Repeat Buyer Prediction for online merchants

Team member: Jiaoyan Chen, Yujun Tian, Qinmei Wu

Background

Merchants sometimes run big promotions like discounts or cash coupons on particular dates. Black Friday is a well-known shopping day in the U.S. In China, November 11th ("Double11" day) has become the largest online shopping day during recent years, which is similar to Black Friday sale in the U.S. Unfortunately, many of the attracted buyers are one-time deal hunters, and these promotions may have little long lasting impact on sales. To alleviate this problem, it is important for merchants to identify who can be converted into repeated buyers. By targeting on these potential loyal customers, merchants can greatly reduce the promotion cost and enhance the return on investment. It is well known that in the field of online advertising, customer targeting is extremely challenging, especially for fresh buyers. For individual merchant, it is difficult to track the user behavior and define who is loyal customer due to the lack of log data.However, with the long-term user behavior log accumulated by Tmall.com, we may be able to solve this problem.

We will implement our idea on a set of merchants and their corresponding new buyers acquired during the promotion on the "Double 11" day of online merchant. We aim to predict which new buyers for given merchants will become loyal customers in the future. In other words, we will predict the probability that these new buyers would purchase items from the same merchants again within 6 months.

Project Overview

Data provided by T-mall contains anonymized users' shopping logs in the past 6 months before and on the "Double 11" day,and the label information indicating whether they are repeated buyers. There are 4 files containing the following features:
user_id, age_range, gender, merchant_id, label, item_id, category_id, brand_id, time_stamp, action_type.

Here is the table description in detail:

User Behaviour Logs

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-favourite.

User Profile

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.

Training and Testing Data

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.

We first cleaned the data and did feature engineering to generate more features and sparse the data. Then We applied the data to several models. When constructing models, we will use different gradient methods to guarantee lower

predicting error rate and better performance which including Random Forest, SVM, XGBoost and Logistic Regression.

We did our work in spark framework because it supports processing of Big Data on the clusters of commodity machines and machining learning library. It also supports python. We did feature engineering work on spark sql and data frame component and machine learning work on spark MLlib component. In addition, we use SQLite for faster join two data tables.

Data Preprocessing

1. Data Cleaning

We deleted all the data records in train.csv and test.csv which label is "-1", since we have known from the data description on website that label "-1" means 'user_id' is not a new customer of the given merchant. It means he/she is already defined as repeat buyer. For merchants, it is definitely they should treat their loyal buyers as the target customer on yearly sales day. Our goal is to predict whether a new customer will purchase from a merchant on 11.11.

2. Missing Values

We treated missing values differently according to their meaning. We deleted data records where missing values appear in the brand id, item id and category id. Maybe filling mode data is a better idea in theory, for example, we can fill in the missing item id with the item appears most frequently in that merchant. However, it takes a lot of time to compute the result and the data is sparse, we may not find such mode. So we just deleted such records. If missing values appear in gender, we filled with 2 to represent unknown gender. If missing values appear in age range, we filled with mode. With unknown time stamp, we deleted such records.

Different strategy when dealing with missing values

Features	**Solution**
item_id, brand_id, cat_id, timestamp	delete records
gender	replace with 2
age_range	replace with mode

3. Data Balance

After we applying our models on dataset first time, we found a problem of data balance. We found that there is too much 0 comparing with 1. By counting our original dataset, we can see that we have about two hundred fifty thousand 1 but around twenty thousand 0.

Because the combination of the true samples and false samples are unbalanced, we decided to apply some method to solve this problem. What we've done called oversampling and downsampling. About oversampling, we tried to connect two sample points in the same category, then selected random points on the line to generate 0 labeled samples. SMOTE method were also used to get more samples.

About undersampling, we used boosting method which could randomly select balanced samples from training set to form weak classifiers. These weak classifiers then could form the final classifier.

