A Novel Framework for Repeat Buyers Prediction with Feature Engineering

Bin Zhang, Xudong Liu, Qing Yang
National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
Beijing, China
bin.zhang, xudong.liu, qyang@nlpr.ia.ac.cn

Abstract

In this paper, we report the solution of our team "Farewell" on the IJCAI 15 competition. This competition provides a set of merchants and their corresponding new buyers acquired during the promotion on the "Double 11" Day and the task is to predict which users will become loyal customers for given merchants in the future. To solve this problem, we firstly do a lot of work in feature engineering. Features are extracted from various aspects and divided into several categories. Besides, we adopt some innovative methods to generate latent factor features and user-merchant cocluster features. In the first stage of the competition, we employ two individual models, Factorization Machines(FM) and Gradient Boosting Decision Tree(GBDT) and we adopt Logistic Regression(LR) model in the second stage. Afterwards, we propose a weighted linear ensemble model to combine results of individual models. Our final result is the blending of FM model and GBDT model in stage 1 and result of LR model in stage 2.

1 Introduction

IJCAI 15 Competition[1] aims to promote applications of advanced techniques from AI research to real-world problems. Task in this competition is to predict which new buyers for given merchants will become loyal customers in the future, by predicting the probability that these new buyers would purchase items from the same merchants again within 6 months.

The competition consists of two stages. In the first stage, two small datasets are provided, contestants can download the dataset, perform feature extraction and train with additional tools, only need to submit the prediction results for evaluation. While in the second stage, competitors will need to submit their code in JAVA, then the distributed computation will be handled by the cloud platform.

The dataset[2], which is provided by Tmall.com, consists of a set of merchants and their corresponding new buyers acquired during the promotion on the "Double 11" day. Each instance contains user_id, age_range, gender, merchant_id, label and activity_log. User_id and merchant_id indicates current user and merchant, age_range and gender are user attributes. Label can be "1", "0", "-1" or "NULL", "-1" means this user is not a new customer of the given merchant, "1" or "0" occurs only in training data, indicating user is a repeat buyer or not, and "NULL" only occurs in testing data, indicating it is a pair to predict. Activity log contains all interaction records between user_id and merchant_id pair, presented with the text format like "item_id#category_id#brand_id#time_stamp#action_type". Action_type could be "0","1","2" and "3", representing click, add-to-cart, buy and favourite action respectively. Data labelled "-1" are not to be predicted but competitors can extract extra information from it.

Dataset from stage 1 contains 7,030,724 instances in training set with 212,062 users and 7,027,944 in testing set with 212,108 users. While only 260,865 records are labelled "1" or "0" in the training set and 261,478 records need to be predicted in the testing set.

The evaluation metric for this competition is the Area Under the ROC Curve (AUC), which is equivalent to the probability that a random pair of a positive sample and a negative one is ranked correctly using the predicted result. During stage 1, the leaderboard is updated each 4 hour. But in stage 2, teams are allowed to submit their code once a hour, an evaluation on a small dataset will be carried immediately, but the evaluation on the whole dataset will be carried once a day. Our team participated the competition in both stages. According to the announced result, our approach achieved 3rd place in stage 1 and 10th place in stage 2.

The paper is organized as follows. Section 2 outlines the framework of our approaches. Section 3 discusses the preprocessing and feature engineering. Section 4 introduces the single effective models we use during the competition. . Finally we conclude this paper in Section 5.

2 Framework

In this section we will introduce the architecture of our system and software we used. Then we will introduce the internal validation set from the training set, which is important in model evaluating and combining different models.

[1] http://tianchi.aliyun.com/datalab/introduction.htm
[2] http://tianchi.aliyun.com/datalab/dataSet.htm

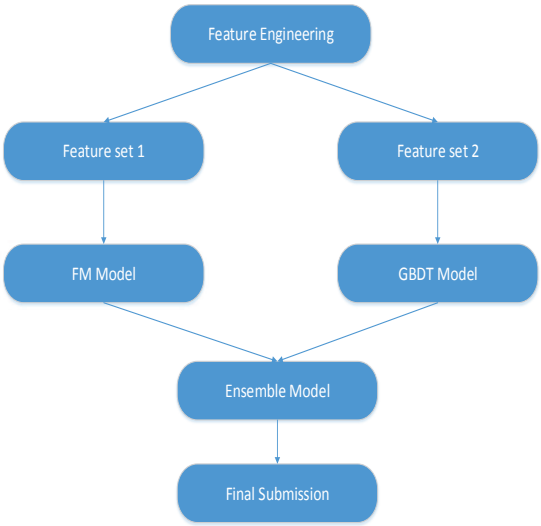

Figure 1: Framework of Proposed Model

2.1 System Overview

Our System consists of three parts: data infrastructure, training individual models and ensemble. We carry out our feature engineering in the first part. Features are divided into several categories. ID features, attribute features, counting features are basic statistic features. Ratio features and rules are generated by combining two or more basic features. At last implicit feedback features and cluster features are extracted. The output of the data infrastructure is features in the text format compatible for individual models we used in our method in stage 1. Detail of data preprocessing and feature engineering will be discussed in Section 3.

