CSCI-B 565: Data Mining

Final Report

COUPON PURCHASE PREDICTION

A Kaggle Project.

Submitted by:

Beixian Xiong bxiong@iu.edu

Sairam Rakshith Bhyravabhotla bsairamr@iu.edu

Siqi Hong siqihong@iu.edu

Table of Contents

Objectives & Significance

Goal of the project

The goal of the project is to improve the current Recommender System of Ponpare(a Japan Coupon company) by predicting which coupons each user would purchase on a weekly level, given the following data:

a) A collection of details of each user.
b) A collection of details of each coupon.
c) The transactions within the preceding 51-week period.

Why is it important?

Recommender Systems are widely used in different businesses and are extremely important to improve the quality of online services. Customers want to save money and time, and the companies want to fulfill this customer's desire to be more efficient, more competitive, and less costly and as a result gains more money. Recommender systems do all these for customers and the company.

We're doing this project for Recruit Ponpare, a Japan's leading joint coupon site, offering huge discounts on everything from hot yoga, to gourmet sushi, to a summer concert bonanza. Ponpare's coupons open doors for customers they've only dreamed of stepping through. They can learn difficult to acquire skills, go on unheard of adventures, and dine like (and with) the stars.

Our motivation

We have learned about the basic concepts of recommendation system in class. Since there are so many famous websites such as Netflix, Amazon or Groupon all need a recommendation system for their products and services, this project will help us to get a deeper understanding and design a better recommendation system in practice.Also, the project has a lot of data preprocessing involved which is similar to the real world scenario. For example, we created new

features to satisfy the requirement of XGBoost algorithm as well as trying to improve the predictive power of our recommendation models, among those new features we transformed categorical data into binary representation feature(eg. Transforming different places of users and coupons to binary feature *is_large_area*).

Background

All important concepts & Background information

There are mainly three approaches in recommendation systems:

- **Collaborative Filtering**: Collaborative filtering is based on user behavior. It can be based on single user behavior or, more widely used and more effective one, based on group knowledge. By using different distance or clustering method we can group users and make predictions based on majority rating within groups. Usually we calculate different distances or use k-nearest neighbors to group users, some use k-means clustering or hierarchical clustering but as reported in a paper these approaches have not been widely exploited for neighbor selection.
- **Content-based Filtering:** Another approach adopted in recommendation systems is content-based filtering,which is based on user's behavior on websites like historical browsing information. Amazon is one of the companies using this approach in their recommendation system.
- **Hybrids:** The third approach is a hybrid version of previous two approaches.This approach can also be used to address the cold start problem in recommendation systems.

In our project we used the hybrids method by taking both coupon features and user behavior features into model building.

The method we used in this research is XGBoost, which is a powerful algorithm in kaggle community.

XGBoost: known as extreme gradient boosting, which utilizes multi-thread parallel computing, has higher accuracy, and supports sparse matrix. It enjoys great popularity in Kaggle

competition since it usually outperforms other algorithms. Thus, we would like to learn more about this algorithm through doing our project.

To start with, tree ensembles combines multiple weak classifiers to improve the model accuracy, and each time it iterates it will generate a new tree, which puts more weight on the instances that are wrongly classified.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F$$

What we want to learn here is f, which are trees. Tree ensemble is invariant to scaling of inputs, so we do not need to do careful features normalization. Also, it gains the benefits of decision tree, such as selecting features inherently, and the model is easy to be visualized and explained. To reasonably add a new tree each time, we then introduce the concept of gradient boosting. Gradient boosting utilizes gradient descent, which bases on all the other trees that we previously generated, and will take a step towards the minimization of the given target (loss) function. The usual target function for one model includes both training loss that measures how well the model fits data and regularization that controls the model complexity, which is a tradeoff between bias and variance:

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

Here, with tree ensemble, our target function thus becomes:

$$Obj(\Theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

and will yield a common gradient boosted machine. What we want to learn here is an f to optimize the objective function. However, gradient boosting tree cannot be run parallel, and thus we have XGBoost. xgboost is a tool for massively parallel boosted tree. Currently, it is the fastest and best implementation of boosted tree.

Previous work on this problem

Overall, the project is a cold start recommendation system. Unlike general recommendation problem, the recommended items are not shown in the training set. The basic solution is to use hybrid methods. There are some more other modified approaches applied previously by participants:

1. A modification of the cosine similarity.
2. A hybrid matrix factorisation approach .
3. TLDR: Learning the score function using three-layer neural networks.
4. Do the dot product of the user profile and the item profile and compute the cosine distance between the two.
5. Coupon to User, large matrix. Coupon to some shared weighted subset, smaller matrix. Find similarity on that smaller matrix. (or get more advanced with weight layer(s) and go RBM)

How is our solution more interesting than previous solutions?

1. *Profiling for coupons and customers*. Profiling data means statistically analyzing the data to assess the values in the data. It is used to find any relationship between already existing columns. For example , in the given data set, four of the columns we have are latitude and longitude of the user location and latitude and longitude of the coupon location(the place where a particular coupon is valid). We can combine these four into a single column namely user-coupon distance. Various attributes to be used to create feature for customers are calculated using the existing data. An example of such features which are created in our project are:
    a. Binary features judging if a coupon belongs to top 3 often visited/purchased genres of certain user
    b. Whether a user is more interested in viewing/buying coupons listed on weekdays/weekends

The above features are created with the purpose of profiling every user behavior, by first combining data tables all together and then build hash tables to create these features.

2. Multithread the clusters by using XGBoost to reduce the time taken.

Methods

Description of the Data.

The datasets contain a year of transactional data for 22,873 users on the site ponpare.jp. The training set spans the dates 2011-07-01 to 2012-06-23. The test set spans the week after the end of the training set, 2012-06-24 to 2012-06-30. The goal of the competition is to recommend a ranked list of coupons for each user in the dataset (found in user_list.csv).

