Online-to-Offline (O2O) Coupon Usage Prediction

by

Daxi (Jack) Cheng
Jianing (Sandra) Cui
Christine Mulcahy
Elena Reynolds
Jianjie (Steven) Zheng

Submitted to

Dr. Joydeep Ghosh

and

Diego Garcia-Olano

MIS 382N - Advanced Predictive Modeling
Master of Science in Business Analytics
McCombs School of Business
The University of Texas at Austin

Fall 2017

Abstract

As retail environments across the globe continue to become more competitive, companies are turning to online-to-offline (O2O) marketing in attempts to increase their customer base and market share. One such company is Alibaba, whose ecommerce platform is the biggest in China and who provided historical online and offline data involving merchants delivering coupons to users for the Tianchi Competition held in late 2016. In this paper, we describe our approach of using the competition's data corresponding to records between January and May of 2016 to predict the usage in June 2016 of a specific coupon offered to a user by a merchant. We describe the raw data and feature extraction work that involved various aggregation levels to attain coupon-specific user-specific, merchant-specific and customer-merchant pair-specific features on a weekly basis, as well as additional RFM and SVD analysis. To account for the time-based data in our predictions, we introduce our cross-validation methodology and process of generating features for the test data set. We employ *hyperopt* in Python to tune the hyperparameters of our online and offline models, whose performance we evaluate with area under the ROC curve (AUC) and prediction accuracy on the test data. Using the online data before including RFM analysis features, we build models using MLP, Random Forests, XGBoost and Adaboost, which we then ensemble together to arrive at the best AUC of 0.85. After adding the RFM analysis features, we attain an AUC of 0.86 solely using XGBoost. We find that day of month for coupon delivery, Recency and Monetary are important features in this model, which is explained by the cyclical effect of monthly salary and marketing analysis. The offline data lacking historical data gives a best AUC of about 0.6. After supplementing the offline data with customer and merchant history from the online data using SVD, our ensemble model AUC increases to 0.8. The important features found in the offline model are the total number of coupons sent out by the merchant, the average number of coupons the user get per week and the distance between customer and merchant. We discover that though merchants usually send out coupons only to the users near them, there is no significant precipitation of redemption rate at farther distances. Therefore, we recommend merchants send out coupons to users within a moderate distance.

Contents

Online-to-Offline (O2O) Coupon Usage Prediction

Introduction

Online-to-offline (O2O) describes the marketing technique utilized by a growing number of businesses around the globe in which retailers offer customers deep discounts via digital means to encourage the purchase of physical goods or services (Zhang 2015). Some of the largest and fastest growing companies that run typical O2O business models include Uber and Airbnb. In China, there are also millions of restaurants, hotels, cinemas, karaokes and barbershops utilizing O2O business models. With the widespread use smartphones, O2O is associated with the large collection of behavioral and location information recorded by various applications, and thus presents an ideal application of big data analysis and machine learning.

Retail and technology giant Alibaba has recognized that in China, a large majority of retail sales are still made at physical stores, and offers a platform for merchants to connect with existing and potential customers (Liu 2017). The Alibaba Group held the Tianchi Competition on the Alibaba Cloud platform from October through December of 2016, allowing participants to access data describing historical consumer behavior and coupon usage. In this project, we make use of this competition's data, using purchase behavior from January through May 2016 to predict whether a specific coupon sent by a merchant was used by a certain customer in June 2016. As coupons from different merchants are of varying popularity, the evaluation metric followed in the competition is to evaluate the average of the area under the ROC curve (AUC) for each coupon. In order to gauge the performance of our predictive models, we look at AUC along with predictive accuracy.

Before we dove deep into the data, we made sure that this project and data set offered opportunities for large scale analysis utilizing the tools learned in our advanced predictive modeling course. Fortunately we did not need to worry about finding additional data, as what was provided by this competition totaled over 7 million unique observations. The data set is big enough, while processing in Python with pandas is still relatively quick. We were also faced with the opportunity to generate meaningful features for our predictive models, as though the number of rows in the raw data is large, very few attributes were given. This feature extraction process highlights a large portion of our project. Finally, we were excited to find a data set provided by Alibaba, the biggest ecommerce platform in China, which offered a description of the data -

something often difficult to find on other popular platforms such as Kaggle.com. And as the combination of big data research and business operation is highly spoken of in the O2O field, this open data from a real business environment with a rather long user history of 6 months can be of great value. Next, we considered what significance our research may provide.

One important reason why companies compete for consumers by offering coupons instead of simply lowering the price of their product is that coupons can engender market segmentation. As noted by Narasimhan (1984), Levedahl (1984), Sweeney (1984) and others, this kind of price discrimination can be profitable as long as coupon users as a whole are more price-sensitive than non-coupon users, an implicit assumption being that companies distribute coupons randomly via the mass-media and rely on consumer self-selection to achieve market segmentation. Such positive effects on sales are confirmed by a number of empirical analyses ever since. For instance, Lattin and Bucklin (1989) found a significant increase in customers' purchase probability after a promotional purchase. Bell, Jeongwen and Padmanabahn (1999) indicated that promotions usually result into brand-switching behavior. Additionally, Taylor (2001) concluded that customers who redeemed a coupon were about seven times more in favor of making purchases in the period after the promotion.

With the advent of panel data on household purchase behavior and the development of statistical procedures, companies can now target households for discount offerings with considerable accuracy and cost effectiveness (Rossi and Allenby, 1993), and thus play a much more active role in market segmentation. Speaking of the advantages of coupon targeting, Moreau, Krishna, and Harlam (2001) believed that the effectiveness of any promotional strategy depends on how accurately channel members predict consumers' perceptions of their promotional activity. Many other analysts predict that targeted promotions will gradually replace mass-media distribution (Shaffer and Zhang, 1995), and that modelling the release of coupons in order to optimize the promotional strategy carries a lot of potential (Buckinx et al., 2003).

Many analysts confirmed this advantage by giving their different perspectives. From a customer-based perspective, Bawa et al. (1997) noted that regular purchasers of a brand have a higher likelihood of coupon redemption than purchasers with low prior purchase probability. So making predictions concerning the proneness of customers for coupons is necessary for defining target segments and for being able to make strategy evaluations. This makes it possible to limit marketing costs and define the exact budgets that need to be allocated. From the coupon-based perspective, Buckinx et al. (2003) recommended to build models for different kind of coupons, as customers can be interested in specific types of promotions. Building

models without taking this into account possibly leads to worse predictions. A merchant-based perspective has also been introduced, as Blattberg and Neslin (1990) noted that redemption rates vary widely across product categories while Taylor (2001) mentioned that preferences for specific categories or brands determine the level of sensitivity for promotions within their respective categories. And from a comprehensive perspective, Papatla and Krishnamurthi (1996) confirmed that the reasons for different sensitivities to promotion could be multidimensional. All the customer features, coupon types and product features matter when it comes to the effectiveness of coupon promotion. As a result building separate models for different kind of promotions seems to be recommended (Buckinx et al., 2003). Continuing this line of thought, we defined our project scope to include extracting features from our raw data based around four perspectives to attain attributes that are either customer-specific, coupon-specific, merchant-specific, before applying any predictive models.

As claimed above, we believe that both customers and merchants will benefit from precise marketing. While the promotion expense of sending coupons through email is close to negligible compared to paper coupon mailing, the random digital delivery of a large number of coupons can irritate the customers and damage the reputation of the merchants. As was observed by Alibaba, people often like coupons issued by their favorite stores and feel disturbed while receiving non-related ads. Thus without careful customer targeting, many merchants may suffer email unsubscriptions from uninterested users. In this project, we address this problem and offer means to target certain groups of customers using certain kinds of coupons by building multidimensional models, which will not only improve the marketing campaign of the merchants but also bring real benefits to the customers. We believe this to be the same reason why Alibaba held this Tianchi Competition last year, and we hope to offer a better solution.

