Predicting Repeated Buyers: Final Report

Ishmail Grady, isg9
Hao Rong, hr335
Frank Zhang, fz252

December 2017

1 Exploratory Data Analysis

1.1 Problem Specification

In a world where amount of revenue generated by retailers is steadily becoming more focused on online sales, improving online marketing strategies can give a retailer a huge competitive advantage. Our project aims to predict the retention of new customers which can be very useful information to improve the online marketing strategy for our online stores. Specifically, our goal is to predict if a new customer of a given merchant will be a repeated buyer of that merchant. The major use of this project is providing insight on which customers should be sent promotions to. For instance, if we are able to predict that a customer will be a repeated buyer for a given merchant, then advertising products from that merchant to the customer can result in more sales in the future.

1.2 Data Cleaning and Statistics

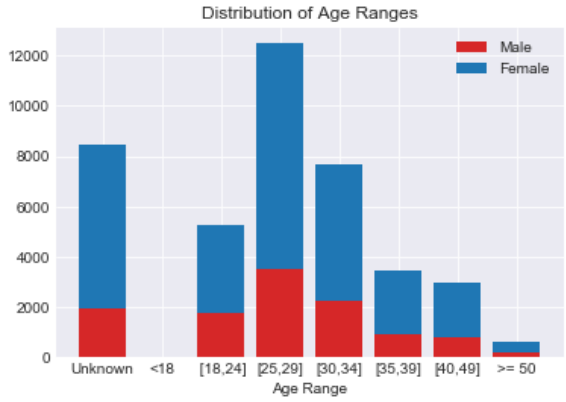
The data set used in this project is from Tmall.com, a Chinese online retailer. The data set is a collection of information about users who used the site in 6 months. The data can be divided into two categories: user behaviour logs and user profile information. The user behavior logs include each interaction a user has with a merchant, with the following information: item ID, category ID (of item), time of action, brand ID, and action type (add-to-cart, add-to-favorite, and purchase). The user profile information contains the gender and age of each user.

The raw data set contains 260,864 users/observations including users who interacted with all 4995 merchants on the platform. However, the amount of unique users across all of the merchants on the platform follow a power-law like distribution, thus many of the merchants included in the data set have very few users to form predictions with. For the purposes of our project we will restrict our data set to users that interacted with merchants that have at least 1000 unique users. Our final data include 44834 users divided among 24 different merchants.

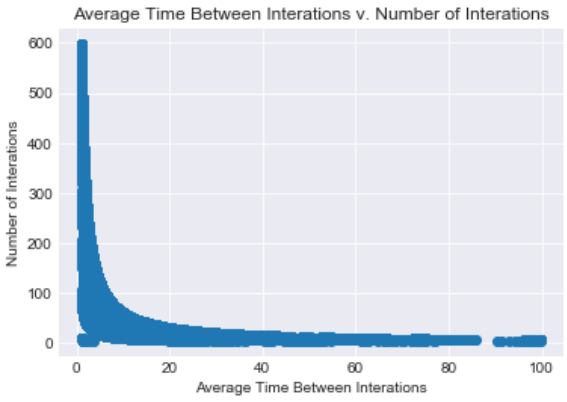
1.3 Data Cleaning and Statistics

The raw data set contains 260,864 users/observations including users who interacted with all 4995 merchants on the platform. However, the amount of unique users across all of the merchants on the platform follow a power-law like distribution, thus many of the merchants included in the data set have very few users to form predictions with. For the purposes of our project we will restrict our data set to users that interacted with merchants that have at least 1000 unique users. Our final data include 44834 users divided among 29 different merchants.

Whenever trying to predict outcomes related to human behavior, demographic information can be very useful. As shown in Figure 1, the mode of the age ranges is [25,29]. Users in this



(a) Figure 1



(b) Figure 2

Figure 1

age range make up about 29% of the repeated buyers in the data set, which is a disproportionate amount. From this figure we can also see that female users represent the majority of the users and the repeated buyers, about 70%. this indicates that these gender and age information could help predict our outcome.

