of time features:

- the time gap since last behaviour;
- count the number of days of each behaviour;
- count the number of hours of each behaviour.

3) Rank features: Count features and time features are easy to come up. Rank features are the highlight of our features. Our idea is to predict the buying items of a user from his action logs. Generally, a user interacts with many items and chooses several to buy. Choosing items is indeed an item ranking process of the user. We can find some clues from click logs of the user, such as the ranking of the user's click count upon this item compared to other items. We construct some rankings of user-item pairs and obtain the rank features.

Examples of rank features:

- the order of a user adding an item to cart among items of same category;
- the ranking of a user's click number among items of the same category;
- the ranking of time a user spent among items of the same category.

4) Rule features: By studying the users' shopping habits, we find some rules that help predict the buying user-item pair. With just one rule, we can get about 5% F1 scores. We also change some rank features to binary features and take it as rule features. By applying those rules to obtain features, the models could perform better.

Examples of rule features:

- whether a user added an item into cart but didn't buy it in the last 12 hours;
- whether a user added an item into cart but didn't buy it in the last 1 day;
- whether an item was first browsed by the user among items of the same category;
- whether a user first added an item into cart among items belonging to the same category;
- whether a user interacted with an item for the most times among items belonging to the same category;
- whether a user spent most of his time on an item among items belonging to the same category.

C. User-Category-Features

Users will interact more frequently with one category compared with one item. So we can explore more habits and preference from user-category behaviours. User-category-features extends the count features of user-item, and combines the ratio features.

1) Count features: In addition to the basic count features of user-item, we further calculate the distinct items of a user clicked within different time spans.

2) Ratio features: Some ratio values reflect users' buying potential, such as the ratio of buying count and browsing count. Besides, we calculate the converting ratio of carting behaviour to buying behaviour.

- the ratio of buying counts and browsing counts of a user over one category;
- the ratio of buying items counts and browsing items counts of a user over one category;
- the converting ratio of carting&buying items counts and carting items counts of a user over one category;
- the average behaviours counts a user spent on each item over one category.

D. User-Features

In contrast to user-item-features which describe the relationship between users and items, user-features describe the property of users. User-features can also be divided into count features, time features and ratio features. Time features and ratio features are almost the same as that of user-category-features. For count features, we calculate users' behaviour counts, the number of clicked items and clicked categories within different time spans. Besides, we calculated the probability that a user will buy an item the next day after the carting behaviour.

E. Item-Features

To describe customers' attention on one item, we save the click count as well as the visited users count of each item. Besides, sales volume and bought users count are the criteria to evaluate the degree of best-selling of one item. The ratio features such as the ratio of buying and browsing are still kept here. We study two groups of special customers: "repeated buyers" and "jumper". "Repeated buyers" are customers who have bought this item for at least twice, while 'jumpers" are those who just browsed this for only once and leaved the page of this item. To distinguish hot items, we add items' rank features too. Here are examples of item-features:

- the counts of behaviours towards an item in the last 3 days;
- the counts of browsing, collecting, carting and buying users in the last 3 days;
- the ratio of buying users count and browsing users count;

- the convert ratio of carting&buying users counts and carting users counts of one item;
- the ratio of "repeated buyers" and "jumper";
- the ranking of bought users count among items of the same category;
- the ranking of buying count among items of the same category;
- the ranking of "repeated buyers" buyer ratio among items of the same category.

F. Category-Features

Most of category-features are similar to item-features. We add several count features to further describe the property of a category.

- the counts of browsing, collecting, carting and buying items in the last 3 days of this category;
- average behaviour counts of a user towards a category;
- the ratio of buying users count and browsing users count.

G. Cross-Features

After calculating the above 5 classes of features, we can select some features from different classes and make product or division.

- whether the user added an item to cart, but did not buy items of same category;
- the product of convert ratio features and binary rule features;
- a user's click count on one item divided by his average click count among the same category;
- whether an item is the only item the user browsed or added to cart in the last day among the same category.

V. Modeling

In this section, we will introduce the models in our solution. There are a myriad of models fitting data by nonlinear functions in machine learning applications, such as Neural Network[1], Decision Tree[2] and SVM[3]. In this problem, we choose Gradient Boosting Decision Trees as the main model, which is referred to as GBDT[4], [5]. It combines weak learners into a single strong learner in an iterative way and can handle heterogeneous data.

