IE 582 Project Report

Group 4

Sualp Say (2020702141) - Samet Öztürk(2021702126)

January 22, 2022

Instructor: Mustafa G. Baydoğan

1 Introduction

The aim of the project is to construct a model that predicts the gender of a customer using clickstream data. Sample training data is provided to work on.

1.1 Descriptive Analysis

Train data is in unstructured format. Before using predictive models, data manipulation and feature engineering is needed to get structured data with features to train. Details about features will be discussed in following sections.

Train data has 5,493,268 rows and 19 columns. However, it included a lot of duplicate observations where the unique row count is 2,077,356. We removed rows causing duplication to avoid miscalculations. The data included actions of 5918 users 3679 of which are female, 1939 of which are male. Hence, we needed to deal with class imbalance, as well. Five actions we have are adding a product to basket, adding a product to favorites, ordering a product, searching a product and visiting a product. According to timestamp information, the actions on the data took place between October 14, 2020 and December 9, 2020. Selling prices of the products are ranging from 0 to 226,320 units where we have 32,013 missing values. Other than selling price, all other columns have character, factor or time information, which are not numerical. Other columns mostly have descriptive information related to the product. We have, 3 level category information, business unit information, brand information and product name. For these columns, we have 2184 missing values. Product gender columns provide 3 levels about the gender of the product: female, male and unisex. It has 257,333 missing values but might be beneficial to create predictors.

2 Approach

Since the data is unstructured, we first needed a transformation to get structured data. After removing duplicates, we proceeded with feature engineering. Then, we needed to come up with a method to deal with class imbalance. We created a function calculating the custom accuracy metric in the project description to compare models and parameter settings during training. To avoid train bias, we used cross validation. We also tried scaling numerical features to see if we can have improvement. Finally we trained multiple models and compared them to select the best model.

2.1 Feature Engineering

We used different set of features leveraging various data types in order to find best ones that can project the variability between genders. After idea generation process, we came up with 9 main bullet points which are as follows:

- Favorite Brand Gender based on Actions
- Favorite Category Gender based on Actions
- Favorite Product Gender based on Actions
- Occurrence Days of Actions
- Occurrence Times of Actions
- Monetary Value
- Basic Action Counts
- Action Counts based on Customer Profile
- Session Time

2.1.1 Favorite Brand Gender based on Actions

Basically, we tried to detect the favorite brands' genders of each customer based on frequent actions. So this module consists of two parts. First, we identified favorite brand and then we assign a gender to this brand using count of product genders under each brand. Final results obtained by majority voting technique. As a default value we used "Unisex" which means if we could not detect a dominant gender, we flagged that brand as "Unisex".

2.1.2 Favorite Category Gender based on Actions

Likewise, this time we applied the same idea to identify the favorite category of customers and gender of those categories.

2.1.3 Favorite Product Gender based on Actions

We were given the product gender column in dataset. By analyzing this column and action types, we tried to find out the which gender of products customers are more reacting or taking action. We thought that this may give an idea about the gender of the customer, assuming that female customers would take more action on "Unisex" or "Kadin" products. We applied the algorithm as above to find out favorite product gender.

2.1.4 Occurrence Days of Actions

This time, we divided the days into two factors: "weekdays" and "weekend" and tried to detect which customers are taking more actions at what period of the week. If we were not able to detect a dominant period we assigned as "Unknown".

2.1.5 Occurrence Times of Actions

Similar to approach above, we divided the daytime into 4 factors: "night", "morning", "afternoon", "evening" and identify the preferences of each customer.

2.1.6 Monetary Value

We calculated different monetary values using 3 actions. For each customer, average selling price of products they favorited, ordered and added to basket calculated. Then we also added extra feature as average price of products they took any action.

2.1.7 Basic Action Counts

Besides all above, we counted all recorded actions for each customer and used to model genders.

2.1.8 Action Counts based on Customer Profile

By mean customer profile, we refer to customer gender. In these set of features, first we detect the brands, categories, business unit and contents which may be highly related to a specific gender. For each of them we count the male and female actions and calculate a ratio. Then based on a threshold value, we flagged those as male or female related things and count the action taken by each customer. At the end, we used these features in numeric type. For example, percent of female actions is nearly 96% in business unit “Makyaj” which means we can somehow relate next person’s gender based on the value his/her actions in specific business unit.

2.1.9 Session Time

We hypothesized that there is a difference in average time spent on website between genders. So, we calculated each customers session time and took the average of it.

Using these ideas and algorithms, we ended up with 13 categorical, 17 numerical, in total 30 different variables to explain and model gender relations.