The average amount of interactions per user is about 18 and has a median of 6. There is a relatively large amount of users with low amounts of interactions, but the data is skewed by users who have very large amount of interactions. Action type 1, clicking on an item, is make up most of these interactions, about 89%.

Another interesting relationship to investigate in this problem is the relationship between the average in between interactions and the total amount of interactions to examine how frequency affects user interaction. Figure 2 shows that these two variables have an approximately exponential relationship. Thus, according to our data, users that interact with the site a high rates are very unlikely to use it very frequently. this is important to not because one would expect that the more times a user interacts with a site, the higher the chance of them purchasing an item.

2 Feature Selection

2.1 Feature Selection for Baseline Model

In order to get a baseline prediction for our final problem, which is predicting a user will be a repeated buyer for a particular merchant, we first tackle the problem of predicting whether a user will be a repeated buyer on the Tmall.com platform. The final product of this project will be one model each merchant. This baseline model will give us a means to compare the performance of our final models. The features used in the baseline model are as follows:

Feature	Notation
Age_range	Age range of the user
Gender	Gender of the user
Action_0	Times the user clicked any item
Action_1	Times the user added any item to the chart
Action_2	Times the user purchased any item
Action_3	Times the user added any item as favorite
avg_time	The average time range between two actions
num_item	Number of items the user acted upon
num_cat	Number of categories the user acted upon

2.2 Features for Merchant Specific Model

In order to obtain more accurate predictions for one particular merchant, we trained specific model for each store, which renders us the ability to produce more store-specific features for prediction-enhancement. For example, we generated features for each category and the interactions one particular user had within a store. Since each store has multiple categories, the matrix X is extremely sparse: for merchant number 1892, there are 1486 users and 474 features. So the matrix has a dimension of 1892*474.

Such a sparse matrix led us to choosing Low Rank Model as the first candidate, because our predicting repeated user for online store is a classic problem that could be resolved by the Low Rank Model which will be discussed later. In addition, we also implemented the decision tree method covered in the lectures since we have mixed data types (categorical and numerical). Although decision tree results are easy to interpret, one should keep in mind that the decision tree method usually yields lower prediction accuracy and unstable, subject to little perturbations.

2.3 Baseline Model

Since we are looking for a general model as baseline to evaluate our more detailed model for each merchant, we implemented the perceptron algorithm for classification at stage. For the perceptron task, repeated buyers are labeled as 1 and non-repeated buyers are labeled -1. We withheld 20% of the total dataset as test data and applied the perceptron algorithm to the 80% training data. The test data and train data contain 41775 and 167100 examples. We iterated the perceptron algorithm 100 times and updated weights (w) each time when the predicted value is different from the correct answer. The perceptron model worked out fairly well and is consistent for both in sample and out of sample data, and achieved accuracy of 92.7% and 92.8%, respectively.

2.4 Merchant Specific Model using Logistic Classification

Another algorithm we select is the logistic regression model which is used when the predicted variable is categorical. The loss function of logistic regression is shown below. It could be understood as finding the parameters that best fit the equation below.

Loss Function of Logistic classification	Parameters β fit the equation
$J(w) = \sum_{i=1}^{m} y^{(i)} \log P(y=1) + (1-y^{(i)}) \log P(y=0)$	$y = \begin{cases} 1 & \beta_0 + \beta_1 x + \varepsilon > 0 \\ 0 & \text{else} \end{cases}$

We first perform Logistic classification without regularization on the original features in the example store. However, due to sparsity and linear dependency in columns, Logistic classification fail to converge. So we need to add regularizer to converge the algorithm. Moreover, we can conduct features selection or alleviate outlier effect by adding different regularizer into the model. To build a measurement in choosing the model based on the in-sample and the out-of-sample error, we randomly split the data into 60% training set, 20% cross validation set, and 20% test set. For each regularizer we will use validation set to choose the best that minimize the validation misclassification. Then we use the the best to construct the model and predict on the test set to get the test error.