In this paper, after introducing our data set, we outline our approach to attaining our best predictions for coupon usage in June 2016. We describe in detail our methodology for initial feature extraction and exploratory data analysis, followed by our improved efforts using RFM analysis and singular-value decomposition (SVD). After these additional steps we describe novel aspects of our work, including our cross validation technique designed to address the time-based characteristic of our data, and our employed parameter tuning methodology using *hyperopt* in Python. Finally, we dive into each individual model built using the initial extracted features before showing how our predictions improve with the addition of RFM analysis. Our final model ensembles together the best performing methods with their tuned hyperparameters.

Data Acquisition and Description

The data downloaded for use in this project is provided by Alibaba Cloud as part of the Tianchi Competition held between October and December of 2016. Served as three separate CSV files, the data describes transactions between Alibaba suppliers and customers starting at the beginning of January 2016 and ending after the first two weeks of July 2016. The online training data found in the first file along with the offline training data in the second file represents records for January through June, while the test data file holds the transactions for July. On the competition webpage, where the data can be downloaded after registering with Alibaba Cloud, brief feature descriptions are given to help interpretation of the vague field names in the raw data sets. We have included a link to the competition webpage in our Appendix.

Both the online and offline training files share six columns in addition to each possessing one unique field. The columns common to the two training files of raw data include *User_id*, *Merchant_id*, *Coupon_id*, *Discount_rate*, *Date_received*, and *Date*. The first two features in this list represent anonymized, unique integer identifiers for specific customers and merchants whose records were sampled for the competition. In the instances when a merchant delivered a coupon to a customer, an integer uniquely identifying a specific coupon offered by a merchant is present along with a discount method. Discounts were defined by one of the following representations: a floating-point number between 0 and 1 that represented the discount rate, a purchase requirement in Yuan and amount discounted in Yuan if the customer meets the minimum purchase amount, or a fixed and discounted price. The *Date_received* column specifies a date, in YYYYMMDD format, in the records where a coupon was sent from the merchant to the customer (wherever *Coupon_id* not null). Finally, if the customer made a purchase from the merchant, the corresponding date of the transaction is stored in the *Date* column.

In these two training data sets, each row or “transaction” falls into one of three potential categories. First, if the *Coupon_id* is null and the *Date* column is not null, this combination represents a customer making a purchase without receiving a coupon. Conversely, where the row’s *Coupon_id* is not null while *Date* takes a null value, the customer receives a coupon but does not make a purchase, or in other words, they do not use the coupon. Finally, where *Coupon_id* and *Date* are both not null, the customer makes a purchase with the coupon. Our team assumes that there are no records corresponding to receiving the coupon and then making a purchase without the coupon.

As mentioned above, the online and offline training data files each possess a data field not found in the other file. Unique to the online training data file is the *Action* field, where each instance is assigned an integer value of 0, 1, or 2, with the values respectively representing an action by the customer of 'click', 'purchase', or 'received coupon'. Found only in the offline training data file is the *Distance* field, where a not-null value falling between 0 and 10 represents the distance (divided by 500, in meters) between the merchant and the customer's most frequently transmitted location.

The test data set containing records for July 2016 contains only the columns found in the offline data set, but without the *Date* column, meaning that we are unable to determine whether a purchase was truly made or not. Screenshots of the first five rows in each of the three raw data sets can be found in Figure 1. While participants in the competition submitted their predictions for the usage of specific coupons sent between merchant and customer in July 2016, our team instead uses the training data between January and May to predict usage in June, as we cannot deduce the target variable and gauge our prediction accuracies using the original train and test splits.

	User_id	Merchant_id	Action	Coupon_id	Discount_rate	Date_received	Date
0	13740231	18907	2	100017492	500:50	20160513	null
1	13740231	34805	1	null	null	null	20160321
2	14336199	18907	0	null	null	null	20160618
3	14336199	38810	0	null	null	null	20160126
4	14336199	18907	0	null	null	null	20160127

	User_id	Merchant_id	Coupon_id	Discount_rate	Distance	Date_received	Date
0	1439408	2632	null	null	0	null	20160217
1	1439408	4663	11002	150:20	1	20160528	null
2	1439408	2632	8591	20:1	0	20160217	null
3	1439408	2632	1078	20:1	0	20160319	null
4	1439408	2632	8591	20:1	0	20160613	null

	User_id	Merchant_id	Coupon_id	Discount_rate	Distance	Date_received
0	4129537	450	9983	30:5	1	20160712
1	6949378	1300	3429	30:5	null	20160706
2	2166529	7113	6928	200:20	5	20160727
3	2166529	7113	1808	100:10	5	20160727
4	6172162	7605	6500	30:1	2	20160708

Figure 1. First 5 rows of each raw data file (top to bottom): online training, offline training, and offline test data. The first row of the online training data represents a customer receiving a coupon, offered as 50 Yuan off of a 500 Yuan purchase but not making a purchase. In the offline training data (middle photo), the third and fifth rows show the same customer receiving the same coupon from a merchant on different dates, without redeeming it.

Methods

Pre-processing

Our first step of data pre-processing involved removing a large number of duplicate rows from the training data sets. This step reduced the number of instances in the online training data from 11,429,826 to 5,822,543 and the number of offline training rows from 1,754,884 to 1,716,991. Because our goal is to predict coupon usage, we were not interested in the instances not involving a coupon being sent, so our next step was to filter both data sets to include only the rows where a coupon was sent between the merchant and customer (i.e., where *Coupon_id* is not null). This step, followed by separating the rows where either date column corresponded to June into a test set, resulted in a final 667,453 rows of online training data and 91,647 rows of online test data, along with 918,142 rows of offline training data and 97,247 rows of offline test data.

An important distinction between the online and offline data sets discovered by our team is such that no single *Merchant_id* is common to both the online and offline data sets. We reasoned that merchants were considered online if they have no physical storefront where their goods and services are regularly purchased, such as the case for a rideshare business like Uber; offline merchants would include retailers whose transactions with coupons occur at a permanent location, like grocery stores or auto repair shops. There are, however, customers whose unique identifiers appear in both the online and offline data sets. In the later sections of this report, we describe in more detail how we handled the online versus the offline data.

Exploratory Data Analysis

We began exploring the data by looking at the relationships between the number of coupons sent out, the portion of coupons used online, and the Merchant ID. We saw that both the number of coupons as well as the portion used were highly variable from one merchant to the next. While the vast majority of merchants sent out less than 25 types of coupons, a few sent out many more, with one merchant sending over 200 types of coupons, as shown in Figure 2. The portion of coupons used for a given merchant ranged from none to 100 percent and was uncorrelated with the number sent out, shown in Figure 3. Similarly, the number of coupons and portion used by a given customer varied among customers. Some customers used all of the

coupons they received, some used none, and everything in between, as shown in Figure 4. This portion of used coupons was independent of the number of coupons received by a given customer.

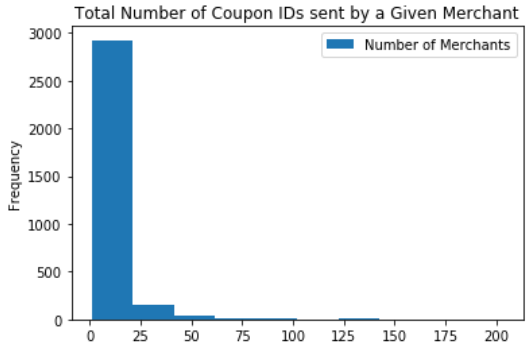

Figure 2. Total number of unique coupon types sent by merchants.