2.2 Class Imbalance

Around 65% of the customers are from one class where 35% are from the other. The classes don’t have the same size. Thus, we need to use a sampling method to balance the classes. We decided to use over-sampling the minority class. In caret package, setting sampling parameter to ”up” we over-sampled ”Male” class with replacement.

2.3 Cross Validation

To avoid train bias during training and model selection we used 10-fold cross validation with 5 repetitions. So for each model (an algorithm and a parameter set), we divided the data into 10 folds, randomly. Using 9 of 10 folds, we trained the model and calculated the metrics on the other fold that is not used in training. Using each fold as test data, we trained 10 models and finished 1 of 5 repetitions. In summary, we trained 50 times to evaluate each

model. Setting parameters of caret package, method as "repeatedcv", number as 10 and repeats as 5, we applied 10-fold cross validation with 5 repetitions.

2.4 Performance Metric

Performance of the models are announced to be calculated as average of area under ROC curve(AUC) and balanced accuracy rate. Balanced accuracy rate is the average of sensitivity and specificity. So the performance metric is calcualted as below:

$$Performance = 0.5 * AUC + 0.25 * Sensitivity + 0.25 * Specificity \quad (1)$$

We created a function calculating this metric and used it to evaluate and compare the performances of our models.

2.5 Scaling

We thought that scale differences between some numerical variables may cause problems for some of the methods. So, we created a version of manipulated train data where the numerical variables are scaled using "preProcValues" function from caret package. We also centered scaled variables to keep them around similar ranges. However, the using scaled version of train data did not provide noticeable improvements.

2.6 Models and Selection

To find the best algorithm and parameter set, we tried CART, Lasso Regression, Elastic-Net Regression, Random Forest, Support Vector Machines, Gradient Boosting Machines and Adaptive Boosting with numerous parameters. After a few iterations for each method, we came up with parameter sets/ranges that the method performs relatively better. As an example, one of the old results for different parameter sets of GBM is shown in Figure 1.

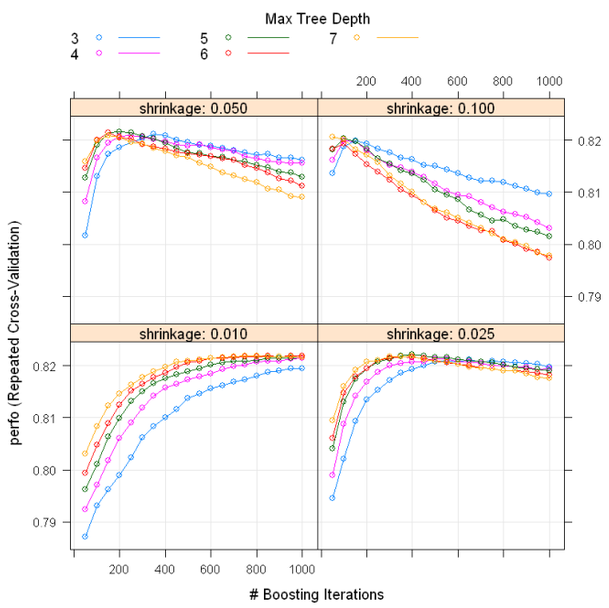

Figure 1: GBM Results for 400 different parameter sets

After selecting best performing parameters for each method, we compared the methods with each other to select the best model.

3 Results

First, we evaluated performances of the different sets of parameters and then we compared methods.

3.1 CART

Comparing runs for 20 different complexity parameter values from 0 to 0.005, equally spaced, the best performing model is obtained with complexity parameter 0.00225 with 80.29% performance. Trying the same parameters with scaled train data, results changed slightly and the models did not improve significantly.