Feature Engineering

The origin training data and test data contain only user id and merchant id. It is not sufficient to build an accurate model for machine learning algorithms. So we must generate more features to expand the origin table. First we merged features in user log and user information with training set according to user_id and merchant_id. Then we expanded features based on these given ones.

1.Aggregation

We can suppose that if an item has been purchased or favourited many times in a month, it is more likely to attract customers to buyer it again on 11.11. It is the same when we considering brand, category and merchant. There are four different action types in origin data: click, add to favourite, add to cart, purchase. We aggregation these different actions on brand /item /category /merchant separately to get new features. Similarly, if a user is an active customer who click or purchase many items is past months, we can assume that he/she may be a repeat buyer on 11.11. So we also generate a new feature aggregation on user monthly actions for all items.

Product diversity is also an important feature when we evaluate whether a merchant can attract repeat buyers. It is defined as the number of unique items, brands, or categories in a merchant that clicked, purchased, added to cart or added to favourite by a certain user. The intuition behind this feature is that if a user is interested in more goods of a merchant, he/she is more likely to buy again from the merchant.

We can also generate a feature similar to the above one. If a certain item, brand and category of a merchant is clicked, purchased, added to cart or added to favourite by more unique users, it is more likely to attract a new customer to buy it again because it is very popular among others.

User aggregation feature of a merchant is defined as the number of users who bought on at least two different days of items, brands, categories or merchants. The intuition behind user aggregation feature is if a user purchases from a merchant more than once on average, other user may also come back. As more users come back, new customers may come back on 11.11 as well.

To sum up all features generated in this part, a tabel is provided below:

Aggregation features	1. Monthly click/add/favorite/purchase count for certain brand/item/category/ merchant.. 2. User's monthly action count of click/add/favorite/purchase for all items. 3.Total number of unique items/brand/category for a merchant actioned by a certain user. 4. The number of unique user who clicked/purchased/added/favourites for a given items/brands/categories/merchant 5. The number of users who bought on at least two different days from a items/brands/categories/merchant.

2.User behavior

Double 11 feature is the counts of a user clicks, purchase, add to favourite and add to cart on last double 11 day. If a user is very active on last double 11, we may infer that he/she may also become an active customer on next double 11.

The more actions he/she performs, the more likely he/she becomes a repeat buyer.

Days counts are the number of days with a particular action type in each month for a certain user. Imaging there are two users who buys something in different days and who buys everything in one day. It is obvious the former one who may more possible to buy again from merchants.

User activity is defined as the number of unique items, brands, categories or merchants that a certain user clicks, adds to cart, adds to favourite or purchases in each month.

To sum up all features generated in this part, a tabel is provided below:

User Behavior	1.Counts of clicked/purchased/added/favorited of items for a certain user on the last Double 11 day. 2.Days of a certains user clicked/added/favorited/purchased items in each month. 3.The number of unique items/brands/categories/merchant that the user clicked/purchased/added/favourites in each month

3. Normalization

The features we generated above have different scales. For example, Double 11 feature which means the counts of a user clicks, purchase, add to favourite and add to cart on last double 11 day and Product diversity. A merchant may contains thousands of items, but a user may only purchase several items on last 11.11. To avoid skewness in parameter, we normalized all the features to the range of 0~1.

Modeling

1.Logistic regression

Binary Logistic Regression is a special type of regression where binary response variable is related to a set of explanatory variables, which can be discrete and/or continuous. In logistic regression Probability or Odds of the response taking a particular value is modeled based on combination of values taken by the

predictors. Like regression, we make an explicit distinction between a response variable and one or more predictor (explanatory) variables.

To implement logistic regression we applied SVMWithSGD model in MLlib and we random split normalized data into training set(70%) and test set(30%). During validation, we compared two indicators test error for all customers and test error for repeated customers only. For the first one, we divided the amount of total customers with the number of wrong prediction. we found the percentage is approximately 9%. The second one is error percentage for repeated customers, which equals to the number of false prediction on repeated customer divided the total amount of actual repeated customers.

