Report for Repeated Buyer Prediction Competition by Team 9*STAR*

Liu Guimei, Tam T. Nguyen, Zhao Gang, Zha Wei, Yang Jianbo, Cao Jianneng, Wu Min, Zhao Peilin, Chen Wei
Institute for Infocomm Research, A*STAR, Singapore
{liug,nguyentt,zhaog,zhaw,yang-j,caojn,wumin,zhaop}@i2r.a-star.edu.sg

Abstract

Repeated buyer prediction is an important task for ecommerce and would make considerable impact on merchants' return on investment (ROI). Recently, Alibaba has announced a two-stage worldwide competition for repeated buyer prediction[1], in which a huge amount of transaction data generated by Tmall.com in the year of 2014 is provided to be analyzed. In this competition, our team *9*STAR* has won the first place at Stage 1 and the seventh place at Stage 2. In this paper, we present our complete solution for both stages. The solution mainly contains two parts: feature engineering and model training. For feature engineering, we provide various techniques to extract a large amount of features to characterize the insights of the data. For model training, we first utilize several advanced machine learning algorithms as the predictive model to approach the problem, and then by comprising them we develop a sophisticated blending learning algorithm to further boost the prediction performance. Through both local testing and the competition's online evaluation, it turns out that our method is a competitive solution for the repeated buyer prediction problem.

1 Introduction

Merchants like Amazon and Alibaba run big sales promotions on particular dates (e.g., "Black Friday" and "Double 11 (Nov. 11)") to attract customers, and wish them to come back again to become loyal customers. However, most customers, especially those online, are one-time deal hunters. Promotions to them do not generate the *return on investment* (ROI) expected by merchants. This competition jointly organized by Alibaba and the IJCAI-15 workshop is to predict repeated buyers for Tmall.com, the biggest B2C platform in China. In particular, given a buyer-merchant pair, where the buyer made the first purchase from the merchant on double 11 via Tmall.com, the contestants of the competition need to apply sophisticated machine learning techniques to predict if the buyer will purchase items again from the same merchant in the next 6 months after "Double 11". Such a competition is very challenging, since repeated purchase is of low probability. But it is very meaningful – merchants with accurate prediction results can provide targeted and personalized promotions to loyal customers, thus not only reducing the promotion cost significantly but also increasing their ROI.

Our team *9*STAR* has participated in the two stages of the competition, and luckily won the championship of Stage 1. In Stage 1, the organizer provided datasets in two formats, of which we selected the second, since it is informative and clearly records the buyer-merchant interactions. The dataset is more than 2 GB and includes 4 files: user behavior logs, user profile, training and testing data. The log file gives the activities (e.g., click, putting into cart, and buying) between 424,170 buyers and 4,995 merchants, and the user profile gives the particulars of buyers, such as age range and gender. The training file contains 260,864 buyer-merchant pairs, with each being labeled, indicating whether the buyer purchases items from the merchant in the next 6 months after "Double 11". The testing file contains 261,477 buyer-merchant pairs, for which the class labels are to be predicted. In Stage 2, the competition runs not locally as in the Stage 1, instead it runs on the cloud platform of Alibaba. In this stage, the dataset is much bigger. But neither the training nor the testing data has been disclosed.

We have generated around 2,000 features, and applied various state-of-the-art classification models on them. It is worth mentioning that we applied cross validation in Stage 1 of the competition to improve the robustness of our prediction results – all the feature engineering and model training/tuning were validated by 10-fold cross validation. On a high level, our approach consists of two steps: *feature engineering* and *model training*. They are summarized as follows:

- **Feature Engineering**. We studied the user behavior log and user profile, and generated features describing the popularity of sellers, the preference of buyers, and also the activities (e.g., purchase) between buyers and sellers. In addition, we have also extracted novel features, such as buyer's purchase trend over the 7 months before double 11, features based on LDA [Blei *et al.*, 2003], as well as PCA-based features of similarity between mer-

*Winning Entry in Stage 1 of the IJCAI-15 Repeated Buyer Prediction Competition

[1]The website is at http://ijcai-15.org/index.php/repeat-buyers-prediction-competition.

chants.