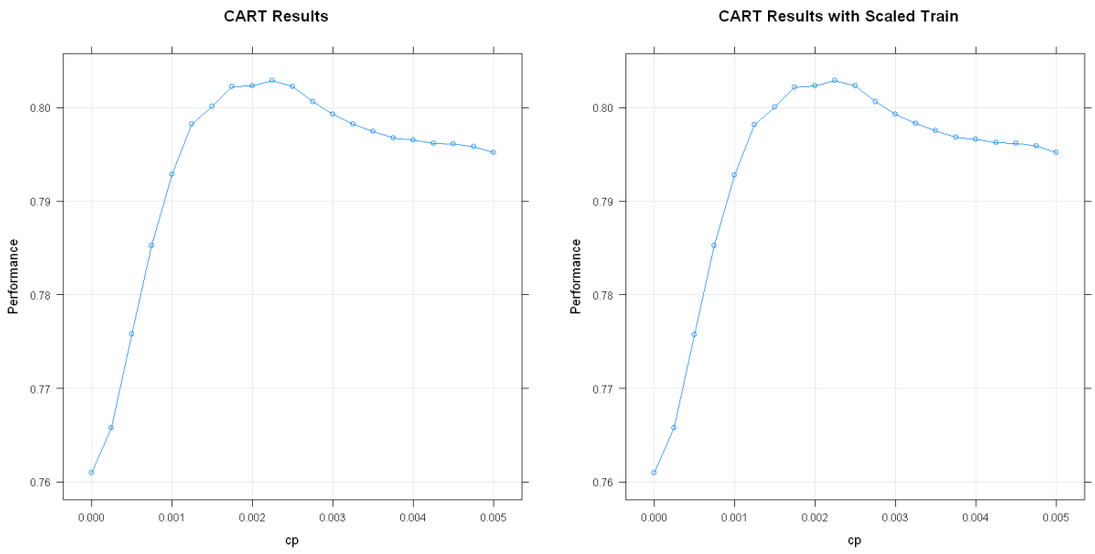

Figure 2: CART Results

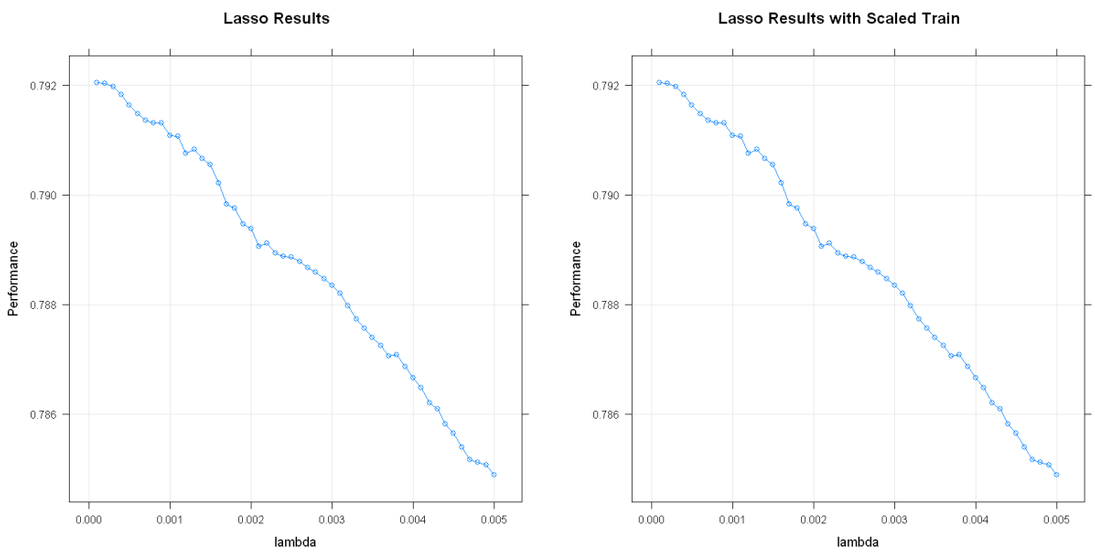

Figure 3: Lasso Regression Results

3.2 Lasso Regression

Comparing results for 50 lambda values from 0.0001 to 0.005, equally spaced, the best performing model is obtained with the smallest lambda, 0.0001 where the performance is 79.20%. So we got better results penalizing less. That might be an indicator that more features might improve the model since best penalization parameter is very small. Trying the same parameters with scaled train data, results changed slightly and the models did not improve significantly. Results for Lasso regression is provided above, in Figure 3.

3.3 ElasticNet Regression

Comparing results for 4 lambda values from 0.001 to 0.05 and 4 alpha values from 0.1 to 0.75, the best performing model is obtained with lambda,0.001 and alpha 0.75. Note that the smallest lambda value is selected, again, along with highest alpha. The model tried to penalize less here, too. The best performance is 79.06%. Results are shown in Figure 4.