Moreover, we also calculate area under ROC and area under PR. Based on the indicators we calculate, we can see that logistic regression has a really good performance on classification of customers. The error rate is pretty low and AUC is over 0.9.

	Training Error
All customer	**~9.33%**
Repeated customer	**~9.67%**

	Area under curve
ROC	**~90.67%**
PR	**~93.05%**

2.Random Forest

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Using a random selection of features to split each node yields error rates that compare favorably

to Adaboost, but are more robust with respect to noise. Internal estimates monitor error, strength, and correlation and these are used to show the response to increasing the number of features used in the splitting. Internal estimates are also used to measure variable importance.

	Training Error
All customer	**~7.83%**
Repeated customer	**~8.72%**

	Area under curve
ROC	**~92.31%**
PR	**~93.78%**

Similar from the previous model, we still use the test error for all customers and repeated customers as our indicators. The Random Forest model achieves a lower test error in both indicators which are 7.83% in all customers and 8.72% in repeated customers. In the area under ROC and PR curve part, our Random Forest algorithm also achieves a better performance than the previous logistic regression model. The Random Forest has 92.31% in area under ROC curve and 93.78% under PR curve.

3.XGBoost

XGBoost is short for “Extreme Gradient Boosting”. At first, we believe XGBoost can give us a better result when comparing other methods, since Logistic regression and SVM are linear classification, but XGBoost offers both CART and gblinear. Comparing with random forest, XGBoost is much quicker in operation speed. Besides that, we think it is pretty useful for data scientists since it can support three main forms of gradient boosting: Gradient Boosting, Stochastic Gradient Boosting and Regularized Gradient Boosting. Therefore, XGBoost includes L1 and L2 regularization terms, which can provide the number of leaf nodes of the tree, and the sum of the squares of the L2 modules of the score output on each leaf node. Considering Bias-variance tradeoff, the

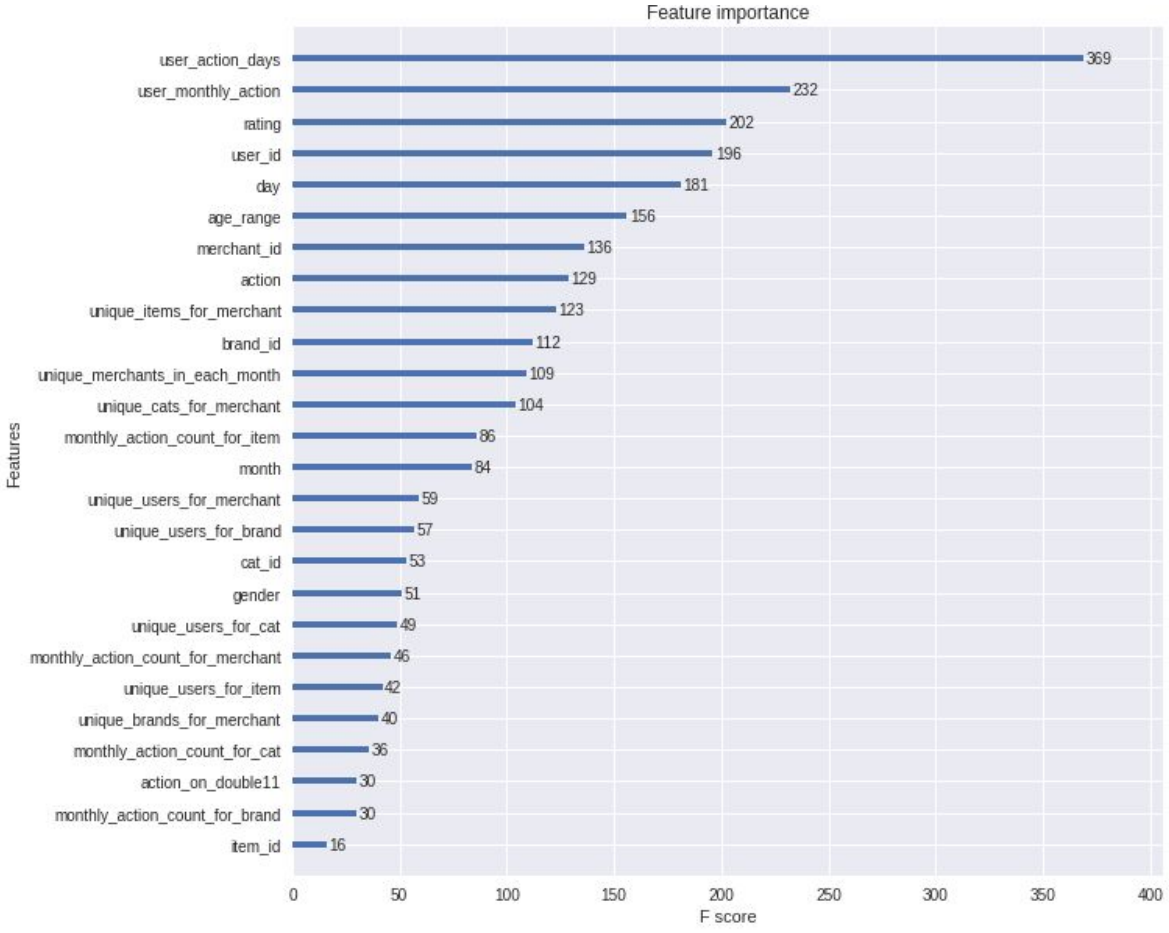
regular term reduces the variance of the model, making the learning model more simple and preventing over-fitting problem.