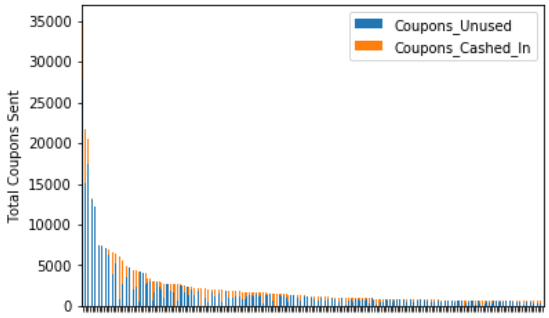

Figure 3. Number of coupons sent by/used for top 200 merchants.

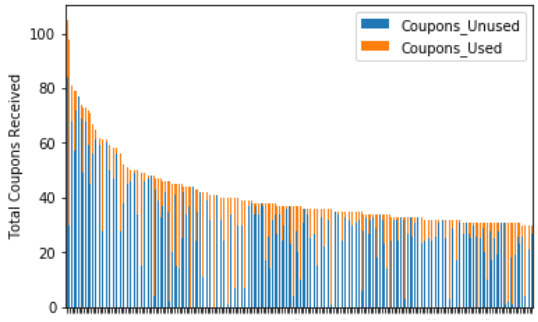

Figure 4. Number of coupons sent to/used by top 200 customers.

Feature Extraction

Our work consists of two parts. In the first part, we mainly investigate the online data, trying to use the historical online data to predict whether redemption of a certain coupon occurs in the

future. Afterwards, we use the same logic for the offline data. Due to the imbalance between the online and offline data sets, we concluded that Alibaba got more online data than offline. The typical question of the O2O businesses is that the offline part doesn't have enough user historical offline data to do the prediction analysis, our group adopted the approach that connects the online and offline historical data together by different user id, and it turns out this way works pretty well. Particularly, for our offline data, which we can only generate a sparse matrix of the user and the merchant interaction, we use a SVD matrix factorization method to generate the features we need for the prediction. We begin by first describing the initial features extracted, shown in Figure 5, before introducing additional approaches using RFM analysis and SVD.

Because our raw data contained only seven unique columns in each training data set, our team was faced with a great opportunity to generate our own features that we thought may be important in predicting coupon usage. This task involved using pandas within Python to perform multiple group-by aggregations on our data, grouping on the different unique identifier columns both separately and then together. For each aggregation, we wrote a main function in which we pass in raw training data to add the new features as additional columns to the data frame.

First, we generated features specific to each *Coupon_id*. We extract two new fields, which we define as *coupon_requirement* and *equivalent_discount*, by mapping self-written functions over the *Discount_rate* field. The *coupon_requirement* is simply the minimum purchase amount *x* when *Discount_rate* takes the form *x:y*, and null otherwise. We assume that the minimum purchase amount *x* is our best guess at the amount spent by the customer, so *y/x* gives us the amount by which the purchase was discounted, and then we calculate *equivalent_discount* by subtracting *y/x* from 1. As an example, Figure 5 shows that if the *Discount_rate* is given a value of '500:50', the *coupon_requirement* and *equivalent_discount* are defined respectively as 500 and 0.90. In addition to these two new fields, we also took the *Date_received* column to extract the day of the week and day of the month when the coupon was sent, with the day of the week with a value of 0 designating Monday and the max value of 6 occurring on Sunday.

Next, we grouped by *Merchant_id* to generate three merchant-specific features. Taken as the unique row count corresponding to each merchant is *Tot_coupon_sent*, representing the total number of coupons that retailer has sent out. If a customer receives the same coupon twice from this merchant, both are included in this sum. We then defined number of unique

Coupon_id values for each merchant as *Num_type*. Finally, we determined what fraction of the merchant's targeted customers have ever made a purchase with their coupon and titled the new field *Merchant's Total Usage Fraction*.

Grouping on *User_id*, we extracted the total number of coupons received (*Tot_coupon_got)* and the total number used (*Num_coupon_used*) by each customer. For our fourth and last aggregation level, we grouped by each unique combination of *User_id* and *Merchant_id* to extract features describing the relationship between the customer and retailer, which have been defined with more self-explanatory names. For each pair, we generated the totals *Customer's Total Usage for Merchant's Coupon* and *Customer's Total Received by Merchant* to then calculate the ratio of the two as *Customer's Total Usage Fraction for Merchant*. We also calculated the fraction of the total number of coupons received by the customer that came from that merchant as *Customer's Inbox Share by Merchant*.

Finally, we generated a new column in the training data for our target as *Coupon Used*, a binary variable designated as 0 when the coupon is not used by the customer or 1 when the customer makes a purchase with the coupon. To get these values, we followed the logic described above at the end of *Data Acquisition and Description* pertaining to whether the *Coupon_id* and *Date* columns are null, rather than trying to ascertain the information from the *Action* column, which is found only in the online data. Figure 5 shows the first five rows of the online training data with all of these new features brought together into new columns.

	User_id	Merchant_id	Coupon_id_x	coupon_requirement	equivalent_discount	Date_received_week	Date_received_month	Tot_coupon_sent	Num_coupon_used	num_type	Customer's Total Usage Fraction for Merchant	Customer's Total Usage for Merchant's Coupons	Customer's Total Received by Merchant	Merchant's Total Usage Fraction	Customer's Inbox Share by Merchant	Coupon Used
0	13740231	18907	100017492	500	0.9	4	13	35140.0	0.0	123.0	0.0	0	6	0.210273	1.000000	0
1	15137031	44706	100071973	50	0.9	3	17	1607.0	0.0	11.0	0.0	0	1	0.168015	0.250000	0
2	15137031	29007	100028000	30	0.9666666666666667	1	5	248.0	0.0	4.0	0.0	0	1	0.088710	0.250000	0
3	15137031	18907	100086665	300	0.8333333333333334	2	6	35140.0	0.0	123.0	0.0	0	2	0.210273	0.500000	0
4	15137031	18907	100160219	300	0.9	3	17	35140.0	0.0	123.0	0.0	0	2	0.210273	0.500000	0

Figure 5. First 5 rows of training data with newly extracted customer-, coupon-, merchant- and customer-merchant-specific features.

The procedures described in this section thus far were first performed only using our decided split of training data and not the data corresponding to June from the raw training file. A pointer to our code is located in the Appendix to assist in understanding our process. For our holdout or test data, we carefully generated our additional features with precaution to avoid data leakage. Because we should not know the *Date* column for the test data, the new data was only used in generating the new columns if the feature depended only on the given information. In other words, we could extract the *coupon_requirement* column from the new data, but for *Merchant's Total Usage Fraction*, we simply linked the value generated with the historical

training data by the *Merchant_id*. Any null values remaining in the test set after feature extraction were dealt with via mean imputation. See Figure 6 below for clarification.

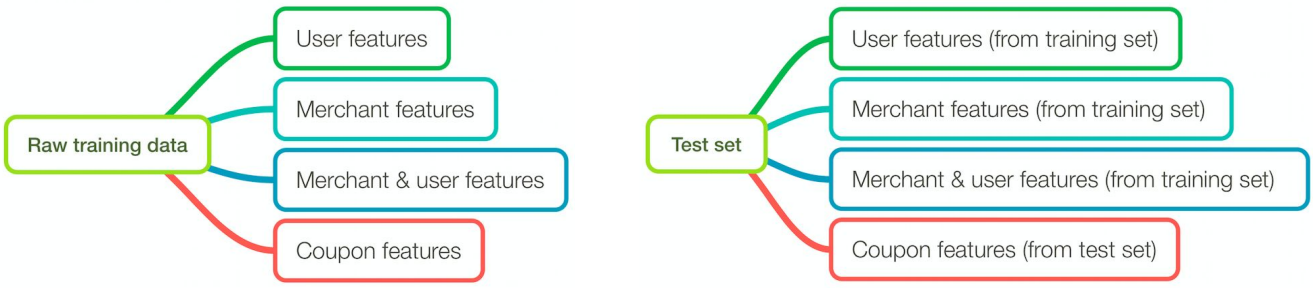

Figure 6. Methodology for feature extraction, performed left to right.