A. Gradient Boosting Decision Trees

We take the decision tree as the weaker learner in GBDT, specifically the Classification and Regression Tree (CART). The goal of the decision tree is to create a model that predicts the output based on several input variables. The major advantages of this method lie in avoiding normalizing features and the capability of handling nonlinear patterns. Further, it is robust and effective in handling large dataset. In the problem, we are facing with a huge amount of features whose ranges are quite different, e.g. the sum of user-item pair browse is in range [0,120], but some ratio features are in range [0,1]. Thus the tree-based models could free us from scaling these features. However, further optimization are stilled needed, since a single decision tree may not perform as well as we expect. We are supposed to ensemble methods such as Bagging, Random Forest, Boosting Trees to improve the model.

GBDT is a method combing Decision Tree and Boosting together. It is an accurate and effective off-the-shelf method that could be used for both regression and classification problems. The general idea of GBDT is to compute a sequence of simple trees in an addictive way, where each successive tree is built by the prediction residuals of the preceding tree. When adding a new tree, the parameters are optimized by the gradient of the loss function. In this problem, we build trees for distinct sampling data for a better model. While GBDT could generate expected predictions, it's sequential execution order makes it difficult to parallelize. For instance, though we only used 400 trees to train models, it still took more than ten hours on the Yushanfang platform. An implementation of GBDT can be found in the python package scikit[6]. Given the "training" sample $\{y_i, x_i\}$, we define our goal as a function $F(x)$ that map x to y.

B. Blending Method

Because every GBDT model is trained by differently sampled data, they predict the final results individually. To have the better performance, we combine each model together by Logistic Regression, a linear model for classification problems. The results of every model and the offline test set are used as the input. After training we can get the blending model, which yields a binary output for each user-item record. Finally, we fetch top k record to submit, where k is a experience value. As figure 2 shows, we adopt 11 GBDT models to in the solution.

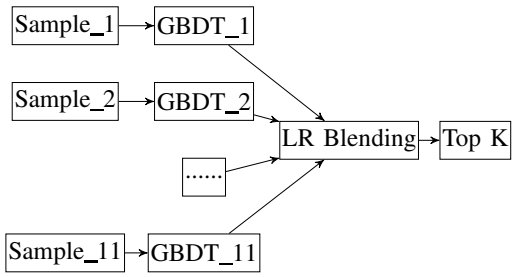

Fig. 10. Blend 11 GBDT outputs together by LR, and fetch top k records to be the final result

VI. Experiments

Our experiments are running on Alibaba's Yushanfang platform, a scalable distributed machine learning tool. From the start of the competition, we propose about 40 submissions in 45 days. The results and descriptions of our submissions are illustrated in table1.

In 6/24, there is a remarkable rise in the F1 score, since the organizers changed the online testing data from 50% users to 100% users. The offline score is lower than online score, since the offline dataset are in Thursday, while online dataset are in Friday. Compared the submission of 6/25 with 7/1, rank-features have a significant effect in improving the performance.

Date	Description	Offline Score	Online Score
5/16	basic features	4.98%	5.43%
5/23	basic features, bagging	5.05%	5.5%
6/10	mixed features(without rank-features)	5.24%	5.87%
6/11	mixed features(without rank-features), bagging	5.37%	5.99%
6/23	adjusting parameters, mixed features(without rank-features), bagging	5.46%	6.08%
6/24	mixed features(without rank-features), bagging	7.85%	8.5%
6/25	mixed features(without rank-features), LR-blending	7.88%	8.53%
6/29	mixed features, bagging	7.88%	8.49%
6/30	mixed features, LR-blending(8 models)	7.97%	8.62%
7/1	mixed features, LR-blending(11 models)	8.02%	**8.66**%

TABLE I
ONLINE AND OFFLINE F1 SCORE OF OUR SUBMISSIONS

Besides, we observe that generally LR-blending renders better results than bagging.

VII. CONCLUSION

Due to some technique issues, users' geological information are incomplete. Hence we do not put geological features into our feature framework.

In our final solution, we use 6 categories of features as the input, and train them by separated GBDTs. By combining all the outputs together, we get the final model which exhibits a good performance. By understanding users' shopping habit, we extract effective features to support our models. Besides, we overcome the difficulty caused by the sales burst on double-12 in original dataset. We believe that the our techniques could shed a light on the current commercial recommendation systems.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China grant 61173009, 61170062, 61222202, 61433014, 61502449 and the China National Program for support of Top-notch Young Professionals.

REFERENCES

[1] M. T. Hagan, H. B. Demuth, M. H. Beale *et al.*, *Neural network design*. Pws Pub. Boston, 1996.

[2] M. A. Friedl and C. E. Brodley, "Decision tree classification of land cover from remotely sensed data," *Remote sensing of environment*, vol. 61, no. 3, pp. 399–409, 1997.

[3] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[4] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[5] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *The Journal of machine learning research*, vol. 4, pp. 933–969, 2003.

[6] 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*, vol. 12, pp. 2825–2830, 2011.