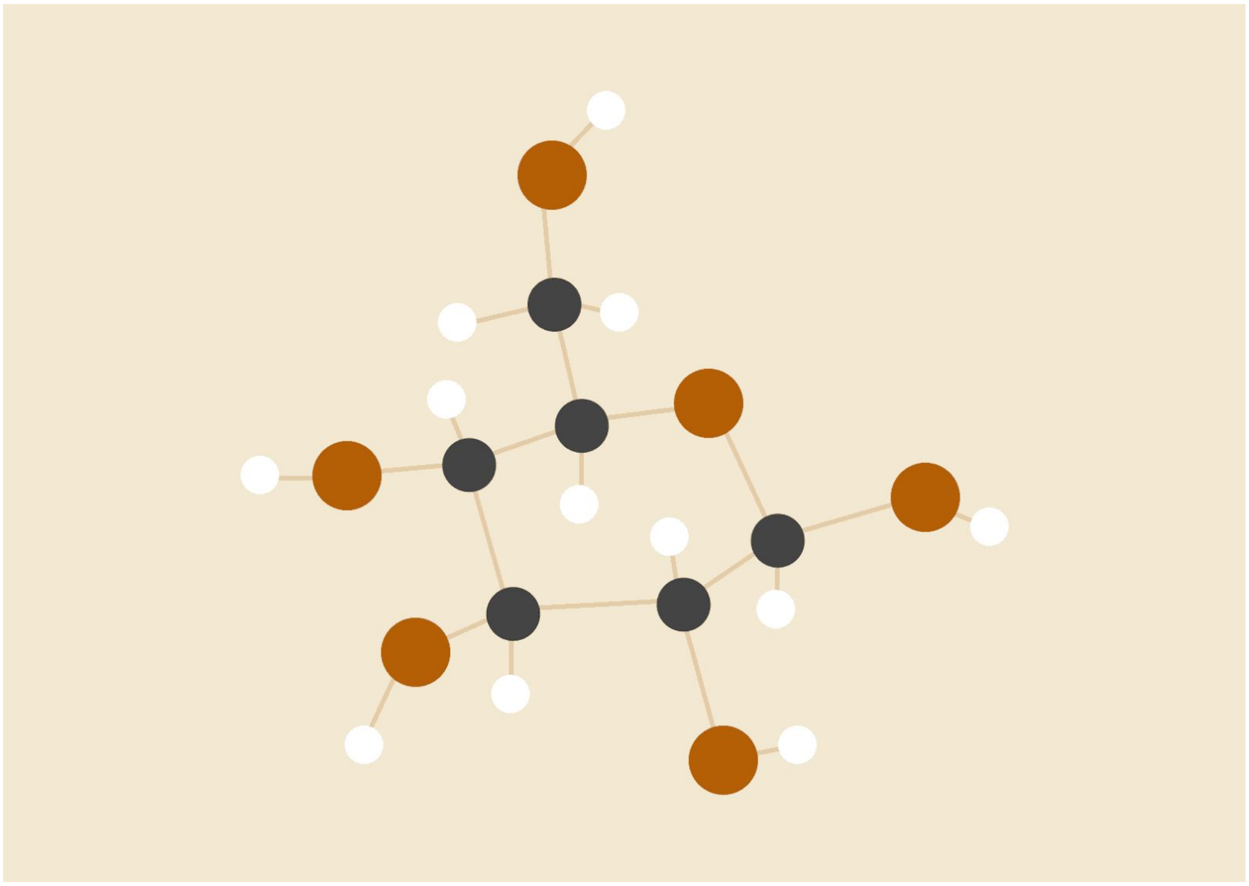
SYST-568 Project Report

Predicting Customer Loyalty Score



Addepalli, Bollam, Burugula, Merupula, Notash, punreddy

12.09.2019

INTRODUCTION

Kaggle competition on customer loyalty score prediction by Elo has provided historical and most recent transactions of customers by loyalty card id and historical data for merchants as well. Three anonymous features were provided in train and test data sets. Our goal is to generate features from the transactional data that better predict the customer loyalty score and get the Kaggle submission scores for each model.

DATA

All the datasets along with data dictionary can be found at this closed Kaggle competition. Data contains below listed files with brief description of data in each file.

- Train and Test - train and test datasets have card id, month of first purchase, three anonymous features and train data has target variable which is the loyalty score. Train has 202K records and test has 124K.
- History - transactional data by card id like purchase amount, product category and etc. Historical transactions file is 2.8 GB in size.
- New Merchant - Same dt as historical but for only last three months transactions and has 1.96 million rows.
- Merchant - this data contains merchant data which gives average sales, purchase at lags of 3, 6 and 12 months and has 335K records.

FEATURE ENGINEERING

Using the historical and most recent customer transaction data and merchant data, following features have been generated to be used as predictors in the models to predict loyalty score.