- **user_list.csv** - the master list of users in the dataset
- **coupon_list_train.csv** - the master list of coupons which are considered part of the training set.
- **coupon_list_test.csv** - The master list of coupons which are considered part of the test set. Your competition predictions should be sourced only from these 310 coupons. You will not receive credit for predicting training set coupons that were purchased during the test set period.
- **coupon_visit_train.csv** - the viewing log of users browsing coupons during the training set time period. You are not provided this table for the test set period.
- **coupon_detail_train.csv** - The purchase log of users buying coupons during the training set time period. You are not provided this table for the test set period.
- **coupon_area_train.csv** - The coupon listing area for the training set coupons.
- **coupon_area_test.csv** - The coupon listing area for the test set coupons.
- **sample_submission.csv** - A sample file showing the correct format for predictions.
- **documentation.zip** - An archive of Excel files containing an entity relationship diagram and English translations.

Methodology & Implementation

1. *Before beginning the process :* As we can see, the data roughly has 3 million instances with 45 features. To run algorithms on this data in a normal computer is a nightmare. So, we have used Amazon Web Services(AWS) , BigRed2 to run the program . The EC2 service from the AWS has been set up using the amazon command line interface and has been used. Also, a torque script has been written to run the code on BigRed2. For the details of how AWS has been set up, please watch the video created by our team. The link to the video is given in Appendix.

   As AWS services has allotted us limited memory (1 GB) , we had to run the main XG Boost algorithm on BigRed2. A torque script (attached with the folder) has to be created to run the program as a batch job.
2. *Data Cleaning* : All the data cleaning part was done in MongoDb which is good in handling large databases. The data obtained was not a very clean data . Firstly, part of the data was in Japanese which need to encode with utf-8 format. Secondly, there are a lot of missing values in the data . Infact, the data obtained was 99.7% sparse as given in the data description.It means 99.7% of the data is filled with 0's or missing values. The data was not only filled with 0's but also with NA's in lot of places. The NAs had to be replaced with 1's. Though it may seem like a dangerous step, most of the NAs observed are *usable_** features in which majority of them are already 1s . It has been tested that effect of changing them to 1 has been very small.

   Another bold decision taken is to combine all the tables into 1 big table and normalize it.Combining all the tables helped not only for faster processing of XGBoost Algorithm but also for more coherent understanding and convenient feature creation of data.
3. *Data Preprocessing*: This has been the greatest challenge in the project. The following are the two important steps in data preprocessing . Several features have been created from the existing features to make more meaning from the features as well as reducing the dimension of data(eg. transforming four coordinates features into one distance feature). The following are the features that have been created after considerable amount of Exploratory Data Analysis (to be covered in the next section).
   a. The columns user latitude, coupon latitude, user longitude, coupon latitude have been combined to get the Geographical Distance between User's Location and Coupon's location. Haversine distance has been used to calculate the distance. It

is given as :

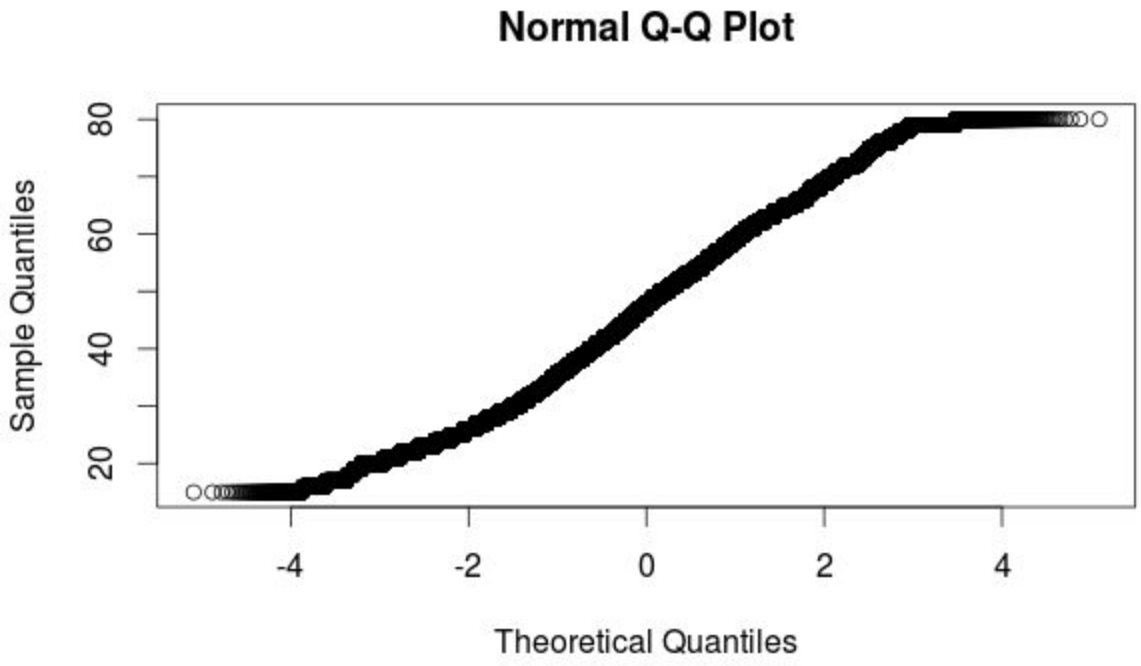
$$d = R * 2 * a * tan2(\sqrt{a}, \sqrt{a-1}),$$

where R=6371 kms is earth's radius and

$$a = sin^2(\Delta\varphi/2) + cos\varphi_1 * cos\varphi_2 * sin^2(\Delta\lambda/2)$$

where φ_1, φ_2 *are the latitudes in radians and* λ_1, λ_2 *are longitudes*

b. The coupons purchased given the age of the customers followed a normal distribution . This has been observed by plotting a Q-Q plot. Below is the Q-Q plot for the age. So, it can be a wise decision to convert the age variable into 1's(above the mean/median value) and 0's(below the mean/median value). This would be faster and at the same time has minimal effect on the accuracy.



c. From the exploratory data analysis, we have understood that purchase of coupons were largely dependent on the basis of large area or small area. So ,

the area has been converted into a binary feature i.e. large area id given 1 and small area is given 0. Our justification of doing this comes from population distribution given on wiki website, where we identified the prefectures with top 3 population(i.e. Tokyo-to,Kanagawa Ken and Osaka-fu), identified these three areas as large area, then categorize the rest of the prefectures as small areas.Same trend is also discovered when we do exploratory analysis, as can be seen later in Exploratory Data Analysis section.