The second part of our system is training individual models. During stage 1, we employ two implementations in our training process, which are libFM[3][9] for FM model[8] and Xgboost[4] for GBDT model. Both of them are fast implementation in C++, and model can be trained in a relatively short time. While in stage 2, we simply extract the same features as used in stage 1, and employ LR model provided by the cloud platform for our training.

The last part of our system is an ensemble model with a weighted linear model combining two results from libFM and Xgboost. Enhancing the diversity can boost the performance when models are appropriately aggregated. We also do some parameter tuning in this part expecting to find the most proper weight for the two models. There is no ensemble model for stage 2.

[3]http://www.libfm.org/

[4]https://github.com/dmlc/xgboost

Table1: ID Features

Feature	Description
merchant_id	id of given merchant
brand_id	id of brand bought buy user
category_id	id of category bought buy user

2.2 Internal Validation Set

A validation set can be used to evaluate single model without submitting the test result to the leaderboard. And validation set is also very important for combining different models. In stage 1, the training set is divided into two parts with the ratio 1:1 in user size, one for training models and the other for validation.

3 Feature Engineering

Before introducing individual models implemented in our approach, we describe our feature engineering process, the most important work in our solution. Features extracted in our model can be allocated into several categories and each of them will be discussed below.

As introduced in Section 1, the dataset contains anonymous users' shopping logs in past 6 months before and on "Double 11" day, and the label information indicating whether they are repeat buyers. Label can be "1", "0" or "-1". "1" donates user is a repeat buyer for given merchant, while "0" is the opposite. "-1" represents that user is not a new customer of the given merchant, thus of out prediction.

About 90% of data instances are labelled "-1", which means a large portion of the labels are out of prediction. While it is obvious that user behavior on merchants in which he has bought before will have an effect on his behavior on a new merchant. At the same time, merchant will attract users with similar interests. Data labelled "-1" can provide plenty of information about user and merchant and these records will play an important role in feature engineering.

We simply call features extracted from one single line data as UserMerchant Features and use the name User Features for features from all records of one user and Merchant Features for features from all records of one merchant.

3.1 ID Features

Discrete ID features often performs fairly well in both FM model and GBDT model. In this competition, we firstly extract merchant_id of the merchant which user buy on the "Double 11" day. Then we get category_id and brand_id of items bought by user on "Double 11" day. Item_id is not included because one item_id belongs to one merchant only. IDs are integers but their values have no numerical meaning, so we have to expand ID features into binary features. For example, there are 4,995 merchants in total, the merchant_id feature should be a 4,995-dimensional binary feature vector and only one value can be "1", with a mapping casting the raw merchant_id to a merchant_id index. ID features are shown in Table 1.

Table2: Attribute Features

Feature	Description
user_age	age range of user
user_gender	gender of user
merchant_age_most	the most age range of merchant
merchant_gender_most	the most gender of merchant

3.2 Attribute Features

Attributes of a user, e.g. user age or user gender are important features. Often users with the same age or gender will have similar behaviour pattern to some degree. What's more, by collecting attribute information of all the users who have visited one specific merchant, we can easily get the distribution of user age and user gender for this merchant. Theses attribute are useful in repeat buyers predicting, for example, if most users of a merchant are female, a male customer will have a lower chance than a female customer becoming a repeat buyer for the given merchant. Based on that, we extract four attribute features, which are shown in Table 2, all these features are multi-dimensional binary feature, too. We use some tricks when dealing with merchant gender features. The merchant_gender_most feature has three value, which are "male", "female" and "balance". We set a threshold(e.g. 3) to balance the ratio between count of male users and count of female users. If count of one gender users is only slightly larger than the other, and the ratio between them is less than the threshold, we consider that the difference between two genders is not so obvious, thus putting the merchant_gender_most feature value "balance", which will be [0,0,1] in binary format. Table 2 describes attribute features used in our model.