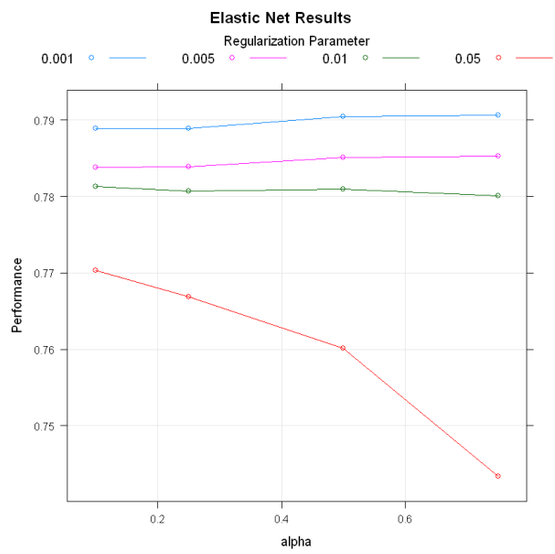

Figure 4: ElasticNet Regression Results

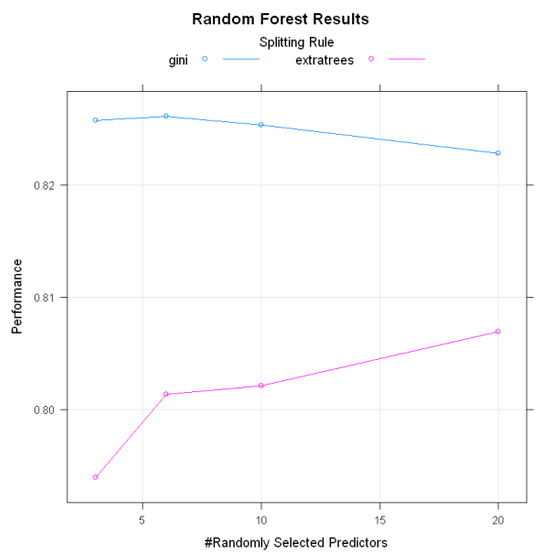

Figure 5: Random Forest Results

3.4 Random Forest

Among runs for "Gini" and "Extratrees" splitting rules and 4 different variable selection number parameters, the best model was the one with 6 variables and "Gini" rule. Here, performance was 82.61%. Using scaled training data on best performing parameter sets did not improve results. Hence we decided not to use the scaled version in following runs to save time. Results for random forest are illustrated in Figure 5 above.

3.5 Support Vector Machines

We used SVM with radial kernel for 4 different cost and 3 different sigma values. Best results were obtained where sigma was 0.01 and cost was 1. The performance was 78.37%. Results for different parameters are provided in Figure 6 below.

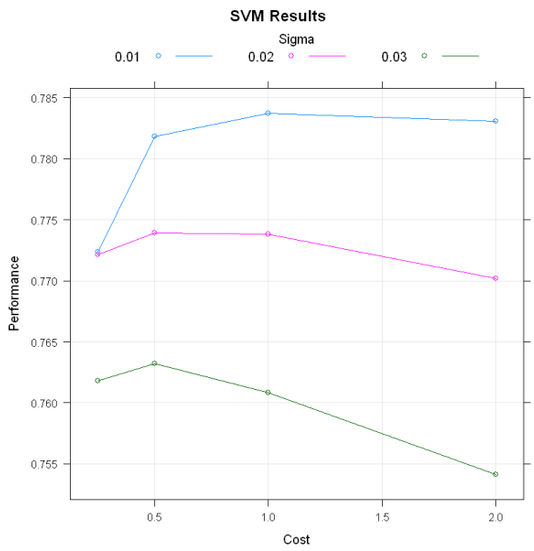

Figure 6: SVM Results

3.6 Gradient Boosting Machines

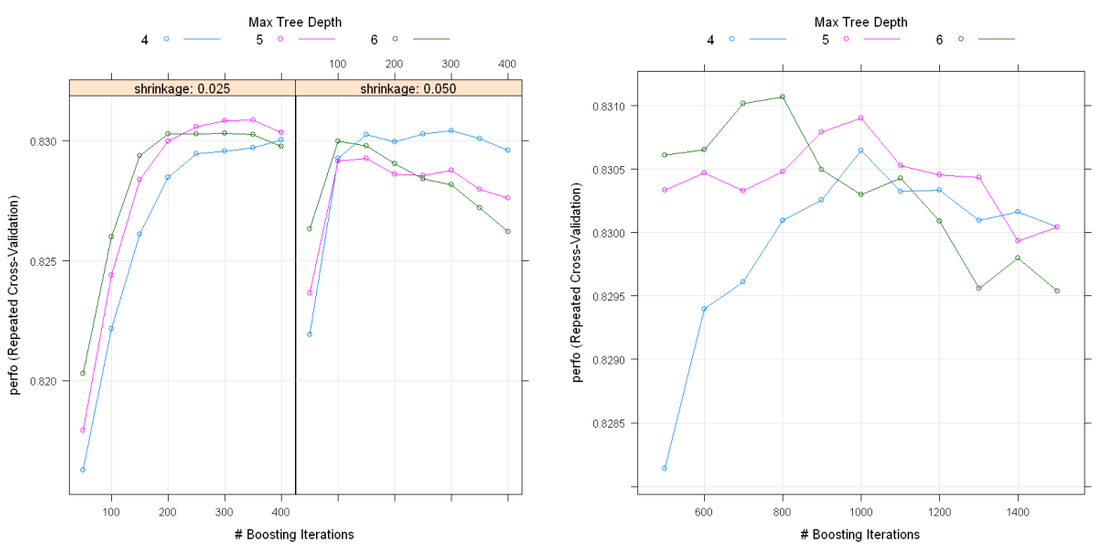

Figure 7: GBM Results (Learning rate is 0.01 for the graph on rhs)