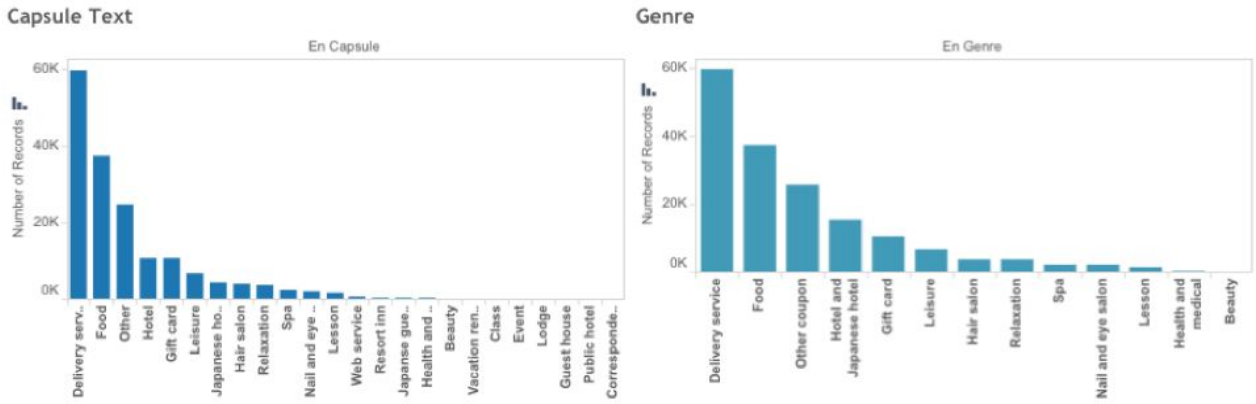
d. Also, there are several *usable_** features, like *usable_monday*, *usable_tuesday* etc. which indicate the usability of coupons on all days of the week , holidays and day before holidays. All these features have been combined to make the following features:
    i. *Usable_weekday*
    ii. *Usable_weekend*
    iii. *Usable_holiday*
    iv. *Usable_dayBefore_Holiday*

    These features represent a better indication of the trends and reduces the dimensionality.
e. The next feature *valid_period* created is by combining the three features *Disp_from*, *Disp_end*, *Reg_date*. *Disp_from* is the date from which the coupon is displayed from. *Disp_end* is the date from which coupon has been stopped from being displayed. *Reg_date* represents the registration date of the user. If *Reg_date* is after the *disp_end*, it means there is no chance the user buys the coupon. So, a value 0 is assigned in this case. In any other case, value 1 is assigned. Thus, these three features were deleted after the new one was created reducing the dimensionality to further extent.
f. Apart from these features, the next feature that is created was *top_genre_purchased* , which is a binary feature set to 1 if the purchased coupon belongs to top 3 genre of certain user.
g. A similar feature is *top_genre_visit* , set to 1 if visited coupon belongs to the top 3 visited genre by the user.
h. Finally, the last feature is *is_male*. Set to 1 if is male and 0 if female. This is transformed from the original gender feature, which has encoding ‘f’ for female

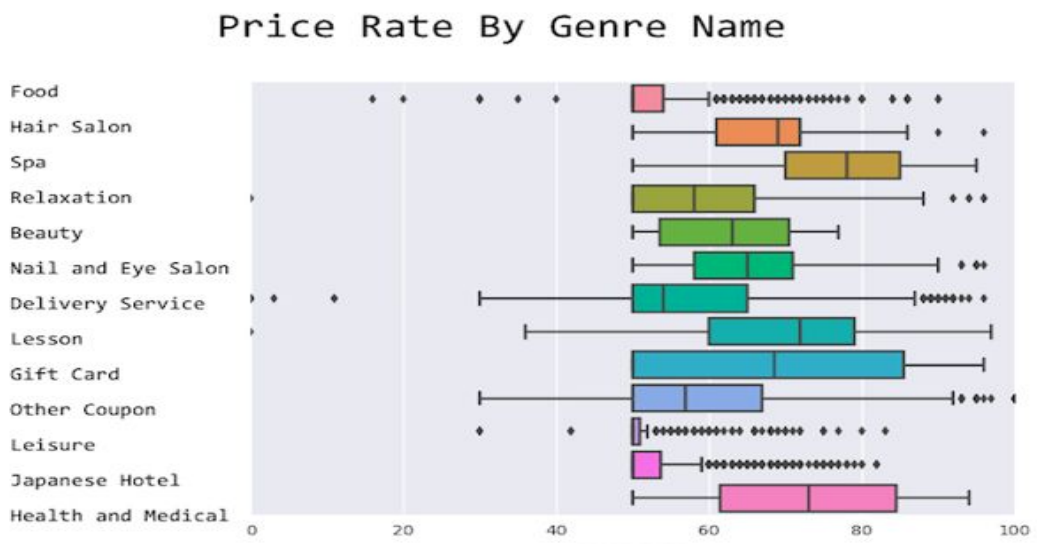
and ‘m’ for male. We transformed this to meet the numerical feature requirement of XGBoost.

Also the main idea of creating binary features is that XGBoost works best on binary numerical classifiers.

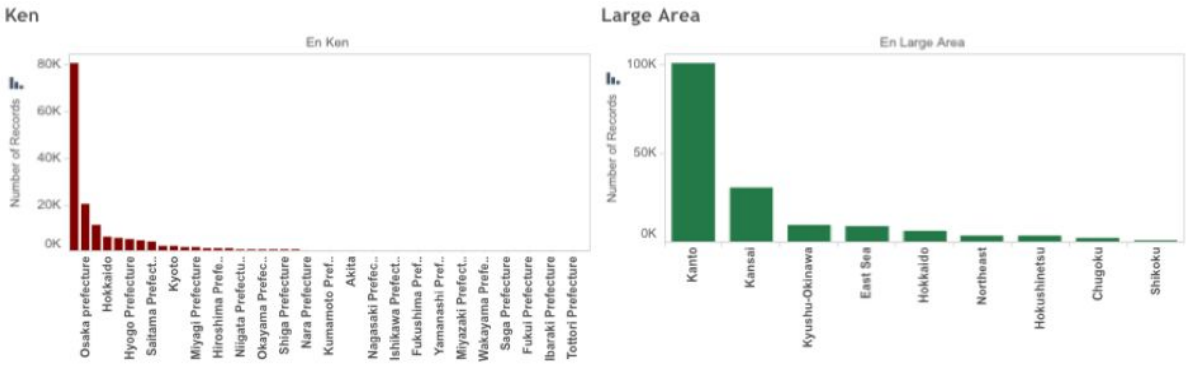
4. *Exploratory Data Analysis*:
    a. Firstly, we have checked how the coupons are distributed by capsule, genre, ken, and large area, and we found that most of the coupons belong to delivery service, food and other.