Feature Selection

Because our raw data did not involve a large number of attributes, we did not have to invest a great amount of time in pursuing feature selection and reduction techniques like Principal Component Analysis (PCA). Even so, the curse of dimensionality still finds its way into influencing our modeling choices. Three of the features supplied in the raw data set which served as the primary keys in our feature extraction steps, those being *Coupon_id*, *User_id*, and *Merchant_id*, if used in our model, would be treated as categorical variables as the integers assigned to these variables are only used as identifiers. In the online training data set alone, there exist 229,819 individual customers, 3,154 unique merchants, and 24,765 different coupons. Including these features as predictive variables would exponentially increase the size of the feature space, and the number of unique values for these variables in our case is only a sample of those from a small period of time. Fortunately, the feature extraction work we performed engrained information about each unique identifier into continuous variables.

RFM Analysis

Our access to customer transaction history allows our team to utilize the RFM analysis, a direct marketing technique that firms have turned to for over three decades to determine the customers who are most likely to respond to their efforts (Hughes, n.d). From a database

containing purchase records, each customer is assigned to a quintile (or cluster) between 1 and 5 based on Recency, Frequency, and Monetary scores relative to the rest of the dataset. Recency describes how recent the customer's last purchase was, Frequency tells us how often the customer buys, and Monetary is assigned as the average amount of money the customer spent over a certain period. A customer who purchased most recently and most frequently, in addition to spending the most money on average, is assigned a 5 for all three measures. In most cases, Recency scores are the most indicative of whether a customer is likely to respond, with Frequency being a less important indicator, and Monetary often showing the least dramatic difference in response rates, hence the ordering of the acronym (Hughes, n.d.).

Though we do not have a direct measure of these customer-based scores, we again were presented with an opportunity for more feature extraction and were able to approximate all three indicators for every customer in order to integrate the information into our model. We set out on this task with the expectation that Recency would hold the most importance in our prediction of the three new features, followed by the Frequency and Monetary measures.

For recency, we simply calculated the measure as the time difference between the date the coupon is sent out and the last time this user made an active purchase (most recent not-null *Date* value). We did not specify the most recent purchase used in the calculation of Recency to be with a certain merchant, as we speculate that the sparsity of our data may lead to a poor prediction result.

For frequency, we computed the each customer's average coupon redemption amount on a weekly basis. We later find that this feature is not of great importance in our predictions and we pose that the reason is that the time span of the data for each customer is variable. In attempts to amend this finding, we then tried to normalize the time period to the time difference between the first time this user appeared in our data set and the date the coupon was delivered. But this also led to low feature importance, as the average redemption rates seemed to vary greatly between customers with different lifetimes. Many users simply use one or two coupons before leaving their account inactive for a long period of time, while many other users' first appearances occur only a few days before the delivery date for the coupon we are trying to predict. Due to this imbalanced data distribution, we decided to give up the Frequency measure for model simplicity.

As for the monetary measure, Alibaba did not provide the exact amount of money in spent in each of these transactions, nor the number of items included in the transaction total. However, since there is a field which tells us something about the minimum purchase amount in

Yuan required for the customer to use the coupon, we can be sure that this purchase would be surely amount to more than that requirement, and the number of items per purchase is not necessary for the Monetary indicator. In addition, because generally larger minimum purchase requirements amount correspond to higher equivalent coupon discount rates, if the customer spent some amount of money that is much more than the coupon purchase requirement, they might use another coupon with a larger requirement value to get higher discount. To summarize, we think that the minimum requirement of used coupon can serve as a good representative of the monetary indicator. We linked our data to Tableau to create visualizations of our RFM analysis. Figures 7 and 8 below show of redemption rates among customers in our data, clustered into deciles for Recency and Monetary measures, respectively. Figure 9 shows all clusters, where the data fall into a decile for each indicator.

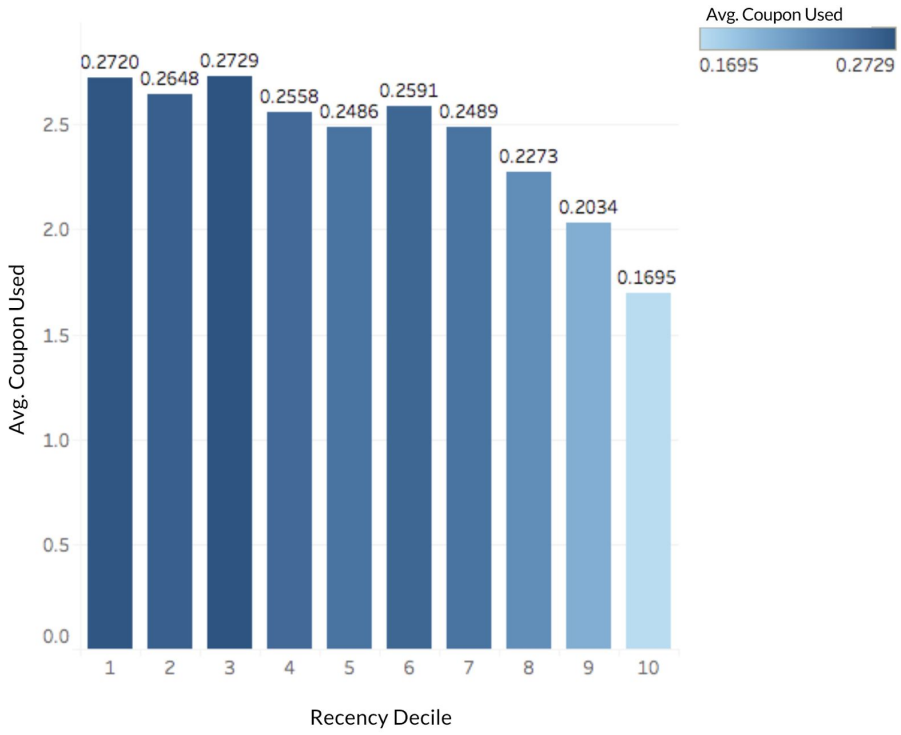

Figure 7. Redemption rates vs. Recency deciles. Rates shown on color scale as average coupon usage rate among clusters.

Figure 8. Redemption rates vs. Monetary deciles. Average redemption rate among clusters shown with a color scale.

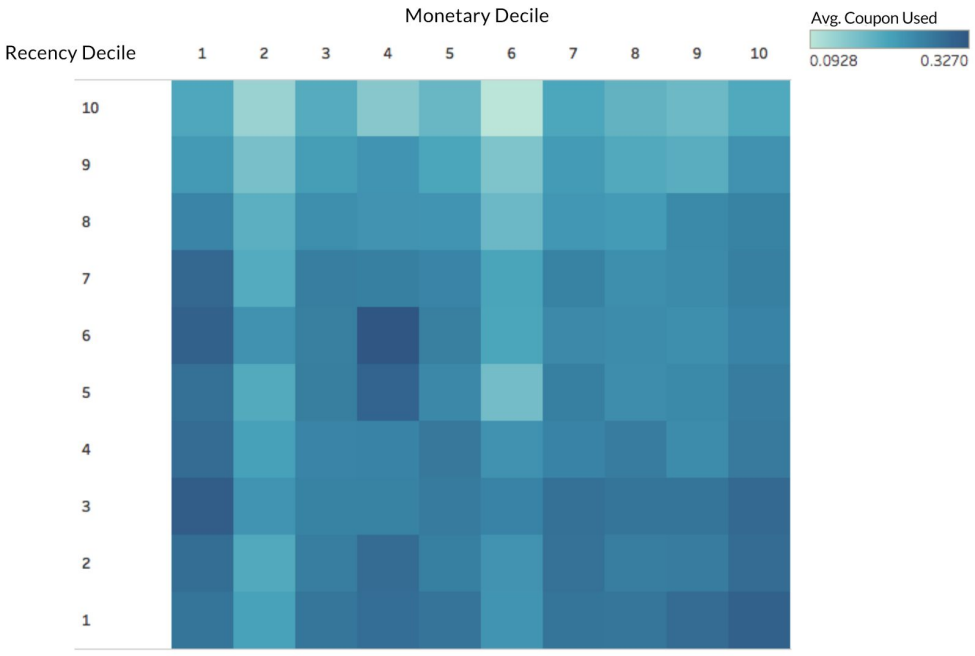

Figure 9. Monetary Deciles vs. Recency Deciles. Color scale shows average coupon usage rates.