- **Model Training**. We have trained various classification models. The important ones include Factorization Machine, Logistic Regression, Random Forest, XGBoost [Chen, 2014], and GBM, of which XGBoost is the major player and provides a best balance between prediction accuracy and training time. To further improve the performance, we have also proposed ensemble techniques to blend multiple classifiers together. In particular, we iteratively ensemble XGBoost with the thus-far best blending model, and incrementally improve the prediction score.

2 Feature Engineering

2.1 Statistics Related Feature

For each user, merchant, brand, category, and item, we computed the frequencies of actions (e.g., click, purchase, and putting into cart) based on months. We then generated histogram features by calculating min, mean, median, max of these frequencies. Therefore, we have user histogram features, merchant histogram features, and so on.

2.2 User Based Feature

Repeated Merchant Feature. For each user in every month, we calculated the number of merchants, from which the user has bought items. This feature shows how much the user likes online shopping and the portion of time (s)he has spent on Tmall.

Action Related Feature. For each user in every month, we also counted the times of his/her actions on category, brand and item. These features give a view of the user's shopping behaviors. Usually, more clicks imply higher chance to make a purchase.

2.3 Merchant Based Feature

Repeated Buyer Feature. For each merchant in every month, we calculated its number of repeated buyers. The trend of monthly repeated buyers serves as an important feature in the prediction.

Action Related Feature. For each merchant in every month, we also counted the number of different actions that buyers put on category, brand and item. These features give a view of the merchant's exposure. Usually, more exposure implies more repeated buyers.

2.4 Trend Feature

By comparing the features in each month, we further derived their trends as our second-layer features as follows.

1. Percentage of monthly difference (PMD): we calculated the different between the current month and the mean of the previous 6 months (i.e., $PMD = \frac{x_7 - mean}{mean}$, x_7 is the feature in current month while $mean$ is the average over last 6 months).
2. Normalized monthly difference (NMD): similar to PMD, NMD is normalized by the standard deviation (σ), i.e., $NMD = \frac{x_7 - mean}{\sigma}$.
3. Slope of the trend as shown in Figure 1.

Trend features give a historical view for both merchant and user. These features can help the model to understand merchants and users better. We have ranked our features according to their importance to the model and found that those trend features are very important for the prediction.

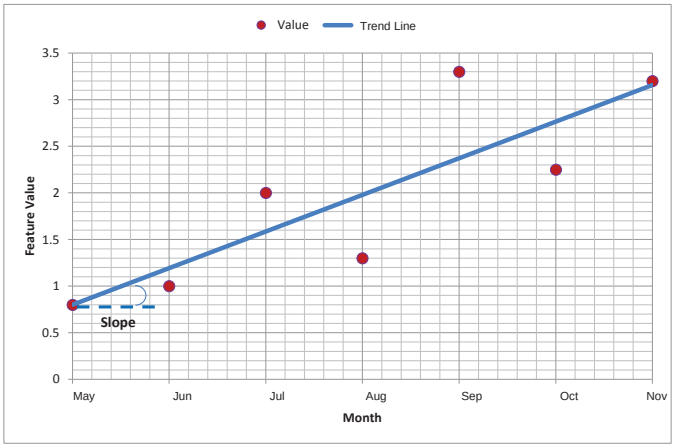

Figure 1: The Slope of Trend

2.5 Other Features

Purchase Frequency Feature. We have extracted features using *purchase frequency*, which is a strong factor indicating the willingness of customers buying items. However, instead of computing the purchase frequency on each item, we generalized it by computing the frequency on each category of items. Such a generalization is to make the derived features generally applicable. Given a category and a time interval (e.g., June), we compute the average number of times a customer purchases the items of the category within the time interval, and take it as the category's purchase frequency. We have also computed the category's purchase frequency over the whole time span before double 11. These frequencies altogether are used as the features to represent the purchase popularity for each category of items. In a similar way, we have also computed the purchase frequencies for each brand of items.