PURCHASE

Mean, median and frequency of purchase amount by card id from historical and most recent data were extracted which gave 6 features - hist_purhase_avg, hist_purchase_med, hist_purchase_frequency, new_purhase_avg, new_purchase_med and hist_purchase_frequency,

PRODUCTS

There are three binary columns based on product categories in the transactional data which indicate if a particular transaction included a product that belonged to a specific category. 6 features were generated which provide us frequency of purchase of a specific category product - hist_cat1_count, hist_cat2_count, hist_cat3_count, new_cat1_count, new_cat2_count and new_cat3_count.

MERCHANT

From merchant data two features historical and new purchase frequency at a premium merchant were generated. Premium merchant list was identified as the top 3000 merchants with highest average purchase lags in 3, 6 and 12 months.

DATA ANALYSIS AND PRE-PROCESSING

All the generated features were merged with training and test data sets. Few features were missing for some card ids. In order to submit predicted data to Kaggle, imputation has been done using “preProcess” function from “caret” package in R with method as “knnImpute”.

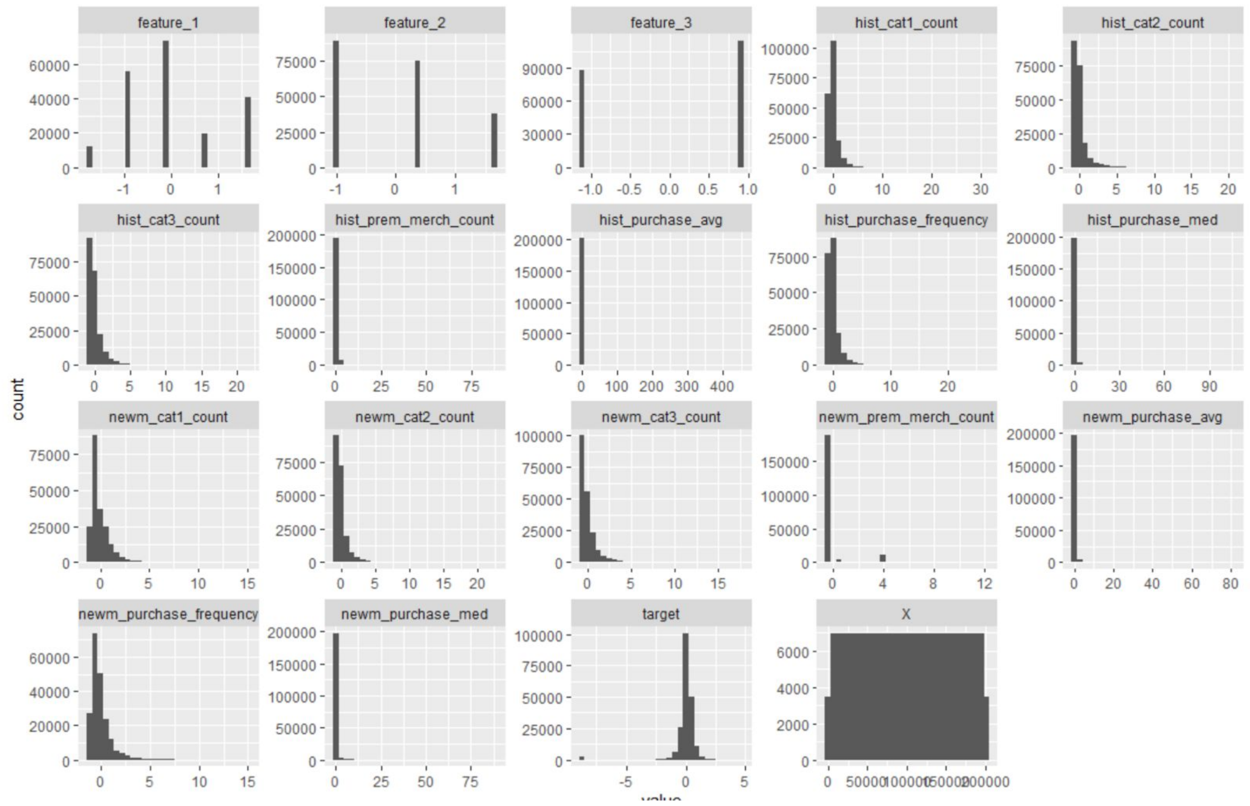


Fig [1]. Distributions of variables in the training dataset

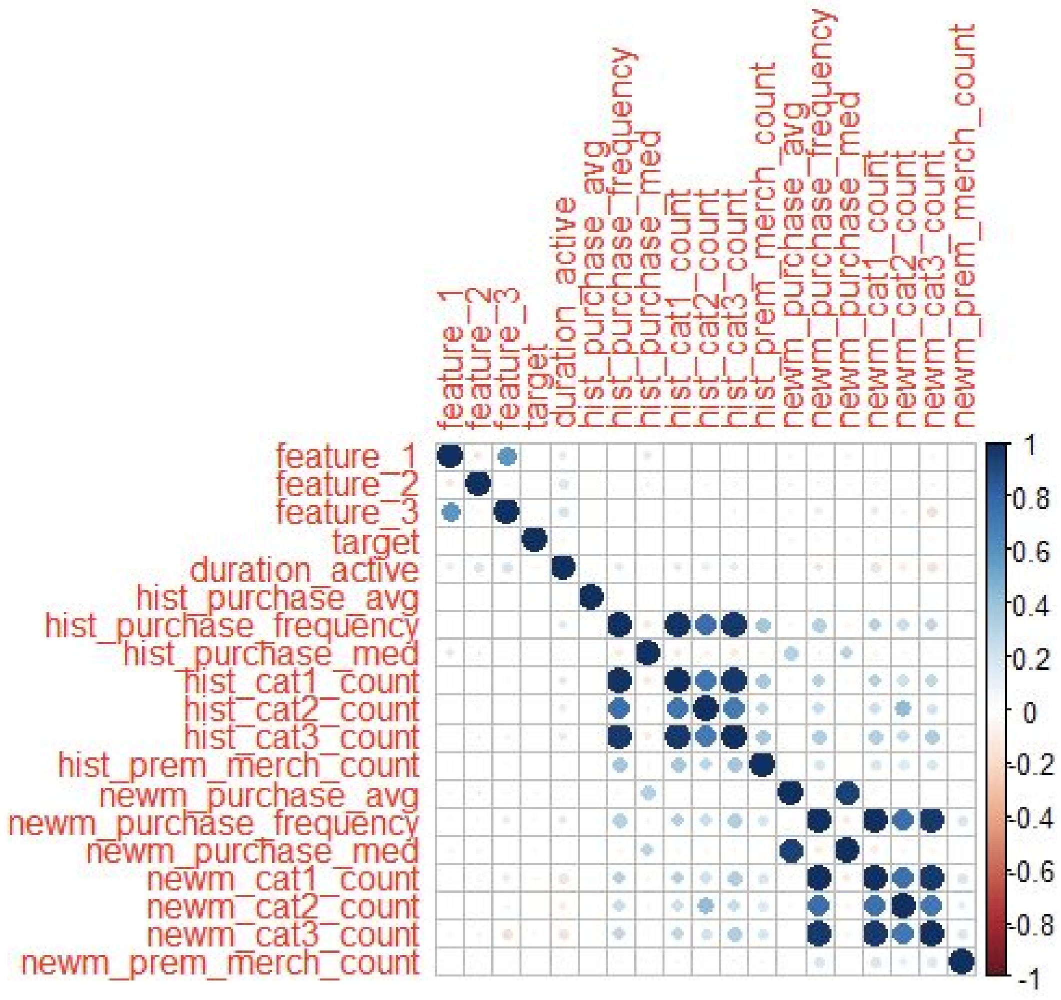
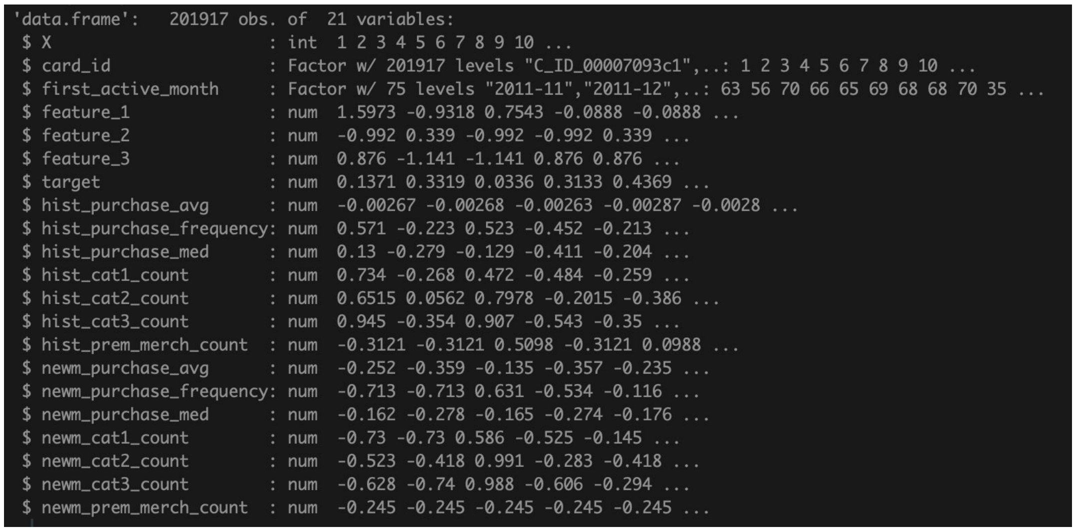