Singular-value Decomposition (SVD)

As in typical collaborative filtering problems, the features built for each combination of merchant and user pose the problem of missing data. The matrix we get from the users' redemption rate of the merchants is very sparse, especially for the offline data where the total data size of transaction is much smaller in that e-commerce platform. So our group took the advice given by Dr. Ghosh about treating the problem as a recommendation system problem and adopting collaborative filtering to deal with this type of missing data problem. Without this, especially in the offline data, the limited historical purchase records would limit our models' AUC to a level of about 0.65.

We implemented the *Surprise* module in Python, which contains various matrix factorization-based algorithms. We used the simple but powerful SVD algorithm due to the limit time and computation expense in this problem. The logic is a quite simple probabilistic matrix factorization approach. First, we did a quantile cut on the redemption rate to make it like a rating from 1-5, which can be easily parsed by the module. Then the module performed the following work. The r_{um} we are trying to predict, the user u's rating for merchant m is modeled by:

$$\widehat{r_{um}} = \mu + b_u + b_m + {q_m}^T p_u$$

If user u is unknown, then the bias b_u and the factors p_u are assumed to be zero. If merchant m is unknown, then the bias b_m and the factors q_m are assumed to be zero. By using ${q_m}^T$ and p_u, which are respectively m-by-D and D-by-u dimensional matrices where D is much smaller than m and u, we use a D-dimensional vector to represent each user and merchant. We use the SGD algorithm to minimize the regularized loss given by:

$$\sum_{r_{um} \widehat{\in R_{train}}} (r_{um} - \widehat{r_{um}})^2 + \lambda({b_u}^2 + {b_m}^2 + ||p_u||^2 + ||q_m||^2)$$

Therefore at last, we could use our sparse matrix to generate a full matrix of users' redemption rate level of merchants.

Cross-Validation

Due to the time-based characteristic of the data, this project posed an interesting challenge when extracting features containing aggregate coupon usage data. Because coupon usage was what we were ultimately predicting, we had to ensure there was no "leakage" of this information into our extracted features for a given prediction period. We accounted for this by producing a separate set of extracted features for each testing period and ensured that it only contained aggregate usage data for the training time period.

This individualized process of feature extraction meant that we could not utilize a traditional cross validation system. Instead, we set up two cross validation periods. The first period trained on data from January through March and predicted for April. The second period trained on data from February through April and predicted for May. Finally, we tested each model using training data from January through May and testing data from June.

Parameter Tuning with Hyperopt

In this project, we are facing the problem of tuning the hyper parameters and ensembling the models at the same time. One model has around 5 hyper parameters to tune in total, which is a fair assumption with not too much complexity, and each hyper parameter has about 10 different possible values to test. Even if each model can be trained quite fast with only 5 minutes to get the final result, we would need several months to finish the entire grid search work of this combinations. As when combining different method models together, their joint combination of different values of hyper parameters would grow exponentially and that would lead to a huge cost of computation, we needed to find a better way than the grid search method which is introduced during class.

We decided to adopt the Python module called *hyperopt*, which is a widely used framework on hyper parameter tuning and can help us to achieve tuning the hyper parameters in a relatively short period of time. Our team specifically used the random search with 10% chance in conjunction with the Tree-structured Parzen Estimator (TPE) method with 90% chance. Unlike the typical Gaussian process method which model the $p(y|x)$ directly (here we denote y as the scoring/loss function, whereas x stands for the hyperparameters combination), this method models the $p(x|y)$ and $p(y)$ and calculates the result by Bayes' theorem. Therefore, this is also an Bayesian method of hyperparameter tuning method. To gain a more thorough

understanding of how this algorithm works, our group studied the 2011 paper by James S. Bergstra et al. We next summarize how this TPE approach works by describing its steps.

First, some weak prior distribution of the hyper parameters is taken. This integrates the domain prior knowledge of the distribution of the parameters into the algorithm. Here we can define different kinds of prior distribution for different types of hyper parameters, such as for the slack variable C in the SVM algorithm. Instead of setting a normal distribution, we can use a log-normal distribution that would make sure the C would not take on nonsensical negative values, and so it would be distributed around 1 as we expected. For the number of trees in boosting algorithms, we need to set this a quantile uniform distribution that would take only integral values.

This algorithm begins with some random guess within the prior space to begin the process. After several random guesses, TPE classify the lowest 20% of loss points we got as class 1, and the remaining 80% as class 2 (This threshold could be changed, we just go by default 20%). For the next test points, the following metrics are computed to determine what kind of hyper parameters combination to test. Select the one with highest score to be the next point to test and use the resulting test loss *y* to refresh the estimation about the $p(x|y)$ and $p(y)$. Here, this algorithm just converts the next combination selection problem to a two-class classification likelihood-modeling problem. Based on the hyperopt framework, we used the algorithm of 0.9 TPE and 0.1 random search of the hyper parameter space. We set out to tune the following parameters shown in Figure 10 below.

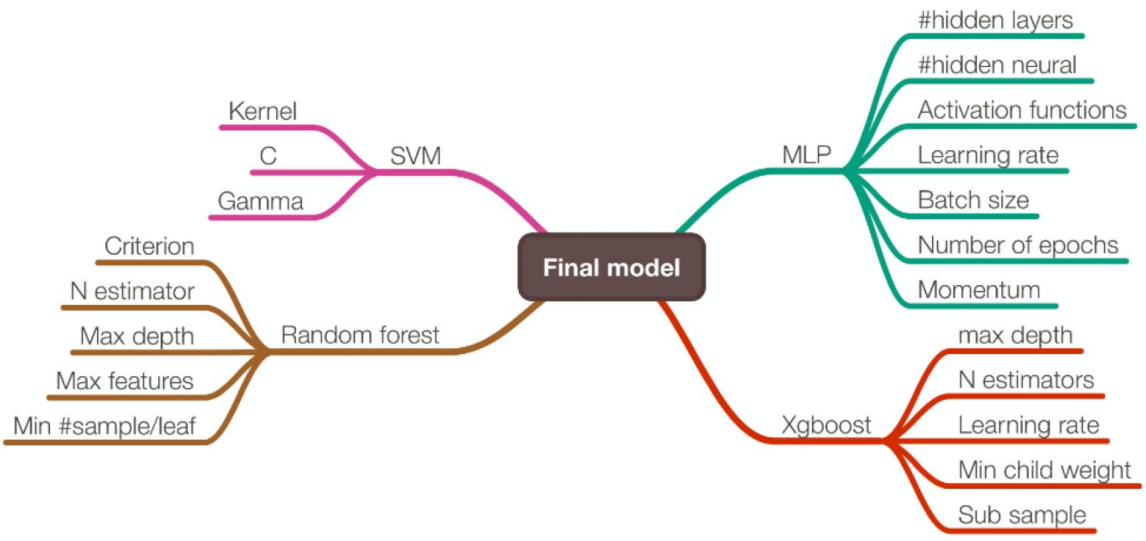

Figure 10. Summary of all model parameters considered during hyperopt tuning.