XGBoost is easy to get hands on, and provide strong parameter system to reach to more ideal result. The parameter mainly cover the three aspects:

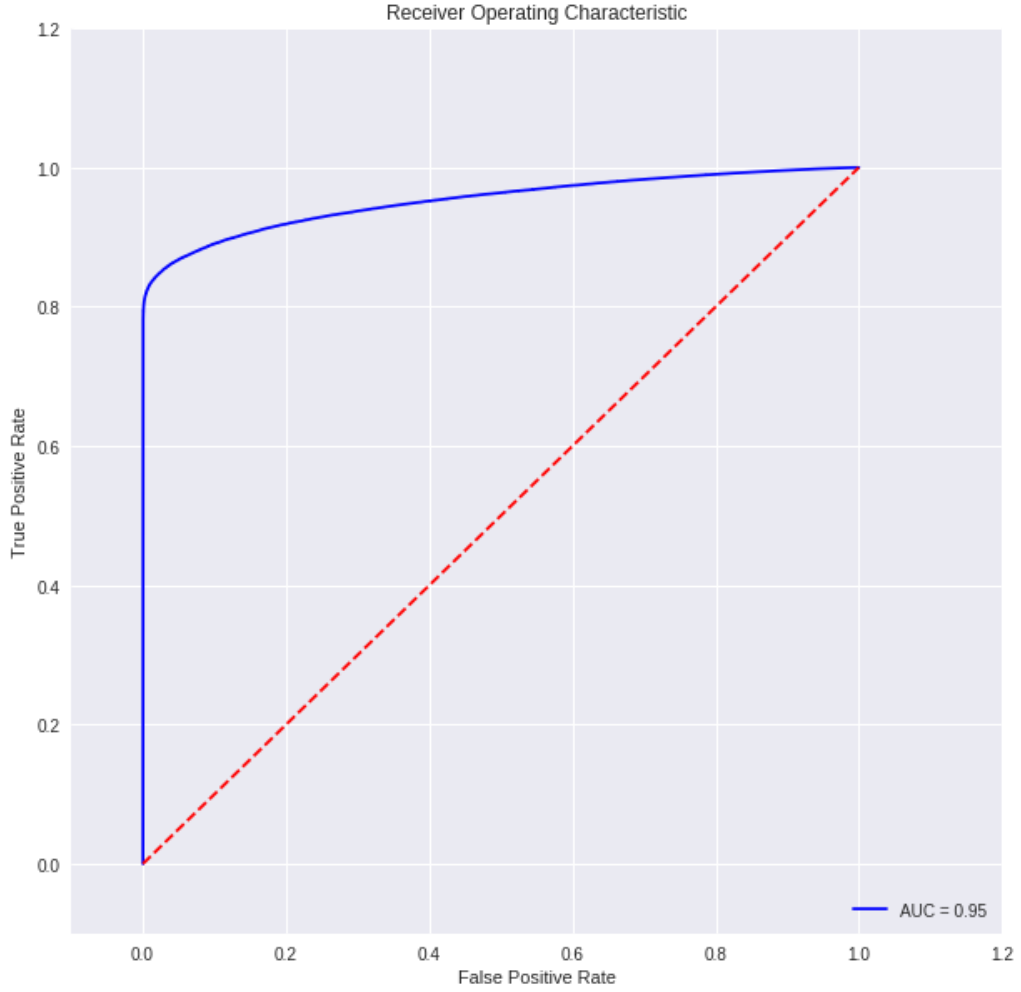
1. *General Parameters: Guide the overall functioning*
2. *Booster Parameters: Guide the individual booster (tree/regression) at each step*
3. *Learning Task Parameters: Guide the optimization performed*