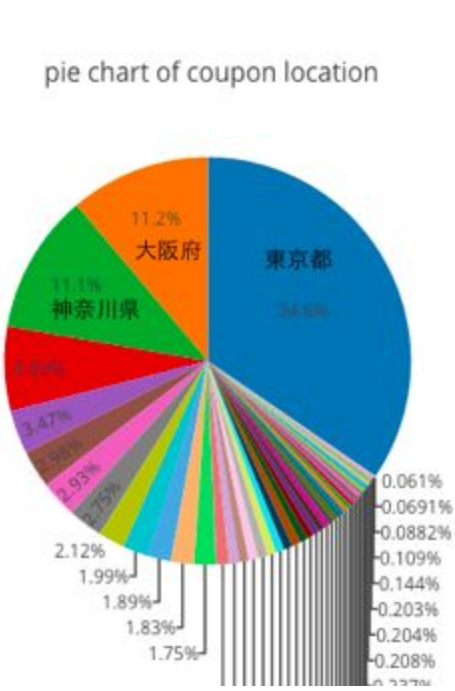


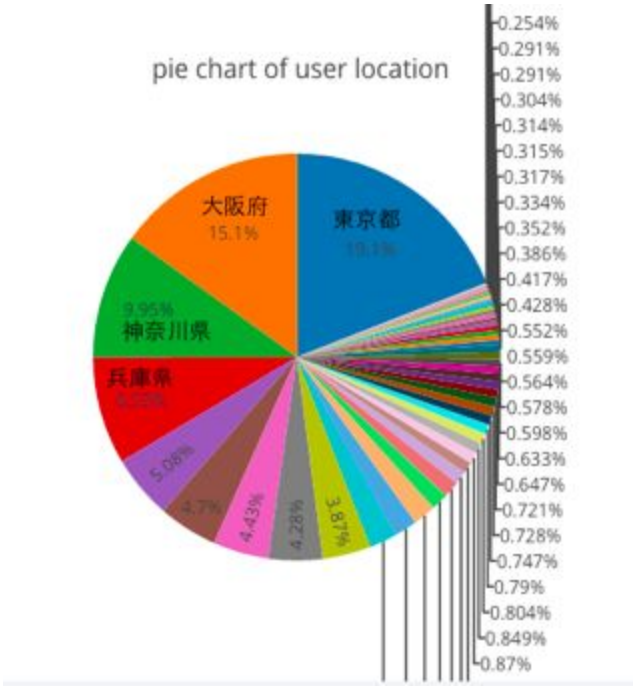
And the price rate (discount rate) for these three major genres stays around 50 to 70, with not too much variation despite some outliers.



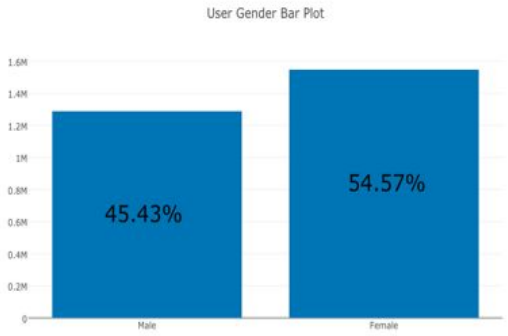
b. Also, we find that the coupons are mostly distributed in Tokyo, Osaka, and Kanagawa when considering Ken (small areas, similar to Cities), and Kanto and Kansai when considering Large Areas (similar to States). The distribution of coupons is similar to the distribution of users, which make sense since demand and supply are usually relevant.

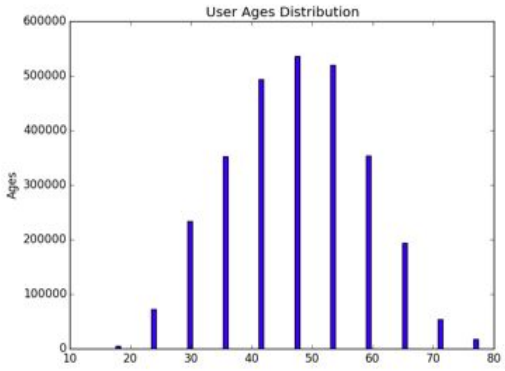






c. After these, we have looked more into the users. There are 45.43% users are males, and 54.57% users are females, and their ages are mostly among 35 and 60.

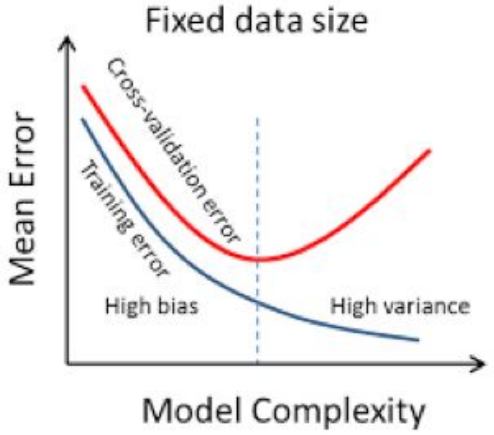




5. *XGBoost Algorithm*:

a. **Parameter Tuning:**Initially, we fixed the learning rate to 0.05, and max tree depth from 4 to 10 to tune the number of trees using 5 fold cross validation. Number of tree is usually dependent on the data size. Since our data is very large, we have tried number of trees up till 130, The plots for the parameter tuning procedures are given in the results .

