Repeat Buyers Prediction - Challenge the Baseline

Project Report for CS150: Database and Datamining

Xiaoye MO
SIST, ShanghaiTech University
68488464
Shanghai, China
moxy@shanghaitech.edu.cn

Zhenyang LI
SIST, ShanghaiTech University
56290897
Shanghai, China
lizhy1@shanghaitech.edu.cn

ABSTRACT

In the course project of CS150:Database and Datamining, we two are required to solve a popular e-commerce problem of predicting the probability of repeated buyers for the merchants. The ROI has become the focus for the merchants in the e-commerce. We have given our own solution with machine learning and dataming skill with data preprocessing, feature engineering, normalization, training, classification and stacking. The report will show the methods as well as the experiments we had faced and settled.

KEYWORDS

Repeat Buyers Prediction, E-commerce, Datamining, Machine Learning, Feature Engineering, Data Training, Classification

ACM Reference Format:
Xiaoye MO and Zhenyang LI. 2018. Repeat Buyers Prediction - Challenge the Baseline: Project Report for CS150: Database and Datamining. In *Proceedings of (Project Report for CS150, SIST)*, Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.475/123_4

Figure 1: Double 11

Figure 2: Black Friday

1 INTRODUCTION

Nowadays, it has come into the age of e-commerce. There has been going through a dramatic growth of the market size as well as the inflow of capital and funds. Merchants flooded into the market searching for the golden opportunities, expanding the business scale and try to make a big fortune.

Whatever the merchants are, they are facing with the questions of customers' selection: who are the potential customer to attract? who are those fresh new buyers and who are those loyal and key buyers. The merchants will take different strategies to buyers.

For the purpose of attracting new buyer, 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 little long lasting impact on sales.
To alleviate this problem, *i.e.* to justify the type of the buyers, 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 (ROI).

It is well known that in the field of online advertising, customer targeting is extremely challenging, especially for fresh buyers. However, with the long-term user behavior log accumulated by Tmall.com, it becomes possible for to solve this problem. In this challenge, we are provided with a set of merchants and their corresponding new buyers acquired during the promotion on the "Double 11" day with the goal of predicting which new buyers for given merchants will become loyal customers in the future.

Thanks to the data set with around 200k users provided by Tianchi, we are able to train, test and predict the probability that these new buyers would purchase items from the same merchants again within 6 months.

Our final solution comes out as a comprehensive feature engineering and model training. Skill like feature engineering has been applied and several classification models have been tested to obtain the best result. Detailed procedures will be discussed later in the paper with section 2 as Methods applied, section3 as challenges and experiments we have faced and deal with and section 4 as conclusion part.

2 METHODS

2.1 Data Preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

2.1.1 Missing Value Filling. The Imputer class provides basic strategies for imputing missing values, either using the mean, the median or the most frequent value of the row or column in which the missing values are located. This class also allows for different missing values encodings. By using *isnull*() function, we can identify the missing vale while imputer may work for the imputation of missing value. For instance, we have found such missing in the gender, such strategy is implemented:

```
from sklearn.preprocessing import Imputer
imp = Imputer(missing_values=0, strategy='mode', axis=0)
>>> imp.fit(X)
```

2.1.2 Encoding. We use OnehotEncoder to encode categorical variable. One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. For example, [house, car, tooth, car] becomes [0,1,2,1] after the encoding.

In the project, we encode all the feature that could not verify as digits. The following example will show how we encode the two different action types (*user_id* and *action_type*):

```
from sklearn.preprocessing import label_binarize
u_actionsType = userLog[['user_id','action_type']].copy()
action_type_hot = label_binarize(np.array
(u_actionsType.action_type),classes=[0, 1, 2, 3])
u_actionsType['u_action_type_hot_0'] =
action_type_hot[:,0]
u_actionsType['u_action_type_hot_1'] =
action_type_hot[:,1]
u_actionsType['u_action_type_hot_2'] =
action_type_hot[:,2]
u_actionsType['u_action_type_hot_3'] =
action_type_hot[:,3]
```

2.2 Feature Engineering

Feature engineering, an integral part of data science, is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive.

A feature is an attribute or property shared by all of the independent units on which analysis or prediction is to be done. Any attribute could be a feature, as long as it is useful to the model.The purpose of a feature, other than being an attribute, would be much easier to understand in the context of a problem. A feature is a characteristic that might help when solving the problem.The quality and quantity of the features will have great influence on whether the model is good or not.