Penetration Feature. Penetration is another important factor indicating the popularity of items. Given an item and a time interval, we define the penetration as the number of customers, who have purchased the item in the given time interval. Again, we generalized the result by computing penetrations on a category (brand) of items, and by considering the penetration in each month alongside the whole time span before double 11.

Merchant Similarity Feature. We have generated features to describe the similarity between merchants. Give a pair of merchants, we use the number of customers, who buy items from both of them, as their similarity. The total number of merchants is 4,995. Therefore, a matrix of 4,995×4,995 is built. This matrix is highly sparse with most elements equal to 0. Simply adding it into the feature list does not obviously

improve the accuracy of classification models, but at the same time dramatically increases the model training time. As such, we have applied PCA (principal component analysis) [Lê *et al.*, 2008] on the similarity matrix, and kept only the top-10 principal components. Then, for each merchant the top-10 principal coordinates are generated as the features.

LDA Feature. It is intuitively to use all user ids and merchant ids for each user-merchant pair to identify the users' preference and corresponding merchants' quality. However our model cannot deal with such a high dimensional(each id cause one dimension) spaces. We propose to use the Latent Dirichlet allocation (LDA) method to reduce the dimension for the id features. LDA is used for extract topics from documents based on the words. Here for each user, we model a user as document, while all the merchant id s/he purchased before in log as the words. We can get a distribution for each user in a pre-defined lower dimensional space. Using the similar idea, we can model each merchant with associated user ids to get another distribution. We use 200 dimension in this completion.

2.6 Feature Selection

Different types of features are extracted as mentioned above, and the amount of such extracted features almost reaches two thousand. It is highly desirable for both the researchers and e-commerce businessmen to understand the relative importance of these features. Moreover, the feedbacks of feature importance also help us refine the feature engineering work in Section 2. To evaluate the importance of features, we developed a feature ranking method[2], in which an importance score is computed for each feature.

Feature ranking is one of fundamental tasks in machine learning problems. In this task, a feature evaluation criterion is often proposed to evaluate the importance of a feature. Based on the connection of the evaluation criterion to the underlying learning algorithm, feature ranking methods can be classified into two categories: filter and wrapper methods [Guyon *et al.*, 2002; 2006]. Filter methods are independent of the underlying learning algorithm while wrapper methods exploit the knowledge of the specific structure of the learning algorithm and cannot be separated from it. Typically, wrapper methods have better performance than filter methods but carry with them a heavier computational cost. Driven by performance, we focus on developing a wrapper method in this competition.

The wrapper method we developed is mainly derived from [Shen *et al.*, 2008; Yang *et al.*, 2009; Yang and Ong, 2011]. In these feature ranking methods, the importance of a feature is measured by the aggregate difference, over the feature space, of the probabilistic outputs of the underlying learning algorithm with and without the feature. The larger the aggregate difference, the more important this feature. The main computational cost of these methods is to remove the feature to be evaluated in the dataset and then compute the probabilistic outputs of the underlying learning algorithm on this updated dataset for each feature. It has been proved that random permutation of the values of a feature has the same effects as removing the contribution of that feature in these methods[3]. Therefore, these feature ranking methods enjoy the computational advantage over other wrapper methods. Different from [Shen *et al.*, 2008; Yang *et al.*, 2009; Yang and Ong, 2011], in this competition we employ the algorithms described in Section 3.1 as the underlying algorithms and compute the corresponding probabilistic outputs accordingly.

We also evaluate the self-contained feature ranking methods in XGBoost. Among them, three different criteria are used to evaluate the importance of a feature: (1) gain contribution of each feature (*i.e.* gain metric); (2) the number of observation related to each feature (*i.e.* cover metric); (3) the relative number of times a feature taken into trees (*i.e.* frequency metric). Generally, these feature ranking methods are also wrapper methods, as they are all embedded into the underlying learning algorithms. The first evaluation criterion is recommended by XGBoost, and we also found that the feature ranking method with this criterion can yield the similar feature ranking results by the one with the criterion proposed in [Shen *et al.*, 2008; Yang *et al.*, 2009; Yang and Ong, 2011][4], so we also use it as a reference to evaluate the importance of all features.