3.3 Counting Features

Counting features are based on the statistic of activity log. In one single line of the dataset, we count the num of user action, then we divide each 1-dimensional numerical feature into a binary vector. Take click_num for example, click_num is the count of one user's click action on one given merchant, if it is counted the a have 5 clicks in total, then we expand click_num feature to a click_num_refined feature with a segment [0,1,3,5,7,10,∞], so the click_num_refined feature for this user merchant pair should be [0,0,0,1,0,0,0]. Besides, click_item_num is different from click_num but also important for the reason one user clicks a single item many times is definitely different from he clicks many items but only once for each item, both of which may have the same value in click_num feature. In general, we counted click_num, favourite_num, buy_num, click_item_num, favourite_item_num, buy_item_num as counting features and each of them is refined by a specific segment.

Date related counting features are extracted here also. One is date_num and the other is day_during. Date_num is the count of dates the user has visited the given merchant except the "Double 11" Day, here visit means click, favourite or add-to-cart. Day_during is the interval of days between "Double 11" Day and the day user visited this merchant for the first time. A large date_num or a large day_during value may represent user has long lasting interest in the given merchant.

There are still many user counting features and merchant counting features, but they have little effect in our model for the overall count of one user have limited effect on the behaviour to one specific merchant, as well for one merchant. We list counting features in Table 3.

Table3: Counting Features of User and Merchant Pair

Feature	Description
click_num	num of click action
buy_num	num of buy action
favourite_num	num of favourite action
click_item_num	num of items clicked
buy_item_num	num of items bought
favourite_item_num	num of items favoured
date_num	num of date user visited
day_during	interval of days

3.4 Ratio Features

Counting features may not work in linear models, especially for user counting features and merchant counting features. Instead, we generate ratio features by combining two or more counting features. They are simple but efficient in our models and most are date related.

We generate more than 20 ratio features in total. Most of them have a value between 0 and 1 and features whose value may be larger than 1 are refined by segments with the same method mention above. To illustrate how we calculate these features, we take some as example and the full list of key ratio features as shown in Table 4.

For a user and merchant pair, We compute the ratio between count of items bought and count of items clicked, named as click_buy_ratio as well as favourite_buy_ratio for representing the ratio between buy action and favourite action. More date related ratio features are also calculated here. Take activity_ratio_on_de(de means "Double 11" Day here) for example, it represents the ratio between count of activities occured on "Double 11" Day and the total count of activities. If activity_ratio is relatively low, it shows the user may have focused this merchant with everlasting interest on it and is not mainly attracted by the promotion on the "Double 11" Day, thus he may have a higher probability becoming a repeat buyer.

User ratio features are extracted from several aspects. A user can be a fast buyer, meaning he prefers buying what he likes directly without comparing with many other similar items, will have a higher user_click_buy_ratio than a slow buyer, which means he would buy one item after comparing many times. From another point of view, a user may prefer to buy new items in a merchant in which he has bought before rather than a new merchant, his value of user_avg_merchant_click may be much larger than other users.

There are two kinds of ratio features relating to date. One is "de feature", it separates activity occured on the "Double 11" Day from others, e.g user_buy_on_de_ratio, representing the how many he buys on "Double 11" Day. The other is "month

Table4: Key Ratio Features

UserMerchant Ratio Features	User Ratio Features	Merchant Ratio Features
click_buy_ratio	user_click_buy_ratio	merchant_click_buy_ratio
favourite_buy_ratio	user_favourite_buy_ratio	merchant_favourite_buy_ratio
buy_ratio_on_de	user_buy_on_de_ratio	merchant_buy_on_de_ratio
	user_new_merchant_ratio	merchant_old_user_buy_on_de_ratio
	user_old_merchant_buy_on_de_ratio	merchant_click_ratio_by_month
	user_old_merchant_buy_ratio	merchant_buy_ratio_by_month
	user_click_ratio_by_month	merchant_more_than_once_ratio
	user_buy_ratio_by_month	merchant_new_ratio
	user_favourite_ratio_by_month	

Table5: Key Rule Binary Features

Feature
has_click_before_double_11_day
has_favourite_before_double_11_day
has_add_to_cart_before_double_eleven_day
only_occur_on_double_11_day

feature", where activity logs are separated by month.
Merchant ratio features are similar with user ratio features but more impressive. We have merchant_new_ratio feature for showing the percentage of the new customers. Take another month feature for example, we may find a merchant sells much more in July and August than any other month, indicating this merchant sells many seasonal items and will have a lower chance to be bought in the next 6 months.