Fig [2]. Predictors distributions and correlation plots.



```
'data.frame':   201917 obs. of  21 variables:
 $ X                      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ card_id                : Factor w/ 201917 levels "C_ID_00007093c1",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ first_active_month     : Factor w/ 75 levels "2011-11","2011-12",..: 63 56 70 66 65 69 68 68 70 35 ...
 $ feature_1              : num  1.5973 -0.9318 0.7543 -0.0888 -0.0888 ...
 $ feature_2              : num  -0.992 0.339 -0.992 -0.992 0.339 ...
 $ feature_3              : num  0.876 -1.141 -1.141 0.876 0.876 ...
 $ target                 : num  0.1371 0.3319 0.0336 0.3133 0.4369 ...
 $ hist_purchase_avg      : num  -0.00267 -0.00268 -0.00263 -0.00287 -0.0028 ...
 $ hist_purchase_frequency: num  0.571 -0.223 0.523 -0.452 -0.213 ...
 $ hist_purchase_med      : num  0.13 -0.279 -0.129 -0.411 -0.204 ...
 $ hist_cat1_count        : num  0.734 -0.268 0.472 -0.484 -0.259 ...
 $ hist_cat2_count        : num  0.6515 0.0562 0.7978 -0.2015 -0.386 ...
 $ hist_cat3_count        : num  0.945 -0.354 0.907 -0.543 -0.35 ...
 $ hist_prem_merch_count  : num  -0.3121 -0.3121 0.5098 -0.3121 0.0988 ...
 $ newm_purchase_avg      : num  -0.252 -0.359 -0.135 -0.357 -0.235 ...
 $ newm_purchase_frequency: num  -0.713 -0.713 0.631 -0.534 -0.116 ...
 $ newm_purchase_med      : num  -0.162 -0.278 -0.165 -0.274 -0.176 ...
 $ newm_cat1_count        : num  -0.73 -0.73 0.586 -0.525 -0.145 ...
 $ newm_cat2_count        : num  -0.523 -0.418 0.991 -0.283 -0.418 ...
 $ newm_cat3_count        : num  -0.628 -0.74 0.988 -0.606 -0.294 ...
 $ newm_prem_merch_count  : num  -0.245 -0.245 -0.245 -0.245 -0.245 ...
```

Fig [2]. Summary of the training data set with generated features merged.

Correlations is strong between mean and median predictor variable which is expected. Predictors distributions how that most of the features are densely populated around small values. Scale, center and BoxCox pre-process techniques have been used in train command for each model.

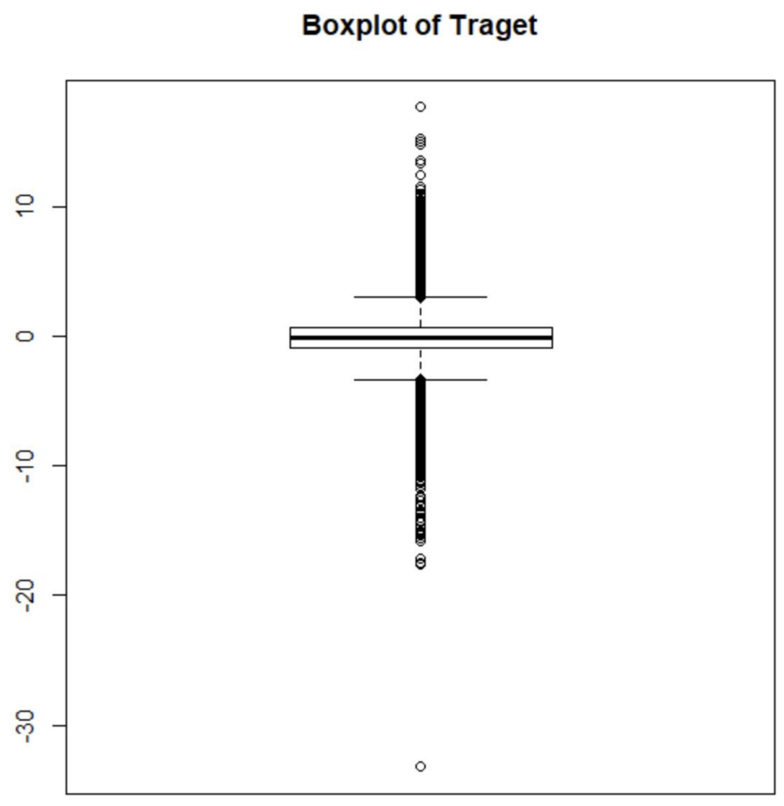


Fig [3]. Boxplot of loyalty score which is the target variable.

From the boxplot of the target variable it can be observed that the target variable is densely populated around small range of values which might cause lower values for R squared in predictor models. Removing outliers will affect the models predictability of higher values of targets.

“createDataPartition” from “caret” package has been used to generate training and test samples and two thirds data has been used for training with 10 fold cross validation and one third was used to testing. Repeated cross validation was not used as the non-linear models were taking long to fit and any repetition will multiply the amount of time taken to fit models.