   We can see that when depth=6 to 8 with number of trees around 120, the results are the best. The training errors will definitely decreases, the test errors however becomes more stable, it looks similar to the following graph:

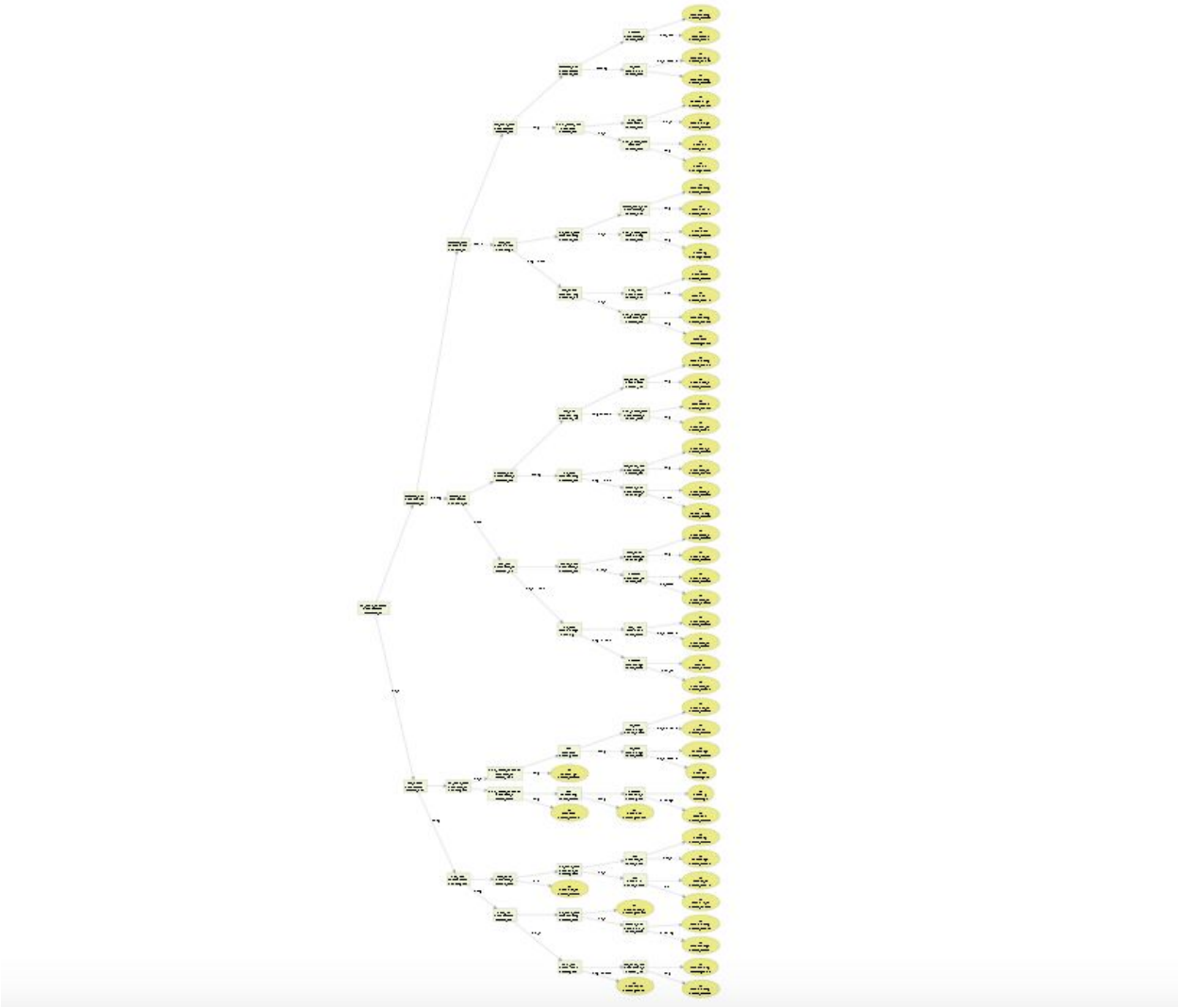


Where the bowl shape depression part is the place to have best trade-off between bias and variance. Thus, we have chosen learning rate=0.05, depth=6, and trees=120 to get our best model.

b. **Feature importance and plot of tree:** It is not surprising to see top_genere purchase, discount prices, top genere visit, catalog prices, and price rates are the top most important features to decide whether the user will buy the coupon. But it is interesting to see that registration date affects the response as well and is the 6th most important feature. Also, how long the coupon is valid, and the distance between coupon location and user location are the next most significant features. This should provide some insights to the coupon companies and make decision consequently to improve their businesses.

The table of feature importance is shown in the results.

As for the tree plot, the overall tree is too large, so we just take a screenshot of the first tree here.



Evaluation Strategy

The evaluation strategy would be Mean Average Precision @ 10 (MAP@10):