For learning rates 0.025 and 0.05, tree depths 4, 5, and 6 and boosting iterations from 50 to 400 the best performing GBM model was the one with 350 iterations, 5 depth and 0.025 learning rate. The performance was 83.09%. For learning rate 0.01, tree depths 4, 5, and 6 and boosting iterations from 500 to 1500, the best model was with 6 depth and 800 iterations where the performance is similarly 83.11%. GBM runs was performing relatively better. Resulting plots for other parameters are shown in Figure 7 above.

3.7 Adaptive Boosting

First, we tried coefficient types Breiman, Freund and Zhu, tree depths 3, 4 and 5. number of trees 6, 9, 12, 15, 20, 30 and 40. We realized that the method performs better for Breiman coefficient, tree depth 4 and high number of trees, we tried models with Breiman coefficient, tree depth 4 and number of trees from 30 to 120. The best model was the one with 100 trees with 82.33% performance. Last results are shown in Figure 8 below.

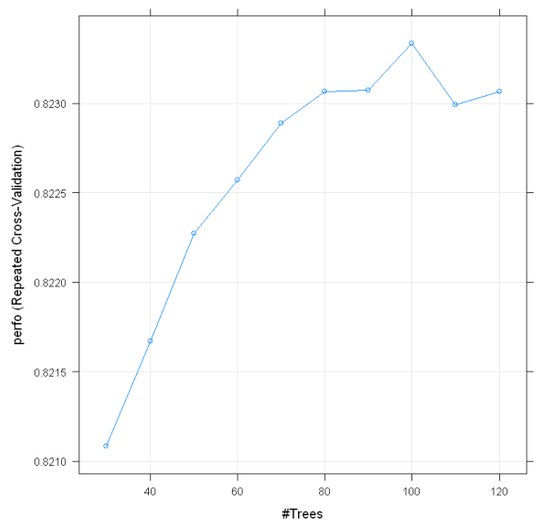

Figure 8: AdaBoost Results

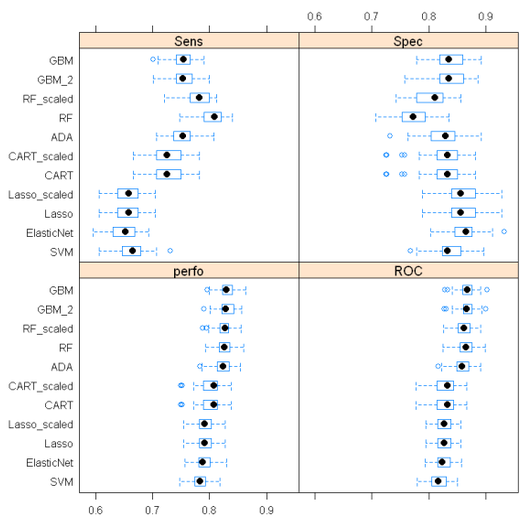

Figure 9: Method Comparison

3.8 Method Comparison

After selecting best models from each method, we compared the results from different methods in terms of average performance and reliability. To compare results from 50 re-samples we obtained the box plot above in Figure 9. Here, we see that GBM results are relatively better than others, in terms of performance metric. In addition, comparing the variances of methods, GBM results are acceptable. Submitting both versions, we obtained better results for GBM2 and finally, selected it.

4 Conclusions and Future Work

We were asked to construct a model predicting gender of a customer using unstructured clickstream data. Extracting 17 numerical and 13 categorical features from raw data, we created structured data. Using 10-fold cross validation with 5 repetitions and oversampling minority class on GBM with learning rate 0.01, depth 6 and 800 iterations, we constructed a model with around 83% performance on train data. To go further, one can

- Add new features that might help distinguish genders of customers
- Check for columns in training data that might cause over-fitting and remove them.
- Try other encoding methods than one-hot encoding for categorical variables.
- Try other sampling/class balancing methods for class imbalance.

5 Code Links

- Descriptive Analysis(ipynb).
- Feature Engineering(ipynb).
- Model Selection(ipynb).
- Feature Engineering(html).
- Model Selection(html).
- Descriptive Analysis(html).