3 Modeling Techniques and Training

In this competition, we have used many supervised learning models including linear and ensemble. We trained multiple models and made the final prediction based on the linear combination of all models. The details of training both single models and blending model are as follows:

3.1 Single Model

Based on our full feature set, we generated one dense dataset and one sparse dataset. While the dense dataset was used to train ensemble models, linear models were trained based on sparse one.

Linear Model

For linear models, we used both Logistic Regression and Factorization Machine which are two linear successful algorithms in data science. For Logistic Regression, we used both implementations in scikit-learn[5] and Generalized Linear Model (GLM)[6].

For Factorization Machine, we used the field-aware factorization machine implementation that was successfully applied to the Kaggle competition[7].

Ensemble Model

For ensemble models, we used the best popular algorithms such as Random Forest, Gradient Boosting Machine, and eX-

[2] In some literature, feature ranking task is known as feature selection task.

[3] The technique of random permutation was also used in feature selection for random forest [Breiman, 2001], in which theoretic analysis is however not provided.

[4] The difference mostly lies in the rear part of the ranking lists.

[5] http://scikit-learn.org/

[6] http://cran.r-project.org/web/packages/glm2/glm2.pdf

[7] https://github.com/guestwalk/kaggle-avazu

treme Gradient Boosting. We also used a meta bagging classification algorithm implemented in scikit-learn.

3.2 Blending Model

After collecting the predictions from various single models, the blending model refers to a weighted vote of these predictions in Equation 1.

$$p(u,m) = \sum_{i=1}^{n} w_i \times p_i(u,m), \quad (1)$$

where $p(u,m)$ is the final probability that an user u will have a repeated purchase in a merchant m. $p_i(u,m)$ is the probability predicted by the i^{th} single model and w_i is the weight assigned to the i^{th} single model ($1 \leq i \leq n$, n is the number of single models we used).

In our experiments, we proposed two strategies to assign the weight w_i to the i^{th} single model. First, since we already obtained the AUC scores for individual models, we manually assign the higher weights to those single models with better AUC scores. Second, we built a classifier (e.g., linear SVM) to learn the weights for single models. In particular, we generated a $n-$dimension feature vector for each user-merchant pair (u,m), and the i^{th} element of this vector is the probability $p_i(u,m)$ predicted by the i^{th} single model. Then we can learn the weights by training the classifier we selected.

As we know, the probability scores predicted by different single models may have different distributions. We utilized the rank of the user-merchant pairs to obtain the final probability scores. Given N user-merchant pairs, we sorted them in an increasing order of their predicted probabilities and $R_i(u,m)$ is the rank of the predicted probability for the pair (u,m) by the i^{th} single model. Particularly, the final probability $p(u,m)$ is then the weighted sum of $\frac{R_i(u,m)}{N}$ instead of $p_i(u,m)$ in Equation 1. Meanwhile, the weight w_i can be similarly obtained by the above two strategies.

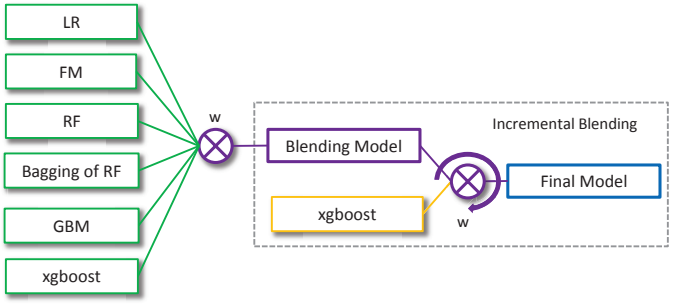

Figure 2: Incremental Blending Approach

Moreover, we also tried an incrementally blending strategy as shown in Figure 2. In this approach, we selected the best blending model and combined it with a new XGBoost model to improve the AUC score. We keeped adding new XGBoost model until we could find a better one.