MODELS FITTING AND TUNING

OLS with all predictors

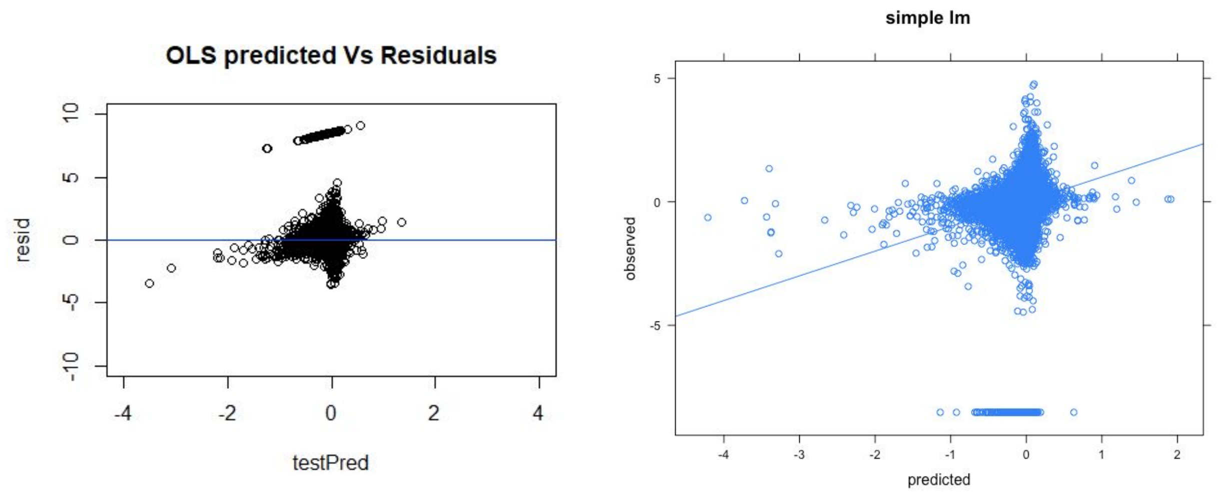


Fig [4]. OLS predicted versus observed and residual plots.

OLS model fit results are as shown for 10-fold cross validation, test prediction and Kaggle submission.

	RMSE			
10 Fold Cross Validation	0.9947685	Cross Validation results		
test set	1.014646	RMSE SD	R SQUARED	R SQUARED SD
Kaggle Private	3.82924	0.02023652	0.01180463	0.001802379
Kaggle Public	3.9469			

Table [1]. OLS model results

It can be seen that R square which describes the ratio of variance explained by the model to the variance in the data is very low. This is the result of the very small prediction interval of the target. Models with low R squared still show the predictability of the features and the R squared is to be interpreted in combination with RMSE and the predictor vs residuals plots. Residual plot shows that the residuals are around the 0 line except for the targets with high values. Model is not predicting higher values of loyalty scores.

PLS

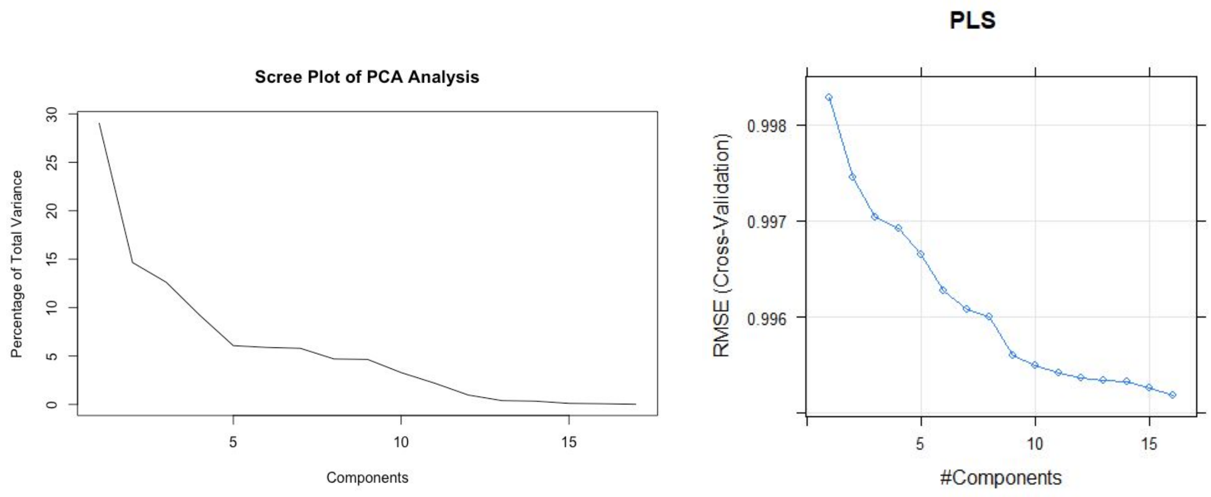


Fig [5]. PLS tuning and PCA results

PLS tuning parameter number of components is used as the tuning gtid for fitting the model and the lowest RMSE was achieved with 16 components. In the context of components Principal Component Analysis suggested that 12 components are enough to capture 99% of the variance in the data. It is to be noted that PCA does not consider the target variable when generating the components but PLS components are optimized in the correlation of the target variable.