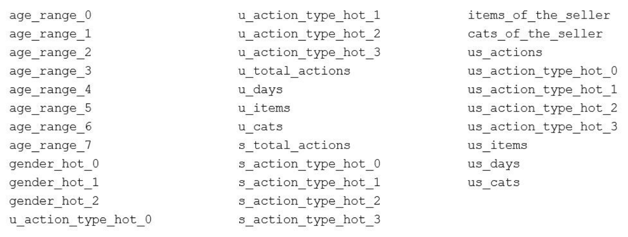
In the project, the user activity log data contain 5 entities: users, merchants, brands, categories and items. The characteristics of these entities and their interactions can be predictive of the class labels. For example, users are more likely to buy again from a merchant selling snacks than from a merchant selling products within six months, since snacks are cheaper and are consumed much faster than electronic products. We generated some features to describe the characteristics of the ïňĄve types of entities and their interactions.

The features we generated range from basic counts to complex features like similarity scores, trends, PCA (Principal Component Analysis) and LDA (Latent Dirichlet allocation) features. All the features of an entity form the proïňĄle of the entity.

For each entity, we have selected the feature related to each other, put them together. Take the following as an example, we select *user_id* and *item_id* as two features, then create the table of *u_items*:

```
u_items = userLog[['user_id','item_id']].copy()
u_items.drop_duplicates(inplace=True)
u_items.item_id = 1
u_items = u_items.groupby(['user_id']).agg('sum').
reset_index()
u_items = u_items.rename(index=str,
columns={'item_id':'u_items'})
```

Then, we have settled the data into the forms with merging as *userFeatures.csv* and *sellerFeatures.csv* and the final training dataset *t_dataset.csv* with the size of 260864×34 has been finally generated.



age_range_0	u_action_type_hot_1	items_of_the_seller
age_range_1	u_action_type_hot_2	cats_of_the_seller
age_range_2	u_action_type_hot_3	us_actions
age_range_3	u_total_actions	us_action_type_hot_0
age_range_4	u_days	us_action_type_hot_1
age_range_5	u_items	us_action_type_hot_2
age_range_6	u_cats	us_action_type_hot_3
age_range_7	s_total_actions	us_items
gender_hot_0	s_action_type_hot_0	us_days
gender_hot_1	s_action_type_hot_1	us_cats
gender_hot_2	s_action_type_hot_2	
u_action_type_hot_0	s_action_type_hot_3	

Figure 3: All Feature Used

2.3 Normalization

Database normalization is the process of restructuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.

Particularly speaking, normalization is the process of scaling individual samples to have unit norm. The process can be useful for using a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.

As a norm is a function that assigns a strictly positive length or size to each vector in a vector spaceâĂŤsave for the zero vector, which is assigned a length of zero. There are two different type of norms for frequent use:

$$L_1 Norm : ||x||_1 = \sum_i |x_i|$$

$$L_2 Norm : ||x||_2 = \sqrt{\sum_i |x_i|^2}$$

```
normalizer = preprocessing.Normalizer().fit(X)
Normalizer(copy=True, norm='l2')
```

In the project, we have implemented a scalar into $[-1, 1]$ for the data usage, i.e.

```
X_challenge_scaled = preprocessing.scale(X_challenge)
X_challenge.age_range_0 = X_challenge_scaled[:,0]
# instance for the scale process
```

2.4 Training

With package in scikit learn, it becomes easy to split arrays into random train and test subsets.

It will fulfill the quick utility that wraps input validation and $next(iter(ShuffleSplit(n_samples)))$ and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

```
from sklearn.model_selection import train_test_split
X = t_dataset.iloc[:,0:-1]
Y = t_dataset.iloc[:,-1]
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,
test_size=0.3,random_state=100)
```

2.5 Classification

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data.

K-fold cross-validation will provide tune parameters for each individual classifier. There are several classifiers to choose:

2.5.1 K-Neighbors Classifier. In pattern recognition, the k-nearest neighbors algorithm ($k - NN$) is a non-parametric method used for classification and regression.In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether $k - NN$ is used for classification or regression: either to be a class membership or the property value for the object. In which nearest neighbor is computed on the basis of estimation of k. That indicates how many nearest neighbors are to be considered to characterize. It makes use of the more than one closest neighbor to determine the class. In which the given data point belongs to and so it is called as $K - NN$. These data samples are needed to be in the memory at the runtime.

Algorithm 1 K-Neighbors Classifier Pseudo Code

```
$K \leftarrow number\ of\ nearest\ neighbors$
**for** each object $x_{in}$ in the test set **do**
    calculate the distance $D(X, Y)$ between $X$ and every $neighborhood$ gets the $k$ neighbors in the training set closest to $X$
    $X.class \leftarrow$ **function** SELECTCLASS($neighborhood$)
**end for**
```

2.5.2 XGB Classifier. Gradient Boosting for classification.GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage $n\_classes$ regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Binary classification is a special case where only a single regression tree is induced.

XGBoost comes as the implementation of gradient boosting decision tree, which has a faster computing speed and better performance. It is easier for Parallelization, Distributed Computing, Out-of-Core Computing and Cache Optimization of data structures and algorithms.