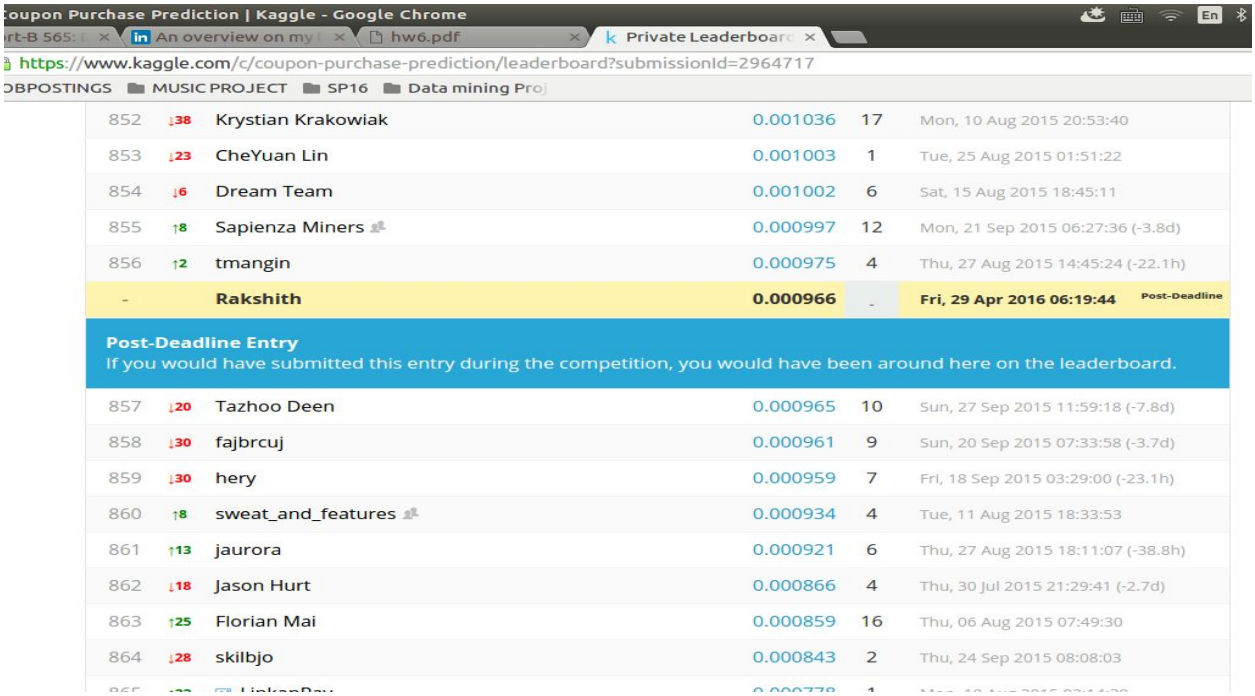
$$MAP@10 = \frac{1}{|U|}\sum_{u=1}^{|U|}\frac{1}{min(m,10)}\sum_{k=1}^{min(n,10)}P(k)$$

$|U|$ *is the number of users*; $P(k)$ *is the precision at cutoff* k, n *is the number of predicted coupons and* m *is the number of purchased coupons for the given user*($m = 0$ *if the precision is defined to be* 0).

MAP is a widely used evaluation metrics in recommendation system. MAP @10 in this case, calculates the precision when we want to recommend 10 coupons to each user by rank positions, and on average how precise our recommendation is. It makes sense since what really matters is how many good results for the top recommendation here in the first 10 recommendation.

Results

Our MAP@10 is 0.000966, which ranks 856 among all competitors.However, we could achieve a computation time of 5-6 hours as expectations, compared to the runtime of other competitors which was around 10-12 hours. We have tuned the parameters several times, but did not get better results. It is possibly because of our data transformation. Although we have created some new features that we believe are more informative than the original features, we have also deleted some features we think that are not important. This may lead to the reduce of important information that helps distinct the classes.Here is the screenshot of the final result image on kaggle: (with depth = 6 and 100 cross validations)



Pathway to achieve this result : The first time we submitted, we got a result of MAP@10 : 0.0004 and stood 924th in the rankings. In this case we have used several non-binary classifiers. The *age* feature was kept as it is without converting it into a binary feature. Also, the 10 *usable_** features weren't reduced to 3 features. The NA's were kept as it is without converting it into 0's or 1's .

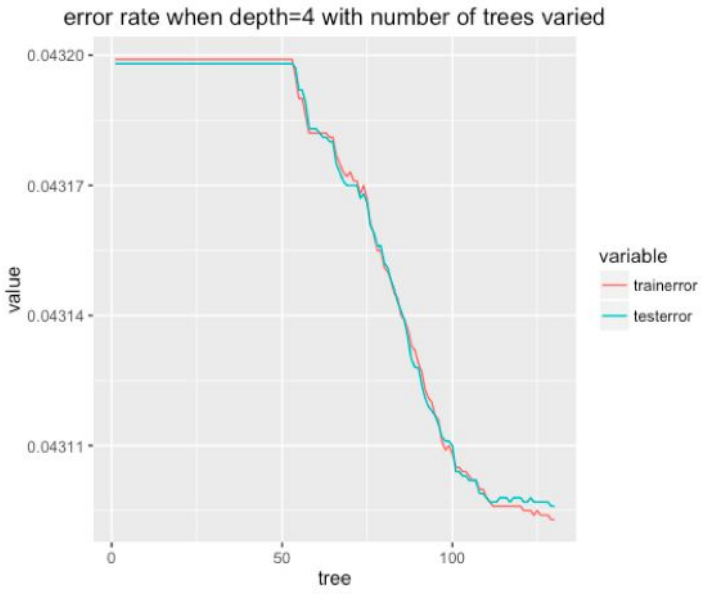
After repeated trials, we have converted the NA's to 0's , reduced *usable_** features to just three features as mentioned in the Methods and Implementation Section. Our MAP@10 drastically improved and got doubled to 0.0008 ranking 864th.
Then, after creating the feature *is_valid_range* and changing NAs to 1's instead of 0 our result got improved to 0.00096 standing 856th.

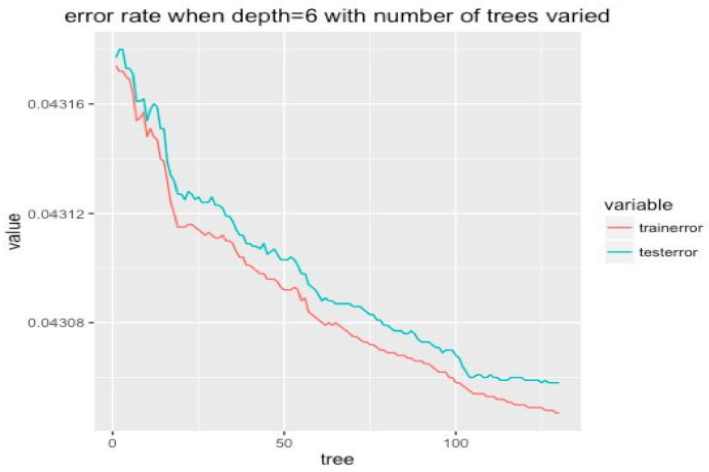
The first time we ran the code without much preprocessing, the overall time taken was more than 10 hours. After repeated optimizations of code as well as features by reducing the dimensionality , our run time ended up to 4.5 hours which is worth mentioning.