	RMSE			
10 Fold Cross Validation	0.9948952	Cross Validation results		
test set	1.014683	RMSE SD	R SQUARED	R SQUARED SD
Kaggle Private	3.82932	0.02876672	0.0118684	0.002546333
Kaggle Public	3.94694			

Table [2]. PLS Results

Ridge and ENET

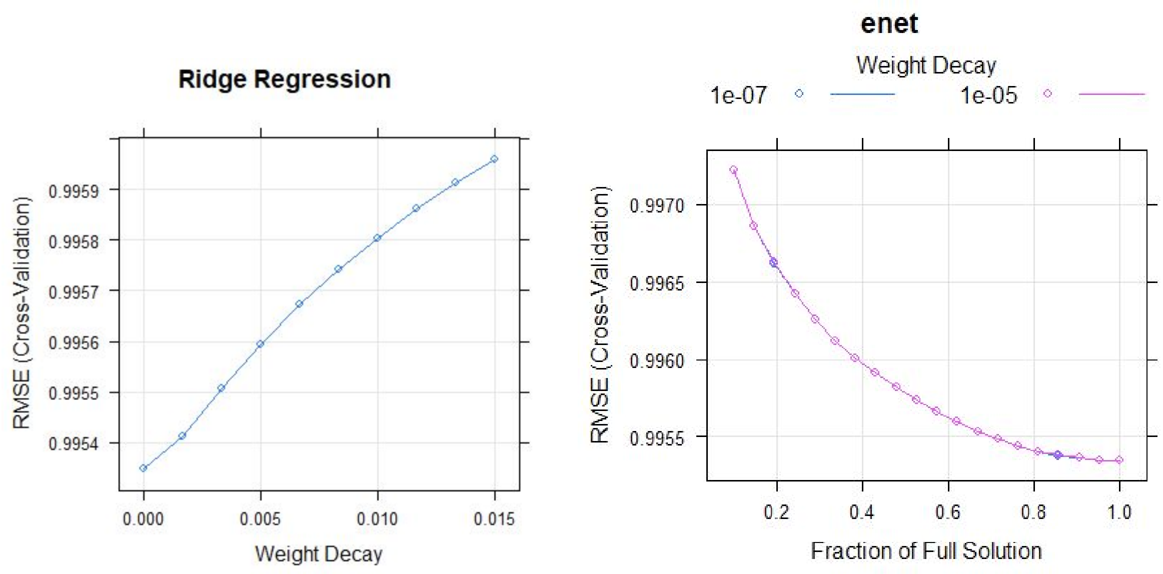


Fig [6]. Ridge Regression and ENET tuning plots.

	RMSE			
10 Fold Cross Validation	0.9953482	Cross Validation resutls		
test set	1.014646	RMSE SD	R SQUARED	R SQUARED SD
Kaggle Private	3.82924	0.02023652	0.01180463	0.001802379
Kaggle Public	3.9469			

Table [3]. Ridgle and ENET results

Both Ridge Regression and ENET results are pretty much the same and close to OLS. This is the result of the weight decay 0 is giving best RMSE in both the cases.

Neural Networks

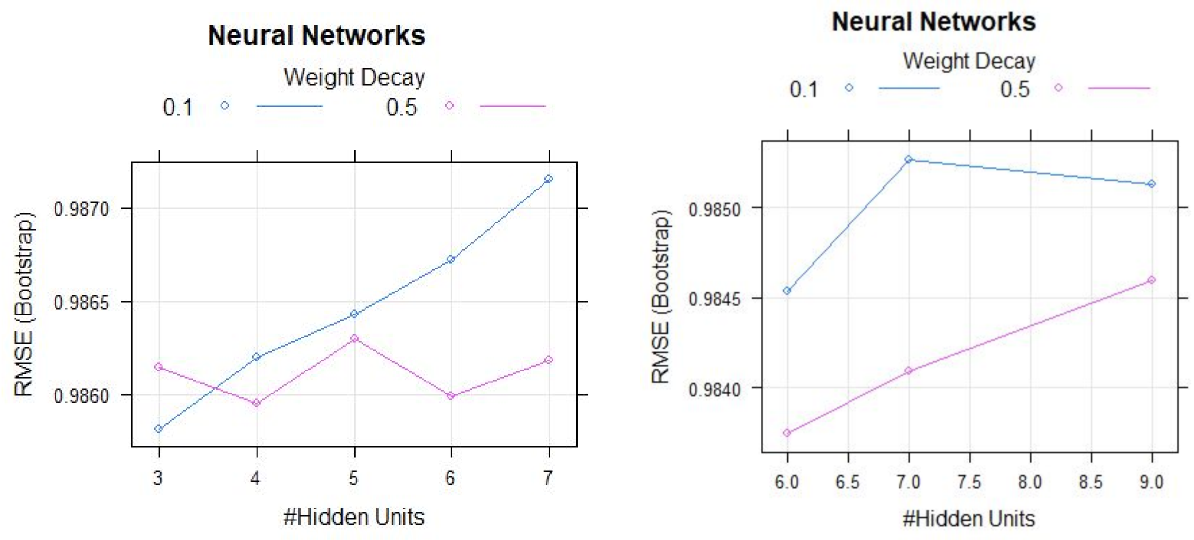


Fig [7]. Neural Networks tuning plots.