3.3 ODPS Cloud Platform Deployment

In Stage 2, we have implemented our feature engineering and predictive models in the Open Data Processing Service (ODPS) platform as shown in Figure 3. We stored all of our feature values in an auxiliary data table. We have developed a feature extractor to load this data together with the setting information such as feature type and feature size to generate features for both training and testing data. While the mapping function generated a list of feature names (keys), the reformat function populated key-value sparse data. The output of the extractor was list of key-value pairs, merchant identifier, user identifier, and label (repeated/not-repeated buyer) that would be converted into a sparse matrix for training the predictive model. In this stage, we used Gradient Boosted Decision Tree (GBDT) algorithm provided by the platform to train our model.

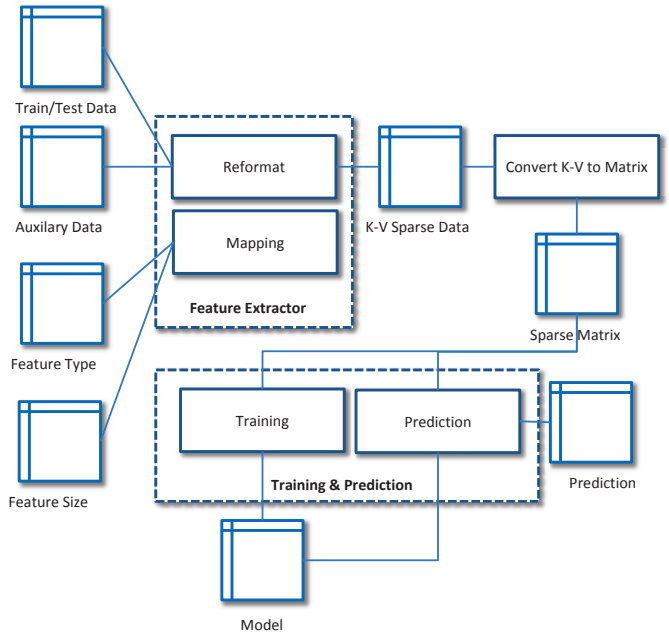

Figure 3: Implementation on the ODPS Platform

4 Performance Evaluation

4.1 Performance on Stage 1 Dataset

In this section, we present our best single model performance. Figure 4 shows the best AUC score for each single model. For linear models, Logistic Regression and Factorization Machine have similar AUC scores, 0.68097 and 0.67982, respectively. Although their scores are not really high but they can contribute to the overall AUC score in the blending model. In ensemble algorithm family, Random Forest has the worst AUC score. However, we found that bagging of Random Forest models can improve the score significantly. eXtreme Gradient Boosting has the best AUC score 0.70282. Compared the runner up Gradient Boosting Machine, the improvement is more than 0.7%.

We have blended 20 single models with various parameter settings to achieved the best AUC 0.70494. Compared to the best single model, the blending model has improved 0.21%. We have applied incrementally blending strategy to

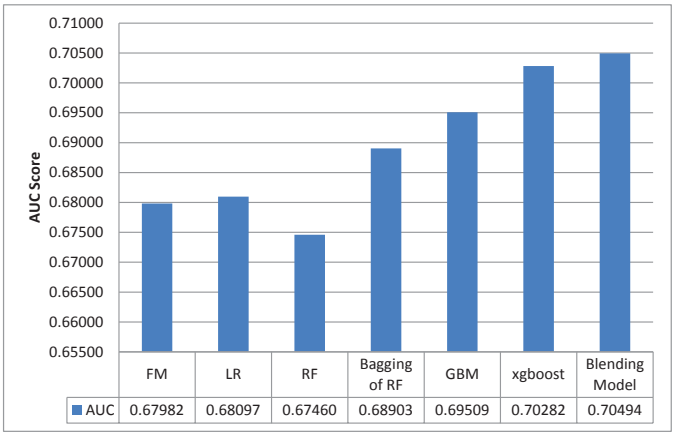

Figure 4: Performance of Best Single Models on the Test Dataset

our blending model. It means that we selected our best blending model and added one new XGBoost model at a time. This helps our blending model score gradually increases one by one. Figure 5 shows the XGBoost performance and its blending model. The blending model consistently performs well together with the XGBoost model.

