Repeat Purchases Prediction

- A machine learning competition held by Alibaba Cloud

Frank, Diana, Joe

CONTENTS

1. Introduction

1.1 Background

1.1.1 The necessity of predicting repeat buyers

Merchants sometimes run big promotions (e.g., discounts or cash coupons) on particular dates (e.g., Boxing-day Sales, "Black Friday" or "Double 11 (Nov 11th)", in order to attract a large number of new buyers. Unfortunately, many of the attracted buyers are one-time deal hunters, and these promotions may have a little long-lasting impact on sales. To solve this problem, it is important for merchants to identify who can be converted into repeated buyers. By targeting these potential loyal customers, merchants can greatly reduce the promotion cost and enhance investment (ROI) return.

1.1.2 The restatement of the problem

The competition provides merchants and their corresponding new buyers acquired during the promotion on the "Double 11" day. Our task is to predict which new buyers for given merchants will become loyal customers in the future. In other words, we need to predict the probability that these new buyers would purchase items from the same merchants again within six months.

2. Sample and data source

Alibaba Cloud provides the data. The data set contains anonymized users' shopping logs in the past six months before and on the "Double 11" day, and the label information indicating whether they are repeated buyers. Due to privacy issues, data is sampled in a biased way, so the statistical result on this data set would deviate from the actual of Tmall.com. Nevertheless, it will not affect the applicability of the solution.

We have user behavior logs and user profiles as the two main raw tables for training. Field description and sample data are as follows:

2.1 User behavior logs

Data Fields	Definition
user_id	A unique id for the shopper.
merchant_id	A unique id for the merchant.
label	It is an enumerated type {0, 1}, where 1 means repeat buyer, 0 is for non-repeat buyer. This field is empty for test data.

Table 1 Definitions about User behavior variables

2.2 User Profile

Data Fields	Definition
user_id	A unique id for the shopper.
item_id	A unique id for the item.
cat_id	A unique id for the category that the item belongs to.
merchant_id	A unique id for the merchant.
brand_id	A unique id for the brand of the item.
time_tamp	Date the action took place (format: mmdd)
action_type	It is an enumerated type {0, 1, 2, 3}, where 0 is for click, 1 is for add-to-cart, 2 is for purchase and 3 is for add-to-favorite.

Table 2 Definitions about User profile variables

2.3 Training and Testing Data

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;0 and NULL for unknown.
gender	User's gender: 0 for female, 1 for male, 2 and NULL for unknown.

Table 3 Definitions about Training and Testing data

3. Data preparation

3.1 Challenges of processing big data

The first challenge we faced is the considerable scale of our data. Our training dataset contains 260864 users' data, including their profile information and log activities. We failed to read in data to pandas data frame directly. To solve this problem, we used pandas 'chunksize = 10000', which enable us to process the user_log and user_info data for further analysis.

3.2 Dealing with missing values

We start our data preparation process by first checking if there is any variable with missing values. We detected three categorical variables that have missing values – brand_id, age_range, and gender. Thus, we add a new category for missing values in each categorical variable (i.e., add gender=2 for missing gender)

3.3 Recoding categorical features

We recoded the categorical features into dummy features according to the definition provided by Alibaba Cloud.

1. For age_range

$$age = \begin{cases} 1 & for < 18 \\ 2 & for\ [18, 24] \\ 3 & for\ [25, 29] \\ 4 & for\ [30, 34] \\ 5 & for\ [35, 39] \\ 6 & for\ [40, 49] \\ 7 & for\ > 50 \\ 9 & for\ unknown \end{cases}$$

2. For gender

$$gender = \begin{cases} 0 & for\ female \\ 1 & for\ male \\ 2 & for\ unknown \end{cases}$$

3. For action_type

$$action = \begin{cases} 0 & click \\ 1 & add\ to\ cart \\ 2 & purchase \\ 3 & add\ to\ favorite \end{cases}$$

3.4 Feature engineering based on domain knowledge and EDA

We use features adapted from variables in 3 tables: user_info, user_log, and train to run our baseline model. Based on domain knowledge and EDA, we did a baseline feature engineering and added some new features:

3.4.1 Action-based features

	Feature name	Explanation
1	Click count	Total number of clicks the user/merchant/user-merchant have done