Initial model tuning grid started with number of hidden units as 6 and the model chose 6 and weight decay as 0.5 for the best RMSE. Later when the number of hidden units were reduced and tried the fit again, number of hidden units was chosen as 4 with weight decay as 0.1.

Model with lower weight decay resulted in lower RMSE with the known data but gave higher RMSE with unknown data on Kaggle. This is a clear case of overfitting and model with higher weight decay is a better option.

	RMSE			
10 Fold Cross Validation	0.9837494	Cross Validation resutls		
test set	0.9839988	RMSE SD	R SQUARED	R SQUARED SD
Kaggle Private	3.8174	0.0125379	0.02674509	0.00221748
Kaggle Public	3.93501			

Table [4]. Neural Networks results

Random Forest and XGBoost

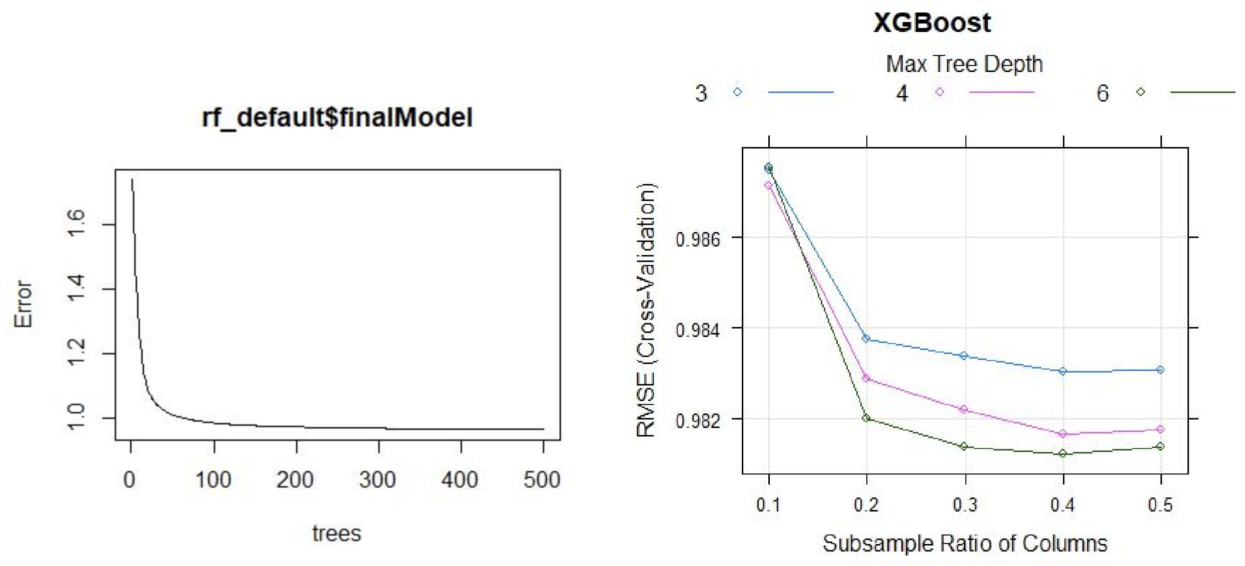


Fig [8]. Random Forest and XGBoost tuning.

Random forest took about 56 hours to run and RMSE turned out to be on par with linear models and Neural Networks and XGBoost outperformed. Optimal hyper parameters for XGBoos are maximum tree depth 6 and subsample ration of columns is 0.4. XGBoost performance in terms of time taken to fit model is better than Random Forest and Neural Networks.

	RMSE			
10 Fold Cross Validation	0.9812294	Cross Validation resutls		
test set	0.9792003	RMSE SD	R SQUARED	R SQUARED SD
Kaggle Private	3.81179	0.04402457	0.03864	0.006661229
Kaggle Public	3.92797			

Table [5]. XGBoost Results.