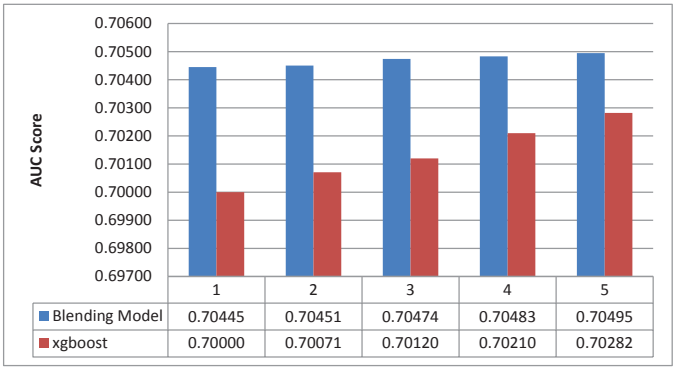

Figure 5: Performance of XGBoost Model and Its Incremental Blending Model on the Test Dataset

4.2 Performance on Stage 2 Dataset

Our advantage in Stage 1 is a rich feature set and an incremental blending model. Although we completed Stage 1 with the first place, we could not make it in Stage 2 where we had to use a single model and running time could not exceed 4 hours.

In order to balance between the running and the AUC score, we have implemented a Java program that the number of features and feature set can be changed using pre-defined settings. It means that if we would like to change the feature set, we only need to change the settings without revising the source code.

To reduce the running time, we can decrease the number of features and vice versa. We have applied feature selection technique as mentioned in Section 2.6. Based on our experience, the best AUC score 0.70897 in Stage 2 dataset can be achieved by selecting top 1000 features and training a GBDT model with the running time about 4 hours. Its parameters are max depth=7, learning rate=0.02, and the number of trees=1200.

One of our major findings in Stage 2 is to speed up the feature extraction process. In the reformat function, we had to randomly access a data list to generate features. However, we noticed that the pre-defined data loading function for each vertex created a linked list in Java. Since linked list is not a good choice for random access, we have revised the source code to use an array list. This trick reduced our feature extraction running time by 5 times so that we could spend more time on the training and prediction phases.

5 Conclusions

In this competition, we discovered several novel insights such as trends and LDA features. We observed that feature engineering is the key and it is best modeled by the Gradient Boosting Machine. We used an incremental blending method in Stage 1 to obtain a better blending model. In Stage 2, due to some objective facts, we could not migrate all our features and modeling methods into Stage 2, and hence, this report may help for further exploitation and implementation on this problem.

Acknowledgments

We would like to thank IJCAI-15 and Alibaba for organizing the competition, so that we have this precious opportunity to benchmark our algorithms and technologies. We would also like to thank our Data Analytics Department management at Institute for Infocomm Research.

References

[Blei *et al.*, 2003] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[Breiman, 2001] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001.

[Chen, 2014] Tianqi Chen. Xgboost: extreme gradient boosting. https://github.com/dmlc/xgboost, 2014.

[Guyon *et al.*, 2002] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, March 2002.

[Guyon *et al.*, 2006] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh. *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[Lê *et al.*, 2008] Sébastien Lê, Julie Josse, and François Husson. Factominer: An r package for multivariate analysis. *Journal of Statistical Software*, 25(1), 2008.

[Shen *et al.*, 2008] Kai-Quan Shen, Chong-Jin Ong, Xiao-Ping Li, and EinarP.V. Wilder-Smith. Feature selection via sensitivity analysis of SVM probabilistic outputs. *Machine Learning*, 70(1):1–20, 2008.

[Yang and Ong, 2011] Jian-Bo Yang and Chong-Jin Ong. Feature selection using probabilistic prediction of support vector regression. *IEEE Transactions on Neural Network*, 22(6):954 – 962, 2011.

[Yang *et al.*, 2009] Jian-Bo Yang, Kai-Quan Shen, Chong-Jin Ong, and Xiao-Ping Li. Feature selection for MLP neural network: The use of random permutation of probabilistic outputs. *IEEE Transactions on Neural Network*, 20(12):1911 – 1922, 2009.