3.5 Rules

We design a lot of binary features by rules, which are all 1-dimensional features and the value can be 1 or 0 only. Rules are based on the understanding of the dataset. We expect these rules to distinguish users who will have persistent interest on one merchant from users who will not. Most of these features are date related. A rule feature called only_occur_on_double_eleven is designed. If a user has clicked or favoured items in one merchant before "Double 11" Day, the value of only_occur_on_double_eleven should be 0, while in the opposite, the value should be 1. Another rule feature has_click_before indicates whether the items user buy on "Double 11" Day have been clicked before. Some useful rule binary features are shown in Table 5.

3.6 Implicit Feedback Features

In recommender system, we only have implicit feedback between users and items sometimes. There are many works[5] in recommending with only implicit feedback data. In this competition, we treat a merchant as an item in recommendation system and we get an implicit feedback matrix between users and items. Then we try some different approaches using implicit feedback data and we put the results as features in our proposed model.

Consider a situation with only interaction between users and merchants. If two users share many merchants, they may have similar preferences on merchants, which means if one user becomes a repeat buyer of a new merchant, the other is more likely to be, too. Our first implicit feedback feature is built from the Matrix Factorization(MF) model[6] with a constructed implicit feedback matrix. The MF model is a Collaborative Filtering(CF) method and decomposes this matrix into two set of low-dimensional latent factors, which are representation of user and merchant preferences.
To construct this matrix, for one single user, we give the "old" merchants, which is labelled "-1" a value "1" in the implicit matrix, and "new" merchants, including merchant labelled "0" or "1" and merchants with no activity log in the past six months, a value "0". One user has activity with only a small part of all the merchants, so the implicit feedback matrix is sparse. Then we employ the MF model with a toolkit libMF[5] to train this model. We directly put the inner product of one user latent factor and one merchant latent factor into our model as a implicit feedback feature.

3.7 User and Merchant Topic Feature

Topic model is a typical statistical model in natural language processing(NLP) area and also used in machine learning area. One of the most classic models is Latent Dirichlet Allocation(LDA)[1]. We adopt the method proposed in our previous work[7] to generate user and merchant topic features. We view a user as a "document" and merchants he has visited as "words" in this "document". Then an open source tool Gibbs LDA++[6][4] is used to generate the latent factors with the dimension of the latent factor set to 10. We consider user latent factor as user topic features and merchant latent factor as merchant topic features and then put them into our training model directly.

3.8 Cluster Feature

Partitioning users and items into subgroups for CF method has been studied by several works[10]. In our approach, we adopt a co-cluster method we proposed which is similar with previous work but haven't been published yet.
Specifically, users and merchant can be grouped into multiple overlapping subgroups, where users in a subgroup will have

[5]http://www.csie.ntu.edu.tw/ cjlin/libmf/
[6]http://sourceforge.net/projects/gibbslda/

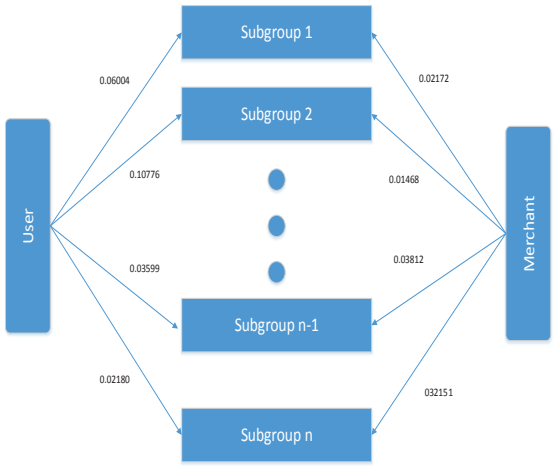

Figure 2: Example of co-cluster result

similar preferences and merchants have same property. Overlapping means each user or merchant can belong to multi subgroups and with a weight indicating its probability belonging to one subgroup. We set up an example in Figure 2.
It is obvious that if a user and a merchant both have a high probability belonging to the same subgroup, they have a good chance building a strong connection in the future, which means becoming a repeat buyer in this competition. Two methods are adopted to evaluate the strength of user and merchant connection. The first one is calculating the inner product of user cluster feature vector and merchant user cluster feature vector as cluster feature, and the value should between 0 and 1. In the other method, we set a threshold(e.g. 0.2) to evaluate whether a user(or merchant) belongs to one subgroup and a weight larger than the threshold will be seen as this user(or merchant) belongs to it. The count of subgroups a user and a merchant both belongs to will be used as cluster feature. In our experiment, we set the subgroup number to 10, and adopt the first method to generate cluster feature.