2.5 Logistic Classification with Lasso Regularizer

Since our data is sparse with many zero entries in most of the features we use above and we have about 80 features in each store prediction model, we might want to conduct feature selection by adding Lasso regularizer to the model. The intuition behind this is to see if the model can perform better with selected variables. The objective function is defined as:

$$minimize\ h(x) = log(1 + e^{(-yw^T x)}) + \lambda \|w\|_1$$

However, by adding Lasso Regularizer, it shrinks the coefficient of all the original features into zero through feature selection except the coefficient of the intercept, which makes sense because any one of the original too sparse features (namely gender, age group, number of different interaction type on specific category of the goods) make little tribute in deciding if the user is a repeated buyer or not.

2.6 Logistic classification with Ridge Regularizer

To be able to converge the model, we also want to add Ridge regularizer into the model. Since Ridge regularizer can only reduce the model coefficient to nearly zero, we can also avoid the problem we have with the Lasso Regularizer. The objective function is defined as:

$$minimize\ h(x) = log(1 + e^{(-yw^T x)}) + \lambda \Sigma w^2$$

In each iteration of random splitting, we first use cross validation to select the best lambda that yield best accuracy, then use it to construct the regularized model. Finally, based on the accuracy rate in the confusion matrix, we decide the threshold of each Logistic classification model. After performing 20 iteration of error estimation, the average in-sample error is 8%. The average out-of-sample error is 14.1%. Since plotting residual vs. fitted value is not very clear visualization in classification problem, a confusion matrix of example model is shown below:

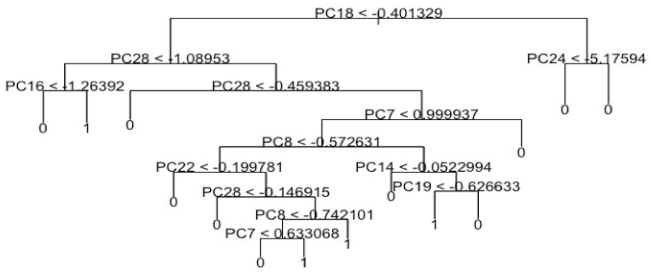
	Predicted 0	Predicted 1
True 0 (Not Repeated)	1316	105
True 1 (Repeated)	45	20

2.7 Tree-based methods with PCA

Recursive partitioning in Tree-based methods can helps us produces easy to visualize prediction of a returned buyers. However, with large sparsity and linear dependency within the customer-store interaction records data, we get a good result from performing the tree methods directly with dimension reduction. Principal component analysis (PCA) can helps us deal with the issue of sparsity and dependency in the data since it perform an orthogonal transformation to convert the dataset with possibly correlated variables into a set of principal components which are values of linearly uncorrelated variables. Although conducting PCA inhibit the interpretability of the features, it does not obstacle our goal of prediction. Before performing PCA, we need to first removed the constant columns which in our case are the columns with all zero entry. Then we get the result of a set of PCs(Principal components) each with standard deviation, proportion of variance and cumulative proportion. We can choose number of PCs by setting a threshold or choosing by cross validation process. In our case, we use a threshold of 0.30 in standard deviation which is a rule of thumb.

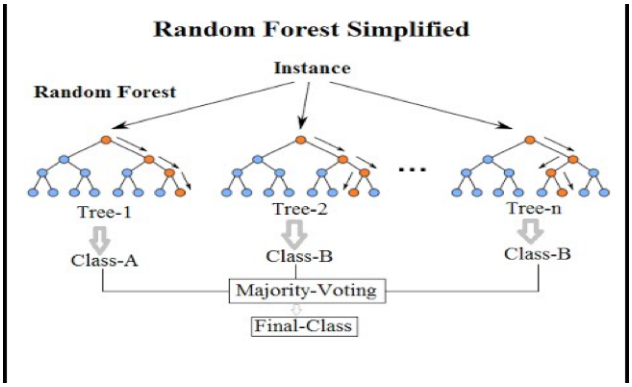
2.8 Decision Tree using Principal Component

Decision tree can separate data recursively to find the best split in the sample space. With principal component, it choose one PC at a time to split and separate the user_label of repeated buyer or not. The tree-pruning technique we use is reduced error pruning where each tree node is replaced with its most popular class to see if the prediction accuracy is affected. A result of the example store is shown below. After performing 20 iteration of error estimation, the average in-sample error is 6%. The average out-of-sample error is 13%.