2	Add to cart count	Total number of add-to-cart the user/merchant/user-merchant have done
3	Purchase count	Total number of purchases the user/merchant/user-merchant have done
4	Add to favorite count	Total number of add-to-favorite the user/merchant/user-merchant have done
5	Action total count	Total number of actions the user/merchant/user-merchant have done
6	Purchase count rate $= \frac{\text{Purchase count}}{\text{Action total count}}$	Ratio of purchases the user/merchant/user-merchant have done

Table 4 Action-based features

3.4.2 Day-based features

	Feature name	Explanation
1	Click days	Total number of days the user/merchant/user-merchant clicks
2	Add to cart days	Total number of days the user/merchant/user-merchant add-to-cart
3	Purchase days	Total number of days the user/merchant/user-merchant purchases
4	Add to favorite days	Total number of days the user/merchant/user-merchant add-to-favorite
5	Action total days	Total number of days the user/merchant/user-merchant have actions
6	Purchase day rate $= \frac{\text{Purchase days}}{\text{Action total days}}$	Ratio of days the user/merchant/user-merchant have done purchase action

Table 5 Day-based features

3.4.3 Product diversity

	Feature name	Explanation
1	item count	Total number of items each seller/user/seller-user has

2	brand count	Total number of brands each seller/user/seller-user has
3	category count	Total number of categories each seller/user/seller-user has

Table 6 Product diversity features

3.4.4 User-merchant similarity

	Feature name	Explanation
1	Brand similarity score = seller_mkt_share_cat * NUB	User-merchant similarity on brand
2	Category similarity score = seller_mkt_share_cat * NUC	User-merchant similarity on category

Table 7 User-merchant similarity

3.4.5 Recent activities

	Feature name	Explanation
1	Double_11 actions count	The number of each action(click/add_to_cart/purchase/add_to_favorite) the user_seller have done during double_11
2	One_week_double_11 actions count	The number of each action(click/add_to_cart/purchase/add_to_favorite) the user_seller have done during one week before double_11
3	One_month_ double_11 actions count	The number of each action(click/add_to_cart/purchase/add_to_favorite) the user_seller have done during one month before double_11
4	double_11 actions rate	The rate of each action(click/add_to_cart/purchase/add_to_favorite) the user_seller have done during one week before double_11
5	One_week_ double_11 actions rate	The rate of each action(click/add_to_cart/purchase/add_to_favorite) the user_seller have done during one month before double_11
6	One_month_double_11 actions rate	The rate of each action(click/add_to_cart/purchase/add_to_favorite) the user_seller have done during one week before double_11

Table 8 Recent activities

Finally, we generated 81 new features and add them to our training and testing dataset, respectively.

4. Model training

4.1 Baseline model selection

To select a model that best captures the features, we deployed models widely used for classification. Since the logistic regression model has some limitations when we have many features (e.g., possible collinearity between features), we decided to employ a more advanced classifier model to train our data. Demonstrating their capacity for prediction many times, random forest, and XGBoost may improve our model performance. Therefore, we fit one random forest model, gradient boosting and one XGBoost model with default hyperparameter and baseline features, whose results are shown below:

4.1.2 Baseline model performance

	random forest	gradient boosting	XGBoost
Accuracy	0.938	0.938	0.938
AUC	0.627	0.663	0.669

Table 9 Baseline model performance

We have found that more than 90% of the labels in the training set is 0, which means the dataset was skewed significantly. This may cause the machine learning algorithms to predict almost every user to label 0, which makes them get very high accuracy but doesn't make any sense. To solve this problem, we chose several oversampling methods, like SMOTE and RUS, to balance the proportion of 0 and 1.

4.2 Model improvement with balanced data

	random forest	gradient boosting	XGBoost
Accuracy_SMOTE	0.937	0.937	0.938
AUC_SMOTE	0.616	0.602	0.625

Accuracy_RUS	0.637	0.644	0.647
AUC_RUS	0.647	0.655	0.659

Table 10 Improved model performance

From the training result of balanced data, we can notice that through XGBoost performs the best, the models' performance is very similar. Thus, we chose 5-fold cross-validation, and AUC scores to help our model selection process.