2.5.3 GridSearchCV Classifier. GridSearchCV implements a "fit" and a "score" method. It also implements "predict", "predict probability", "decision function", "transform" and "inverse transform" if they are implemented in the estimator used. The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid.

2.5.4 RandomForest Classifier. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, ..., x_n$ with responses $Y = y_1, ..., y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1, ..., B$:

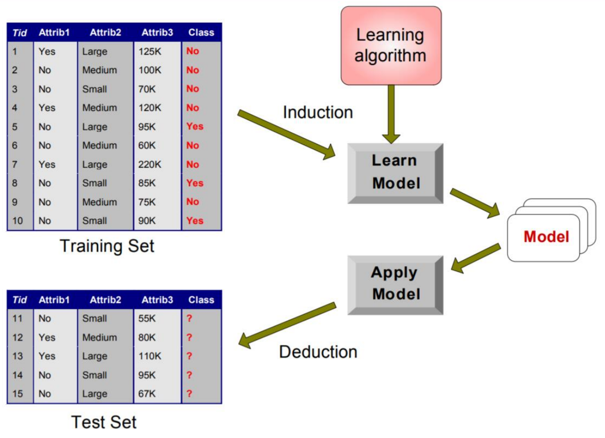
Sample, with replacement, n training examples from X, Y; call these X_b, Y_b. Train a classification or regression tree fb on Xb, Yb.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x:

$$f = \frac{1}{B} \sum_{b=1}^{B} f_b(x')$$

A random forest use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features as "feature bagging". It is also a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Model.jpg



Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Figure 4: Classification Model Sample

2.6 Normalization

2.7 Stacking

Stacking is a ensemble of models combined sequentially.It uses a "meta learner" instead of voting to combine the predictions of "base learners."The base learners (the expert) are not combined by voting but by using a meta-learner, another learner scheme that combines the output of the base learners.

There are two levels of 0 and 1 model for prediction:

For example ,Base learners: level-0 model; Meta Learner: level-1 model.

The predictions of the base learners are input for the meta-learner.

3 EXPERIMENTS

3.1 Data Preprocessing

The data without preprocessing has brought us with the difficulties of not able to select the features well-off. However, by fill the missing value, drop duplicates, encoding and testing with *isnull*(). The preprocessing goes smoothly and helps us to select the feature we experienced.

In this project, some missing data are the user's age range and gender. We fill the missing data with symbols which are different from the origin data. For gender, we fill the missing data with number 2. For age range, we fill the missing data with number 0.

Furthermore, we handled some special data in age range. We treat 8 as 7, because the amount of 8 and 7 data is small, we did this for better performance.

Part of the code is shown below:

```
userInfo.age_range.fillna(0, inplace = True)
userInfo.age_range.replace(8, 7, inplace = True)
userInfo.gender.fillna(2, inplace = True)
```

3.2 Feature Engineering with scalar

At first, we use about 10 features, but we saw the result is not good enough, then we tried to add some features. So we have once trained the 30+ features with their origin values, as the trained data are too large, which perform terrible as the result, whatever the classifiers selected.

However, with certain feature scaled into the range of $[-1, 1]$ as a normal distribution, the prediction gets proper functioning.

3.3 Classifier Selecting

We have chosen several different kinds of classifier: K-neighbors, XGB, GridSearchCV, GradientBoosting and RandomForest classifiers. Once the scalar has been completed, the features selected and the data itself has showed the greatest practicability of Random Forest, which come as our final selection.

For XGB classifier, we found it over-fits the training data easily, even using K-Fold algorithms, the result can also over-fit the training data. So for prediction, the result is bad. One thing you can try is to change the parameters when training, which can help to reduce the over-fitting problem.

When using GridSearchCV, our result becomes better but not as higher as using RandomForest. But in our opinion, it could be better than RandomForest's result, if setting better parameters in the function, the result could be better.

For RandomForest, maybe the random factor are helping us, our result seems pretty well among above classifiers. So for the final submission, we use the result of RandomForest.

4 CONCLUSION

All through the project, we have implemented the machine learning and data mining concepts and knowledge from the course into the actual exercises to the project of repeated buyer prediction. By following the project guidance, we have gone through the procedures of the project step by step. We have selected an appropriate number of features to get the accurate prediction and get the result of 0.65803673. We hope what we do will offer the proper solution for the popular e-commerce problem. The projects thus adds to our programming skills with the python package of Pandas, numpy and sklearn, helps us to build the thought and train the practical skill for them. Thanks for the instructors and the TAs for the patient guidance.

By the way, our group has the workload of 50% **to** 50%**.**

REFERENCES

[1] Gumeri Liu, Tam T.Nguyen, Gang Zhao(2016). Repeat Buyer Prediction for E-Commerce.