2.9 Random Forests using Principal Component

Another tree-based method we use is random forests. It is an ensemble learning method for classification, regression and predictions. The mechanism behind random forests is first constructing a various of decision trees using training set where each tree uses a set of features that are randomly drawn from the entire feature set. Then, when making the prediction, it aggregate the results that each subtree made through ensemble methods. For classification question, it performs majority votes, which is select the mode prediction among set of results. An example diagram is shown below.



2.10 Random Forests using Principal Component

For our random forests model, we generate 500 decision tree with 5 variables used to predict in each subtree model on the training set. Then we use test set to measure the model performance. After performing 20 iteration of error estimation, the average in-sample error is 0.4%. The average out-of-sample error is 7.4%. The result is much better than the decision tree. This means we might overfit a little. But because out-of-sample error is not too high, we think by increasing the size of the training set in the future when a store gather more and more data, the prediction of random forests model will become better.

3 Low Rank Models

The data sets constructed for each merchant are very sparse. Let ℓ_p be the total number of item categories for the pth merchant. $k \in \mathbf{C} \subset \{1, ..., \ell_p\}$ be the k^{th} category of of the p^{th} merchant. This is due to the fact that a given user has only interacted with a small amount of items relative to the total amount of items that a merchant sells. Our data set contains 5 columns for each category of items that a merchant sells; one for the total amount of interactions a user has had with an item category and one column for the number of actions of each action type. Therefore if a customer i has not interacted with any items in a given category k, x_{ij} will be missing for all j corresponding to the k^{th} category. Thus, the observed values in our data set are the entries (i,j), where customer i has interacted with an item in the category corresponding to column j.

Discarding the columns or rows with missing entries is not a good solution in this setting because it greatly decrease the size of our data sets. Instead , we will fit low rank approximations for both data sets using the Generalized Low Rank Model framework. The goal of imputing the missing entries in our data set is to first solve the problem of trying to predict how many interactions a user will have with an item category and the how many actions a user will have for each action type for an item category. We will then use the data sets with the imputed entries to solve the original problem of predicting if a user will be a repeated buyer for a given merchant.

3.1 GLRM

To accommodate the various data types in out data set we formulated identical GLRM models for each merchant. The optimization problem for each model follows the following structure:

$$\sum_{(i,j)\in\Omega} \ell_j(x_i W_j, Y_{ij}) + \sum_{i=1}^{m} 1_+(x_i) + \lambda \sum_{j=1}^{2} |w_j| + \sum_{j=3}^{n} 1_+(w_j)$$

where for $j \in \mathcal{F} \subset \{1, ..., d\}$, ℓ_j is the ordinal loss function for the ordinal valued columns, the hinge loss function for Boolean valued columns, or the Huber loss for real valued columns.

The Huber loss function was chosen over quadratic loss for our real valued data because of frequency of outliers in our data set. For instance, some users have interacted with items of a category an unusually high amount of times or executed a particular action an unusually high number of times. The robustness of Huber loss in dealing with these outliers is a major advantage. Non-negative matrix factorization was chosen due to the fact the that the number of interactions and action types cannot be negative.

To demonstrate this framework we have selected two merchants, one with a relatively low dimensional space (i.e low number of total categories) which will be Merchant 1760 , and one from a high dimensional which will be Merchant 1892. The data set for Merchant 1760 is a (968, 107) matrix with 21 item categories.

3.2 Matrix Completion Results

For both models, we made a training set using 80% of the observations and a test set using the remaining data. In order to select the parameter k, the rank of matrix approximation we ran 5-fold cross validation separately. The cross-validated rank of the model k = 15 for Merchant 1760. The out of sample error for the matrix approximation was 7.54%.

3.3 Linear Model Results

The data set was split into a 80%/20% split training and testing sets. Using the imputed matrix, the logistic classifier had an out of sample error of 8.7% which is a higher that the error of the baseline model. This model can used for the full list of merchants in the future in order to see if predictions can be improved from the baseline model.

References

[1] Madeleine Udell, Corinne Horn, Reza Zadeh, Stephen Boyd *Generalized Low Rank Models: https://arxiv.org/abs/1410.0342* (October 2017)