4 Models

In this section we will introduce several different models used in our system. In general, we employ Factorization Machines(FM)[8] model with the toolkit lifFM[9] and Gradient Boosting Decision Tree(GBDT) model with the toolkit Xgboost in the first stage of the competition. The final submission of stage 1 is a blending with a weighted linear ensemble model. In stage 2, we only use the Logistic Regression(LR) model provided by the cloud platform.
Features used in three models are similar but not the same, detail about feature selection will be discussed below.

Table6: FM model performances with different features in stage 1

Features	AUC Performance
Random Guess	0.500000
+ ID Features	0.670354
+ Attribute Features	0.672786
+ Counting Features	0.689357
+ Ratio Features	0.694432
+ Rules	0.696223
+ Implicit Feedback Features	0.701514

4.1 Factorization Machines

Factorization Machines(FM) is widely used in Recommender System. In this competition, we employ the implementation libFM to train a FM model. The learning method in our model is Markov Chain Monte Carlo(MCMC), iteration is 200 and other parameters with default values.
We conduct a series of experiments using FM model, and features extracted above are put into model step by step. ID features and attribute features are used for training model first, then counting features, ratio features and rules. Feature selection is carried out carefully during this step, for the reason mentioned before not all features are useful in our training model. We test our model on both internal validation set and public leaderboard. Useless features who make AUC performance decline on internal validation set and overfitting features making the AUC performance gap between validation set and public leaderboard are filtered out, thus only some key features kept.
Finally we try implicit feedback features. Experiment result shows that implicit feedback can make a big promotion in our FM model. Detail about AUC performance promotion with different feature types are shown in Table 6.

4.2 Gradient Boosting Decision Tree

A single model(e.g. a decision tree) is not powerful enough for describing the whole data, while a boosting model[3] can produce a prediction model in the form of an ensemble of weak prediction model. Gradient Boosting Decision Tree(GBDT) is a generalized boosting model which augments the power of single decision tree[2]. In our approach, we employ Xgboost, a fast implementation of GBDT to train our model. Features are similar compared with FM model, while the counting features need not to be divided into bins and features selection is not demanded for GBDT can do feature selection automatically. Besides, implicit feedback features are not included in GBDT model. With some parameter tuning, the best AUC performance is 0.696524 on public leaderboard.

4.3 Logistic Regression

Logistic Regression(LR) model is classical model for binary classification. We adopt LR model with the implementation provided by alibaba cloud platform in the second stage of this competition. Features used are nearly the same as FM model. We have to mention that because of the unavailability of the

whole dataset, all merchant features are built from dataset of stage 1 and merchant counting features are removed. Based on the public leaderboard, our method achieve 0.704467 on the full dataset.

4.4 Ensemble

Final submission of stage 1 is a blending of FM model and GBDT model. We adopt a weighted linear to blend the results from two models. Weight of MF model and weight of GBDT model sum to 1 but not equal. We tried several pairs on our internal validation set, and 0.67 for MF model and 0.33 for GBDT model performs best. The final result is 0.704084 on the public leaderboard.

5 Conclusion and Future Work

We describe our solution of our team in the IJCAI 15 competition in this paper. The most important work in our approach is feature engineering. Features are divided into different categories and each category performs fairly well in our model. Meanwhile, we try three innovative methods to generate implicit feedback features, which play an important role in finding similar users and merchants. Afterwards, three models are adopted in our solution, FM model and GBDT model for the first stage and LR model for the second stage of the competition. Finally, an ensemble model is proposed to aggregate results from different models to the boost performance of final submission.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, pages 993–1022, 2003.

[2] T. Chen, L. Tang, Q. Liu, and D. Yang. Combining factorization model and additive forest for collaborative followee recommendation. *KDD*, 2012.

[3] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals pf Statistics*, pages 1189–1232, 2001.

[4] G. Heinrich. Parameter estimation for text analysis. *Technical report*, 2005.

[5] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. IEEE*, pages 263–272, December 2008.

[6] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, pages 30–37, 2009.

[7] X. Liu, B. Zhang, T. Zhang, and C. Liu. Latent feature based fm model for rating prediction. *Recsys Workshop*, 2013.

[8] S. Rendle. Factorization machines. *Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE*, pages 995–1000, December 2010.

[9] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2012.

[10] B. Xu, J. Bu, C. Chen, and D. Cai. An exploration of imporving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st International Conference on World Wide Web*, pages 21–30, New York, NY, USA, 2012. ACM.