Our final model consists of four different models and ensembling them all together with a weighted average of their predictions. For different types of the hyper parameters, we defined five different types of prior distributions and choices, outlined below.

1. Choice-based: Such as the type of kernel in SVM, the types of activation function in MLP and the Criterion in random forest algorithms. These types of parameter are based on the choice of different options. We define a random choice of all possible option with equal probability for each of them.
2. Hyper parameters associate with count and number: Like number of hidden layer in MLP, number of estimator in random forest and the max depth in XGBoost algorithm, they can only take integral numbers. We define the quantile uniform distribution for all of these hyper parameters.
3. Parameters with variation range: Like the slack variable C in SVM algorithm can only be meaningful when C is larger than 0. We define a log-normal distribution for this parameter variable.
4. Parameters where scale matters: For parameters like the learning rate in both MLP and boosting algorithms, what really matter might not be the exact number of the learning rate itself, but the magnitude of the scale of those parameters. In other words, 0.001 and 0.002 might not be different significantly, but they are quite different from 0.1 and 1. For those kind of parameters, we search the log space to get the proper scale of those data and define a log uniform distribution for those hyper parameters.
5. Model probability weight: When we ensemble the different models together, here we adopt the typical way of just weighted average the predict probability of each model so we leave the weight of each model also a hyper parameter for hyperopt to tune and define a uniform distribution for it.

Predictive Modeling Methods

We tested our data using five individual predictive models: Random Forests, Multi Layered Perceptrons (MLP), Gradient Boosted Decision Trees (GBDT), XGBoost, and Adaboost. Then we selected the most successful models to include in an ensemble model to further improve our results. Initially, we ran each model before adding the RFM features to our dataset. This gave us acceptable, but more modest success in predicting coupon usage. After adding RFM

features to our training and testing data, our models saw great improvement. Each of the models we trained are discussed in detail in the following paragraphs.

I. Random Forests

Random Forests are ensembles of decision tree models which use a random subset of features in each tree. They're known for running relatively quickly and handling both missing and unbalanced data well. Because of our large and somewhat unbalanced dataset, we expected Random Forests to be a good fit for our predictions. When tuning the parameters for Random Forests, we used hyperopt to search for optimal parameters such as criterion, max_depth, and max_features.

II. Multilayer Perceptron (MLP)

The multilayer perceptron (MLP) is a feed-forward neural network consisting of three or more layers of perceptrons. Because MLP is so good at modeling complex problem, it is recognized as a universal approximator. However, its high flexibility means that it can also have a tendency to overfit to a training set. It can also take a long time to train a new model. We expected that the MLP model would do a good job of predicting on our dataset and we were hopeful that our cross validation method would be able to counteract any overfitting. When tuning the parameters, we adjusted several parameters such as hidden_layer_sizes, activation, solver, learning_rate, batch_size, and learning_rate_init.

III. Gradient Boosted Decision Trees (GBDT)

Gradient Boosted Decision Trees (GBDT) is another ensemble method based on decision trees. Unlike Random Forest however, GBDT builds each tree one by one while trying to account for the residuals of the previous tree. GBDT shares the advantages of Random Forests, but are known for their often better performance. One disadvantage of GBDT is that there are many hyperparameters to tune, however, we suspected that the use of hyperopt would help us to overcome this drawback. And we have chosen to tune several parameters including loss, learning_rate, n_estimator, and max_depth.

IV. XGBoost

XGBoost is a version of GBDT that has been optimized for speed and performance. It performs well for large datasets and classification problems like ours. It is known as a favorite model of many Kaggle competition winners and as such, we expected the model to work well for our predictions. Some key parameters we have chosen to tune are nthread, max_depth, n_estimators, learning_rate, max_delta_step, and objective.

V. AdaBoost

Adaboost is another ensemble method for combining weak classifiers. Like GBDT, it uses the previous tree's residuals to determine the training set for the next tree. It also optimizes the weights applied to each tree when averaging the results. Adaboost is simple to implement with few parameters to tune and is fairly resistant to overfitting. However, it is sensitive to noise in the data and requires adjustment for imbalanced class datasets. For AdaBoost, we have chosen to tune parameters such as n_estimator, learning_rate, and algorithm.

VI. Ensemble Method

For our ensemble method, we combined the four best models in terms of AUC and accuracy we achieved. We used hyperopt to determine the best weights to apply to each model in the final averaged model.

Results

Model Performance

First we will present the model performance comparisons from using the online data, before and after additional feature extraction with RFM analysis. As shown in Figure 11 and Table 1 below, our results suggest both Random Forests and MLP perform relatively better than other methods including GBDT, XGBoost, and AdaBoost in terms of prediction accuracy. On the other hand, GBDT, XGBoost, and AdaBoost perform slightly better than Random Forests and MLP in terms

of AUC. And we include our final ensemble model, the ensemble performs better than all the other models in terms of both accuracy rate and AUC value. After we extracted the RFM information from our dataset, we then fit a model using XGBoost with tuned parameters. Figure 12 shows the improved performance from the addition of RFM features to the XGBoost model, giving us an AUC of 0.86. The addition also increased the prediction accuracy to 0.79 with XGBoost, from a previous maximum value of 0.77 with the ensemble method.

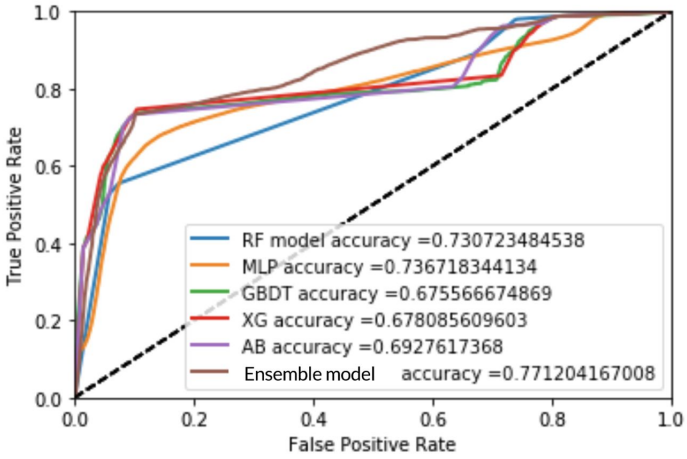

Figure 11. Receiver operating characteristic (ROC) curve comparison across online models prior to RFM addition.

Table 1. Online Model Performance Comparison Prior to RFM

Model	AUC	Accuracy
Random Forests	0.78	0.73
MLP	0.79	0.73
GBDT	0.81	0.68
XGBoost	0.82	0.68
AdaBoost	0.81	0.69
Ensemble	0.85	0.77

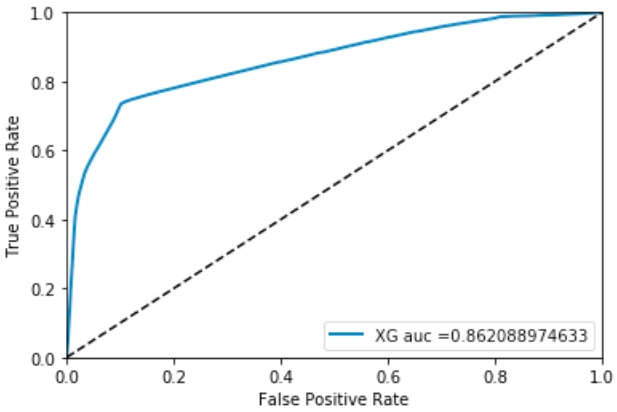

Figure 12. ROC curve for online XGBoost model including RFM analysis, showing the improved AUC of 0.86.

Before adding SVD or the customer and merchant historical data into the features, we initially approached the offline data by running the same first set of models as the online data to compare the model performances. Figure 13 and Table 2 below show that all the models perform relatively similarly in this case, achieving predictive accuracy around 0.57. In terms of AUC value, Random Forest performed the best with 0.62 AUC value.