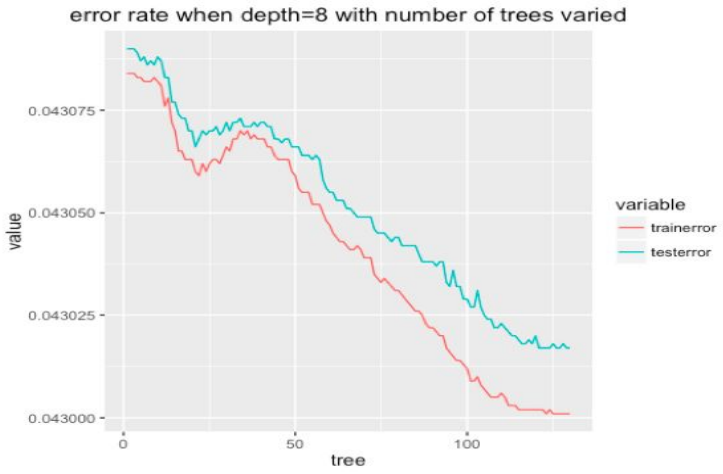
There can be several modifications that can still be done but given the time frame and computational limitations, we had to keep them for the future work mentioned in the conclusion.

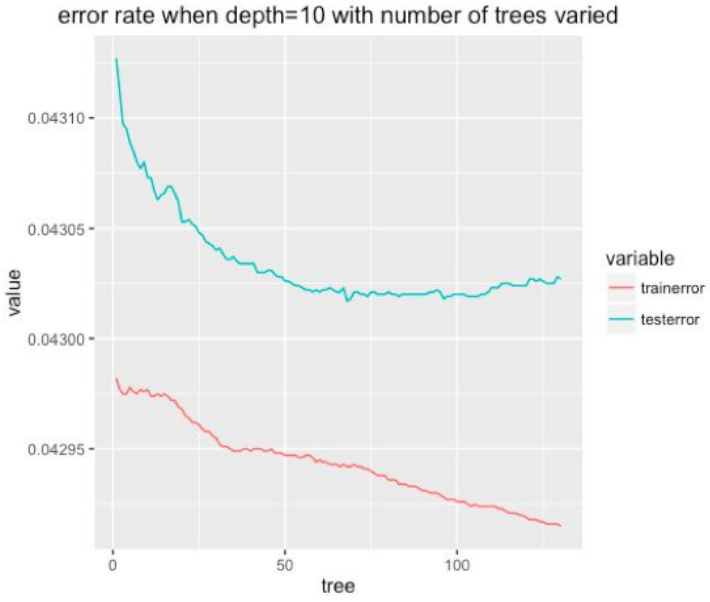
Below are the outputs for various experiments we did:

1. Parameter Tuning.






error rate when depth=8 with number of trees varied
value
0.043075
0.043050
0.043025
0.043000
0
50
100
tree
variable
trainerror
testerror


error rate when depth=10 with number of trees varied
value
0.04310
0.04305
0.04300
0.04295
0
50
100
tree
variable
trainerror
testerror

2. The following is the table for feature importance.

	Feature	Gain	Cover	Frequency
1:	top_genre_purchase	5.474780e-01	1.716089e-01	0.020161290
2:	discount_price	2.137549e-01	2.304237e-01	0.093750000
3:	top_genre_visit	4.086894e-02	9.134408e-02	0.083165323
4:	catalog_price	3.570212e-02	3.715241e-02	0.068044355
5:	price_rate	3.278517e-02	1.045307e-01	0.072748656
6:	reg_date	2.169171e-02	5.706120e-02	0.104334677
7:	disp_period	1.780314e-02	6.084973e-02	0.053427419
8:	valid_from	1.246983e-02	3.545398e-02	0.040994624
9:	valid_period	9.961764e-03	1.774053e-02	0.055275538
10:	distance	9.564739e-03	2.641742e-02	0.060147849
11:	disp_end	9.048083e-03	1.502514e-02	0.036794355
12:	valid_end	7.657141e-03	4.026749e-02	0.030745968
13:	prefer_wkday_visit	7.150381e-03	1.197285e-02	0.030241935
14:	usable_sat	6.978153e-03	2.090316e-02	0.021505376
15:	prefer_wkend_visit	5.470510e-03	2.044206e-02	0.021169355
16:	prefer_wkday_purchase	5.134529e-03	1.008260e-02	0.070396505
17:	age	4.294000e-03	1.183873e-02	0.060651882
18:	disp_from	3.262639e-03	6.948185e-03	0.016801075
19:	coupon_large_area	3.217544e-03	1.005229e-02	0.015456989
20:	usable_sun	2.056283e-03	9.459286e-03	0.006888441
21:	withdraw_date	1.390395e-03	2.000490e-03	0.005544355
22:	usable_date_before_holiday	7.180915e-04	4.769402e-03	0.001848118
23:	usable_fri	4.761541e-04	2.913831e-03	0.003360215
24:	prefer_wkend_purchase	3.852438e-04	1.861753e-04	0.015288978
25:	user_large_area	2.940935e-04	1.997252e-04	0.004704301
26:	usable_mon	1.917851e-04	1.842837e-04	0.002184140
27:	usable_thu	9.571574e-05	1.073007e-04	0.001176075
28:	usable_holiday	7.105470e-05	2.416972e-05	0.001512097
29:	usable_tue	2.794279e-05	4.017983e-05	0.001680108

It is not very surprising to see that the top 5 features are top_genre_purchase, discount_price, top_genere_visit, catalog_price, and price_rate. These make sense since people tend to purchase products that they mostly viewed and purchased, and when prices are reasonable. But it is interesting to see that registration date comes next, which shows that people who have started to purchased coupons on the website earlier are easier to purchase coupons. So besides focusing on the most viewed and purchased coupon genre, and setting a reasonable prices discount, the company can also target more on the customers who register earlier. How long the coupon is valid, and the distance between the user and the coupon is another factors that have important influence on customers' purchase decision.

Conclusion

The project did not give the expected results. However, it was not a complete failure as well. Despite of the unclean data with around 3 million instances and more than 40 features, in addition to it's extremely sparse nature(99.3%) , we could achieve to clean the data, tune the

parameters several times, conduct a strong Exploratory Data Analysis and get the results within the limited time frame for a 4 month Kaggle Project. There might be several reasons for results ending up below expectations. Despite tuning the parameters several times,we did not get better results possibly because of our data transformation. Although we have created some new features that we believe are more informative than the original features, we have also delete some features we think that are not important. This might have lead to the reduction of important information that helps differentiate the classes.