To implement XGBoost in spark, I used scala language to implement the algorithm, for the dataset, I used processed data after adding all the features and having been balanced, all the data with label of "-1" having been deleted and NA value appropriately processed.

After adding many parameters, I took a look at the importance of all the features by XGBoost, as a result, the feature important ranking is:



To achieve ideal result of the dataset, I used different combinations of parameter. When “eta” = 0.1, “min_child_weight” =5, “max_depth” = 5, “silent” = 1, “lambda” = 1, “nthread” = 1, I can achieve the best result. At this time, the test error is 94.14%. And the AUC is 0.95.



The confusion metric is as following:

		Truth data			
		Class 1	Class 2	Classification overall	Producer Accuracy (Precision)
Classifier results	Class 1	348794	2679	351473	99.238%
	Class 2	38556	0313307	351863	89.042%
	Truth overall	387350	315986	703336	
	User Accuracy (Recall)	90.046%	99.152%		

To help clarify the meaning of confusion metrics, I used the online calculator and find that true positive is 90.046%, which is relatively accurate. And personally thinking, TP is much more important in that every company or merchant will do their best to capture all the potential repeating buyers. The true negative(TN) rate is not as good as TP. If the cost of merchant or company to pay for attracting potential repeating buyers is high, then TN rate is important.

Conclusion

In this project, we generated multiple features to expand the training data and test data to improve the accuracy of repeat buyer prediction. We trained our data on four different models: Random forest, Logistic regression and XGBoost. Our result shows that after expanding the origin features, the accuracy has improved greatly. Among these four models, XGBoost achieves the best performance when predict repeating buyers which test accuracy is up to 94.14%. Maybe we can apply this model on T-mall platform in the furture.

Reference

data set:
https://tianchi.aliyun.com/datalab/dataSet.htm?spm=5176.100073.888.13.6lOUPw&id=1
spark MLlib
https://spark.apache.org/docs/latest/mllib-guide.html
spark sql and dataframe:
https://spark.apache.org/docs/latest/sql-programming-guide.html
XGBoost:
http://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf
https://github.com/dmlc/xgboost
http://xgboost.readthedocs.io/en/latest/model.html
Logistic regression
https://en.wikipedia.org/wiki/Logistic_regression
Random Forest
https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.html
SQLite:
https://www.sqlite.org/docs.html