4.3 Imbalanced stratified k-fold cross-validation with different sampling methods

We can identify that a highly imbalanced outcome exists in our dataset from the initial Exploratory data analysis and model fitting. However, the traditional k-fold cross-validation algorithm would not work as it does not support applying different samplers. Therefore, we decide to implement a stratified k-fold cross-validation algorithm. We built a simple k-fold cross-validation algorithm that takes in arguments of the model, data input, data outcome(label), number of folds, scoring metrics, and sampling method. This algorithm supports all scoring metrics that are available in 'sklearn.metrics' package. Overall, this algorithm computes the cross-validation score while adjusting for the imbalance in the outcome.

Then, we implement this algorithm to compute the time, accuracy and AUC score using under-sampling methods such as RUS and over-sampling method such as SMOTE, ADASYN on different models (Random Forest, Gradient Boosting, XGBoost).

	Time	Accuracy	AUC	Sampler_method
Random Forest	923.635	0.938	0.619	SMOTE
Random Forest	53.142	0.637	0.653	RUS
Random Forest	1583.441	0.938	0.620	ADASYN
Gradient boosting	91.117	0.938	0.603	SMOTE
Gradient boosting	38.275	0.644	0.663	RUS
Gradient boosting	202.692	0.938	0.603	ADASYN
XGBoost	767.406	0.939	0.631	SMOTE
XGBoost	66.105	0.652	0.664	RUS
XGBoost	431.024	0.939	0.629	ADASYN

Table 11 k-fold result

5. Conclusion and Future improvement ideas

5.1 Result analysis and conclusion

From the table 11 above, we can easily notice a significant difference in time cost between different algorithm and sampler method. Given the same algorithm, the RUS method always runs the fastest while the ADASYN and SMOTE are more time-consuming.

A majority of methods show very high accuracy as well as the AUC score. Moreover, XGBoosting with RUS performs the best. It got the highest AUC score, 0.664. What is more, it merely cost 66.105 seconds to run, which means it is efficient.

5.2 Improvement ideas

Due to time and computational constraints, we did the best we can do. However, we do bear in mind that there is neither a perfect model nor an accurate prediction in the real world. After reflected upon our efforts, we had come up with some ideas for future improvement.

Some new features depicting user-merchant behavior are pretty sparse namely, there are too many zeroes in those features, which might be meaningless in the training process. Therefore, we could make some feature selection to preserve the feature with more information. Additionally, we can also do some re-feature engineering work to dig deeper to seek the lurking features we did not generate.

When things come to training algorithms, we have tried three crucial methods that are widely used in classification problems. However, there is some more complex method we did not have time to apply, especially the KNN and neural network.

6. Appendix

6.1 Competition performance

After submitting our prediction results to Alibaba Cloud on another new dataset, our AUC is 0.6709, ranked 242 on the leaderboard. There are a totally 3721 teams that join the competition.

We published our work on Github. The link is here:

https://github.com/frankcj6/Repeat_Buyers_Prediction_ML

6.2 References

[1] Guimei Liu, Tam T. Nguyen, Gang Zhao. Repeat Buyer Prediction for E-Commerce https://www.kdd.org/kdd2016/papers/files/adf0160-liuA.pdf

[2] Rahul Bhagat, Srevatsan Muralidharan. Buy It Again: Modeling Repeat Purchase Recommendations[J]. KDD 2018, August 19-23, 2018, London, United Kingdom https://assets.amazon.science/40/e5/89556a6341eaa3d7dacc074ff24d/buy-it-again-modeling-repeat-purchase-recommendations.pdf

[3] Huibing Zhang, Junchao Dong. Prediction of Repeat Customers on E-Commerce Platform Based on Blockchain[J]. Wireless Communications and Mobile Computing Volume 2020, Article ID 8841437, 15 pages https://www.hindawi.com/journals/wcmc/2020/8841437/

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. Journal of Machine Learning Research, 3(4-5):993–1022, 2003.

[5] L. Breiman. Random forests. Mach. Learn., 45(1):5–32, 2001.

[6] T. Chen and T. He. Xgboost: extreme gradient boosting. Available on https://github.com/dmlc/xgboost.

[7] M. Dash and H. Liu. Feature selection for classification. Intelligent data analysis, 1(1):131–156, 1997.