Because the performance in offline data was far from what we expected, we implemented the SVD factorization method to extract the features to use for our prediction. Further, we also decided to include the historical data from the online data set to see how well the models then performed. Figure 14 and Table 3 show the result of our model after we include SVD factorization method and historical data. Overall, all the updated models performed much better than the original offline models. Among all the individual models, GBDT performed the best with 0.79 AUC value and 0.71 accuracy rate. After combining all the models, our ensemble model outperformed GBDT with a 0.80 AUC value and 0.72 accuracy rate.

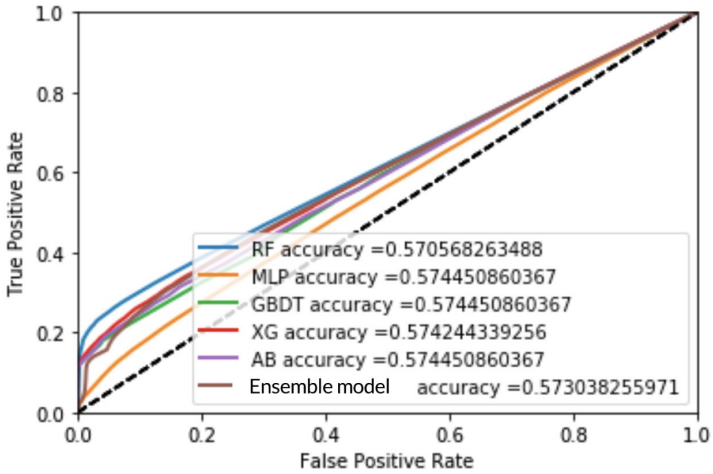

Figure 13. ROC curve comparison across offline models before integrating SVD and historical data.

Table 2. Initial Offline Model Performance Comparison

Model	AUC	Accuracy
Random Forests	0.62	0.57
MLP	0.55	0.57
GBDT	0.59	0.57
XGBoost	0.60	0.57
AdaBoost	0.59	0.57
Ensemble	0.60	0.57

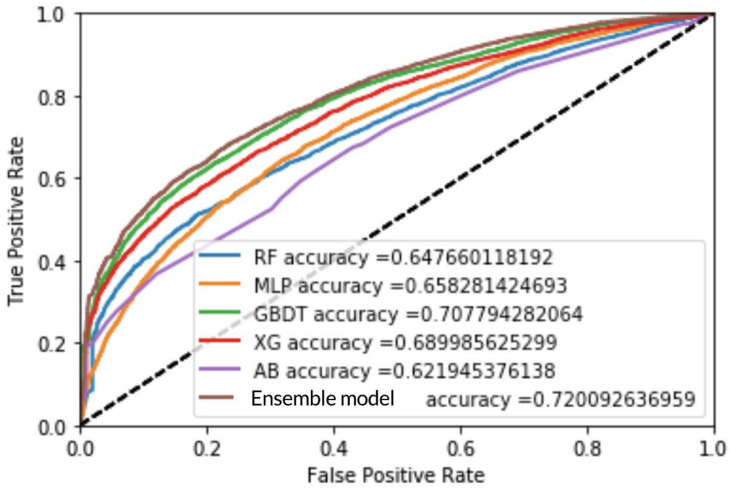

Figure 14. ROC curve comparison across offline models after integrating SVD and historical data.

Table 3. Offline Model Performance Comparison with SVD

Model	AUC	Accuracy
Random Forests	0.71	0.65
MLP	0.72	0.66
GBDT	0.79	0.71
XGBoost	0.76	0.69
AdaBoost	0.68	0.62
Ensemble	0.80	0.72

Feature Importance

Analyzing the feature importance showed that the features we generated through RFM approximation were highly important in the final XGBoost model, ranking second and third in the feature importance table shown in Figure 15, even more important than the equivalent discount rate and the minimum requirement for the coupon on the online part of our data.

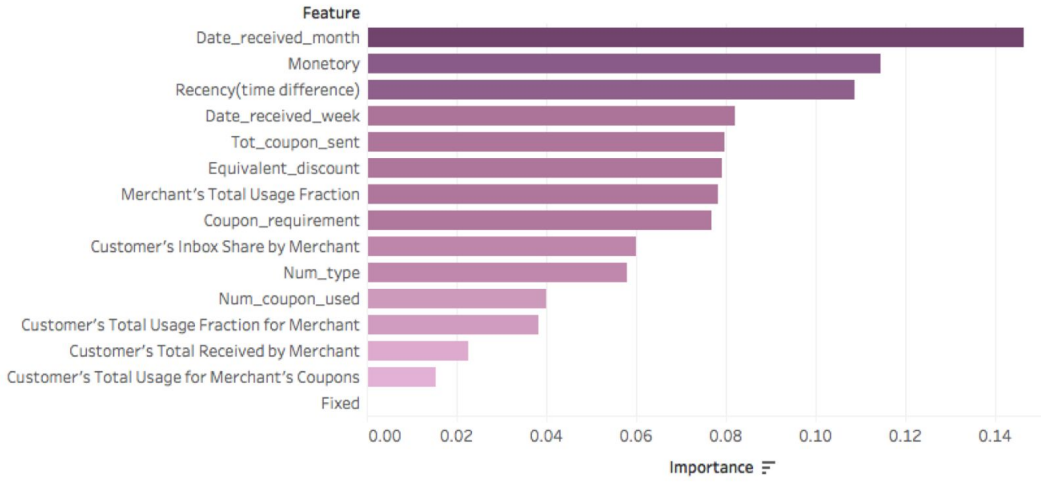

Figure 15. Feature importances found through online data analysis.

For the offline data, the result of feature importance analysis was slightly different, as shown by Figure 15. We can see that the total number of coupons sent by a merchant and the average number of coupons received by a user matters a lot. Besides, the distance between the user's usual login place and the merchant matters, which intuitively makes sense in the O2O business. Figure 17 shows that most of the offline merchants tend to send coupons mainly to the users near them, in other words, within 500 m. However, if we look at the average redemption rate vs. distance, as shown in Figure 18, there is not a direct simple cut-off line for distance. Though distance is an important feature in our model, it is jointly segment the feature space together with other features instead of making some judgement all by this feature itself.

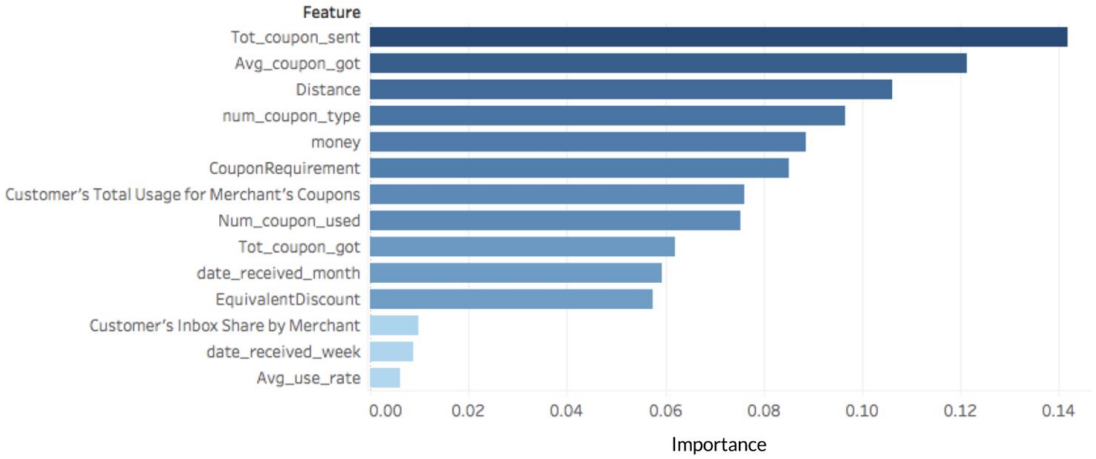

Figure 16. Feature importances through offline data analysis.