Future Work: Also when we are creating new feature there's room for further tuning, for example, instead of creating Top 3 visited/purchased genre, we can expand to Top 5 or shrink to Top 1, these give room for future improvement of our project performance.
And instead of using only XGBoost, we can also use Support Vector Machines(SVM) to train our model because SVM is good at dealing with sparse matrix.
Also, probabilistic modelling of the data can be done.

$$P(purchase) = P(user\ online)\, P(visit|online)\, P(purchase|visit).$$

P(user online) is a Beta-Bernoulli distribution.A Beta-bernoulli distribution is a bernoulli distribution in which the probability of success is taken randomly from a beta distribution instead of being taken as ½ . It gives weights to coupons based on how many days they were displayed.
Also, exploring more sophisticated features by giving weights to each of the existing features can be a possible option. For example, using perceptrons by assigning weights to current features can be a possible explorations. Finally, use of Neural networks and deep learning techniques can be explored.

Individual Tasks

During the final project everyone has their own specific task but we also cooperate with each other. In the very beginning we look at literatures and forums and slides from industry together to have ideas what we're going to do later. Then I generated combined data for them to do exploratory and feature creation, but we all take part in feature discussion as to what kind of

features we need to create to reduce dimension and also improve power of our models. Then Siqi did exploratory data analysis so we have a better understanding of data distribution. Rakshith and I came up with useful features to profile user data, he also wrote functions to do NA value and date format transformation. I cooperate with him in transformation by combining his functions with MongoDB to prepare transformed data ready to feed XGBoost. Siqi learned the algorithm and developed in R, while Rakshith helped set up AWS and BigRed2 environment to train model using training data, I at the same developed code to generate test data set for them to test results. Siqi also tuned the parameters for XGBoost for us to test result later. We all participated in the result testing. As for project management, we all acted as leaders of the group, everyone tried to organize the meeting process, our task distribution and time management. To manage tasks we used JIRA. We used google cloud services like google doc,google drive to manage and update files.Also we used whatsapp to communicate with each other effectively.

My individual task in detail was:

1. Combining coupon list table, coupon detail(purchase) table, prefecture location table, user list table and coupon visit table. Using MongoDB and python together, and use indexing to speed up the query. The code to generate combined.csv data file is cleaning.py.
2. After discussion with Siqi and Rakshith, I implemented data preprocessing and transformation to transform NA and empty values, formatted date into numbers, generated distance feature, and generated user profiling features(*top_genre_purchased*,*top_genre_visited*,*prefer_wkday_visit*,*prefer_wkday_purchase* etc.). The output of this is data_x.csv data file, generated by transform.py, transform2-delete_visit_purchase.py,transform2-transform_gender.py and delete_id.py
3. Create test data by matching test user ids with all coupons we have. The code is create_test_data.py

References

1. https://www.kaggle.com/c/coupon-purchase-prediction/forums/t/16789/approach-sharing
2. https://www.kaggle.com/c/coupon-purchase-prediction/forums/t/16450/anyone-approaching-this-as-a-classification-problem
3. http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

4. http://www.ibm.com/developerworks/opensource/library/os-recommender1/index.html
5. http://ir.ii.uam.es/pubs/recsys12-bellogin.pdf
6. http://jmlr.csail.mit.edu/proceedings/papers/v32/titsias14.pdf
7. http://www.machinelearning.org/proceedings/icml2007/papers/407.pdf
8. https://www.quora.com/How-do-I-setup-a-Python-script-to-run-on-an-AWS-EC2-instance
9. https://www.linkedin.com/pulse/overview-my-first-kaggle-competition-coupon-purchase-saeed-mirshekari
10. https://en.wikipedia.org/wiki/List_of_Japanese_prefectures_by_area
11. https://docs.mongodb.org/manual/tutorial/

Acknowledgements

"**This research was supported in part by the Indiana Genomics Initiative. The Indiana Genomics Initiative of Indiana University is supported in part by Lilly Endowment, Inc.**"

We thank Prof. Predrag Radivojac and the AIs for their continuous support throughout the semester. So much of learning would not have been possible without them.

Appendix

All the files mentioned below are attached in the following google drive link.
https://goo.gl/UhZIf9

1. For the video we created on how to setup AWS and python, **https://goo.gl/PGVG4e**

2. **R_job** : the torque Script used to run programs on BigRed2.

3. **apr24train.R** : the code we used to implement XGBoost Algorithm

4. **apr24Explore.R:** the code used to explore the trained model

5. **apr24Test.R:** The code we used to test the model

6. **format.R** : code to convert output into required format as per Kaggle requirements

7. **MongoDB data preprocess.pdf**: Documentation on Setting up MongoDb and Pymongo, also basic description on data preprocessing.

8. **Cleaning.py :** Combine coupon list table, coupon detail(purchase) table, prefecture location table, user list table and coupon visit table. Output: combined.csv

9. **Transform.py:** transform NA and empty values, format date into numbers, generate distance feature, and user profiling features. Output: transformed_new.csv

10. **Transform2-delete_vist_purchase.py:** delete capsule_text, PAGE_SERIAL, purchase amount,I_DATE and bought_date. Output: tranformed_new2.csv

11. **Transform2-transform_gender,delete_id.py:** transform gender into is_male 0-1 representation. Output: data_x.csv

12. **Create_test_data.py:** match testing user ids with all coupons in training data, along with relevant coupon and user features from transformed table. Output: test_ids.csv, test_x.csv,test_wid.csv

13. **Test_instructions.docs :** Readme File for apr24test.R and apr24explore.R

14. **Train_instruction.docs:** Readme file for apr24train.R

15. Submitzero.csv: output submitted to the Kaggle community.

16. **myCode.R :** Transforming the data to improve the accuracy. Data_x.csv is converted to newdata.csv & test_x.csv is converted to newTest.csv

17. **Siqisample.zip**: sample data and test files included.

18. **Newdata.csv , newtest.csv :** latest and final training and test data