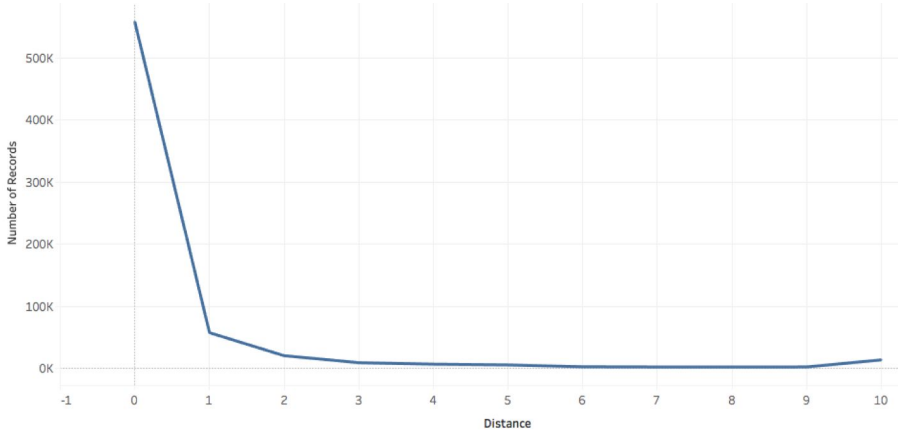

Figure 17. Number of data points (total coupons sent) versus distance to retailer in 500 m.

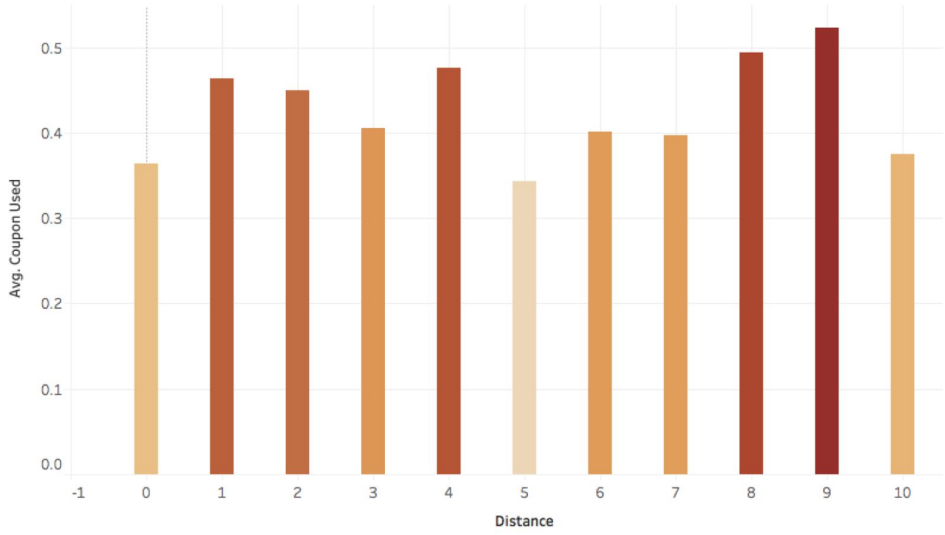

Figure 18. Average redemption rate, represented by color scale, for offline data clustered into deciles by distance from retailer.

Conclusion

Reflecting on our analysis, we learned several key lessons. First, we found that hyperopt is a valuable tool to tune the hyperparameters for our models. Using this, and by combining the models with tuned parameters, we were able to improve the predictive accuracy and AUC in our models, which was our ultimate goal. Furthermore, in the online data set, it was highly beneficial to include the RFM features to improve coupon usage rate predictions. On the other hand, when we discovered that the model built with the offline data did not perform as well as that for the online data, we implemented the SVD factorization method and merged past online information with the offline data to improve our prediction for offline coupon usage rate. The models we developed can be reproduced to predict coupon usage for new customer and merchant combinations. This has the potential to add business value for merchants and satisfaction for customers.

References

Bawa, Kapil, Srini S. Srinivasan, and Rajendra K. Srivastava. "Coupon attractiveness and coupon proneness: A framework for modeling coupon redemption." *Journal of Marketing Research* (1997): 517-525.

Bell, David R., Jeongwen Chiang, and Venkata Padmanabhan. "The decomposition of promotional response: An empirical generalization." *Marketing Science* 18.4 (1999): 504-526.

Bergstra, J.S., Bardenet, R., Bengio, Y. et al. (2011) Algorithms for Hyper-Parameter Optimization. Presented at the 2011 Neural Information Processing Systems Conference, 12-18 December, Granada, Spain. Available at: https://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.

Blattberg, Robert C., and Scott A. Neslin. *Sales promotion: Concepts, methods, and strategies*. Prentice Hall, 1990.

Buckinx, Wouter, et al. "Customer-adapted coupon targeting using feature selection☆." *Expert Systems with Applications*26.4 (2004): 509-518.

Hughes, A. (n.d.). A Note on RFM Analysis. *Harvard Business Publishing*. Access granted through MSBA: Marketing Analytics I---Fall 2017 Coursepack.

Jeuland, Abel P., and Chakravarthi Narasimhan. "Dealing-temporary price cuts-by seller as a buyer discrimination mechanism." *Journal of business* (1985): 295-308.

Lattin, James M., and Randolph E. Bucklin. "Reference effects of price and promotion on brand choice behavior." *Journal of Marketing research* (1989): 299-310.

Levedahl, J. William. "Marketing, price discrimination, and welfare: Comment." *Southern Economic Journal* (1984): 886-891.

Liu, P., Bai, X. B., Jia, J., and Wang, E. (2017). The Accelerating Disruption of China's Economy. *Fortune, Commentary: China* [online]. Available at http://fortune.com/2017/06/26/china-alibaba-jack-ma-retail-ecommerce-e-commerce-new/.

Moreau, Page, Aradhna Krishna, and Bari Harlam. "The manufacturer-retailer-consumer triad: Differing perceptions regarding price promotions." *Journal of Retailing* 77.4 (2002): 547-569.

Papatla, Purushottam, and Lakshman Krishnamurthi. "Measuring the dynamic effects of promotions on brand choice." *Journal of Marketing Research* (1996): 20-35.

Rossi, Peter E., and Greg M. Allenby. "A Bayesian approach to estimating household parameters." *Journal of Marketing Research* (1993): 171-182.

Shaffer, Greg, and Z. John Zhang. "Competitive coupon targeting." *Marketing Science* 14.4 (1995): 395-416.

Sweeney, George. "Marketing, price discrimination, and welfare: Comment." *Southern Economic Journal* (1984): 892-899.

Taylor, Gail Ayala. "Coupon response in services." *Journal of Retailing* 77.1 (2001): 139-151.

Zhang, W. (2015). Behind China's runaway online-to-offline commerce. *McKinsey Quarterly* [online]. Available at: https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/behind-chinas-runaway-online-to-offline-commerce.

Appendix

Link to Dataset：
https://tianchi.aliyun.com/datalab/dataSet.htm?spm=5176.100073.888.30.62b8a501JD1dj8&id=23

Link to our project Github to access code：
https://github.com/91jackcheng/APM_project

Author Information

Name	***UT eid***	***Email Address***
Daxi (Jack) Cheng	dc43342	jack.chengpku@gmail.com
Jianing (Sandra) Cui	jc84297	jianing.cui@utexas.edu
Christine Mulcahy	cm57789	christine.mulcahy@utexas.edu
Elena Reynolds	ekr428	elenareynolds@utexas.edu
Jianjie (Steven) Zheng	jz23373	jianjiezheng@utexas.edu