VARIABLE IMPORTANCE

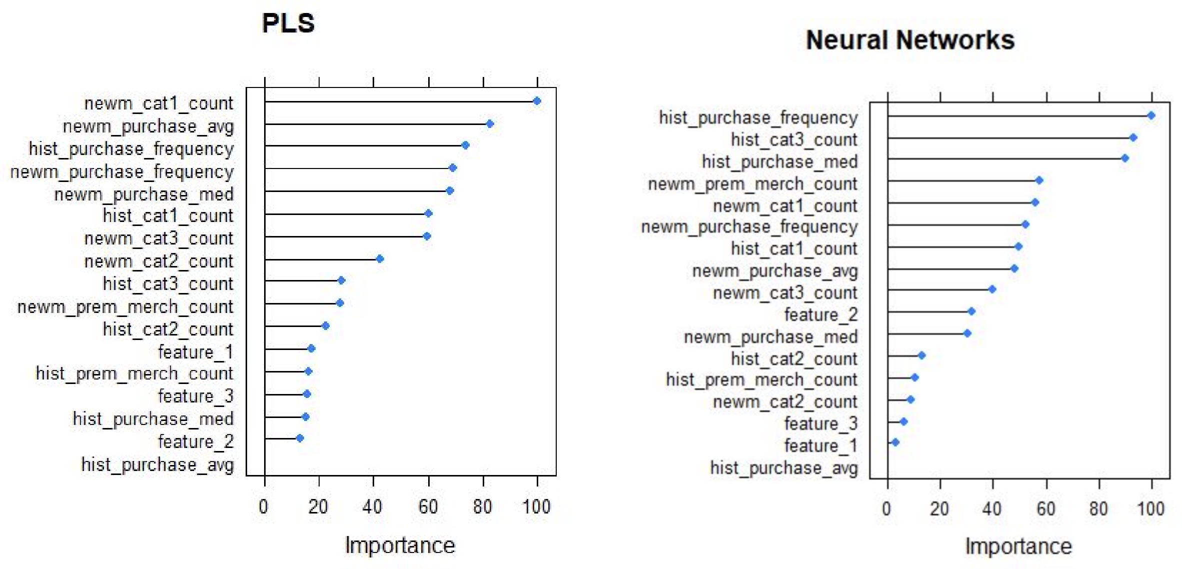


Figure [9]. Variable Importance by PLS and Neural Networks

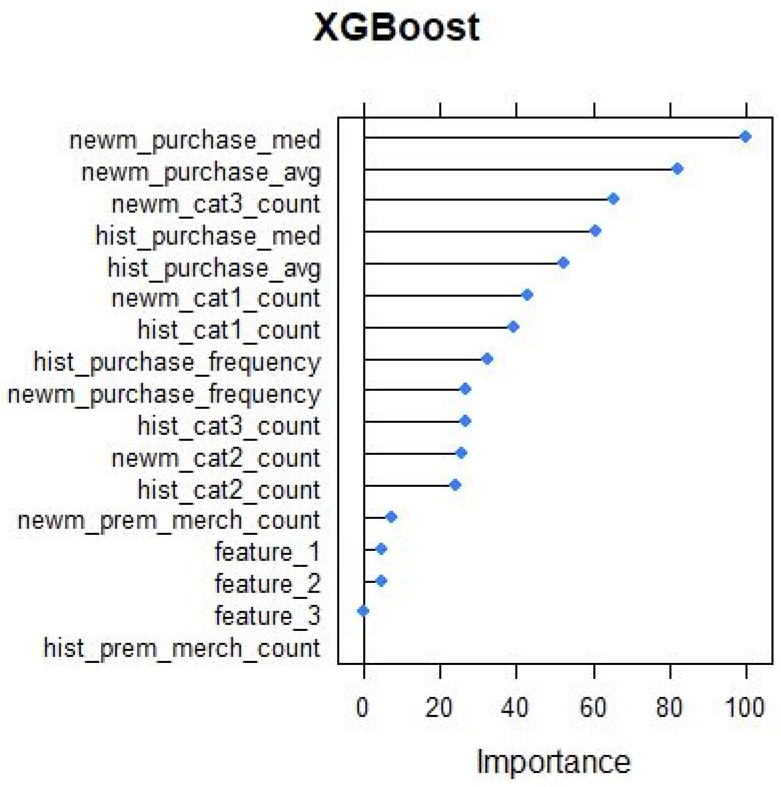


Figure [10]. Variable Importance by XGBoost

Historical average purchase amount was least preferred in both PLS and Neural Networks but interestingly XGBoost chose it to be one of the top 5 important variables. Most recent purchase median and average and purchase features related to products in category 3 also of high importance in all the three models. Purchase frequency at one the premium merchants is chosen to be important by Neural Networks model.

RESULTS

MODEL

XGBoost has the lowest RMSE and better predictor versus residuals spread. Also XGBoost took less time to fit model compared to the other non-linear models. PLS seems to be the lowest performing model and all the linear models took far less time to fit compared to any other non-linear model.

OBSERVATIONS

We anticipated linear models would perform better than non-linear models and the target, customer loyalty score, could be a calculated score based on certain quantity and quality measures. Linearity is not observed in the data.

It is hard to predict continuous values which are densely populated in a short range of values with outliers spread across very high values.

Prime location would be one of the important features that will be chosen by every model, we thought. Neural Networks identified this as one of the top 5 important features. As the distribution of this feature is not well spread, other models could not make use of this feature.

None of the anonymous features provided by Elo were not of any importance in any of the models.

ENHANCEMENTS

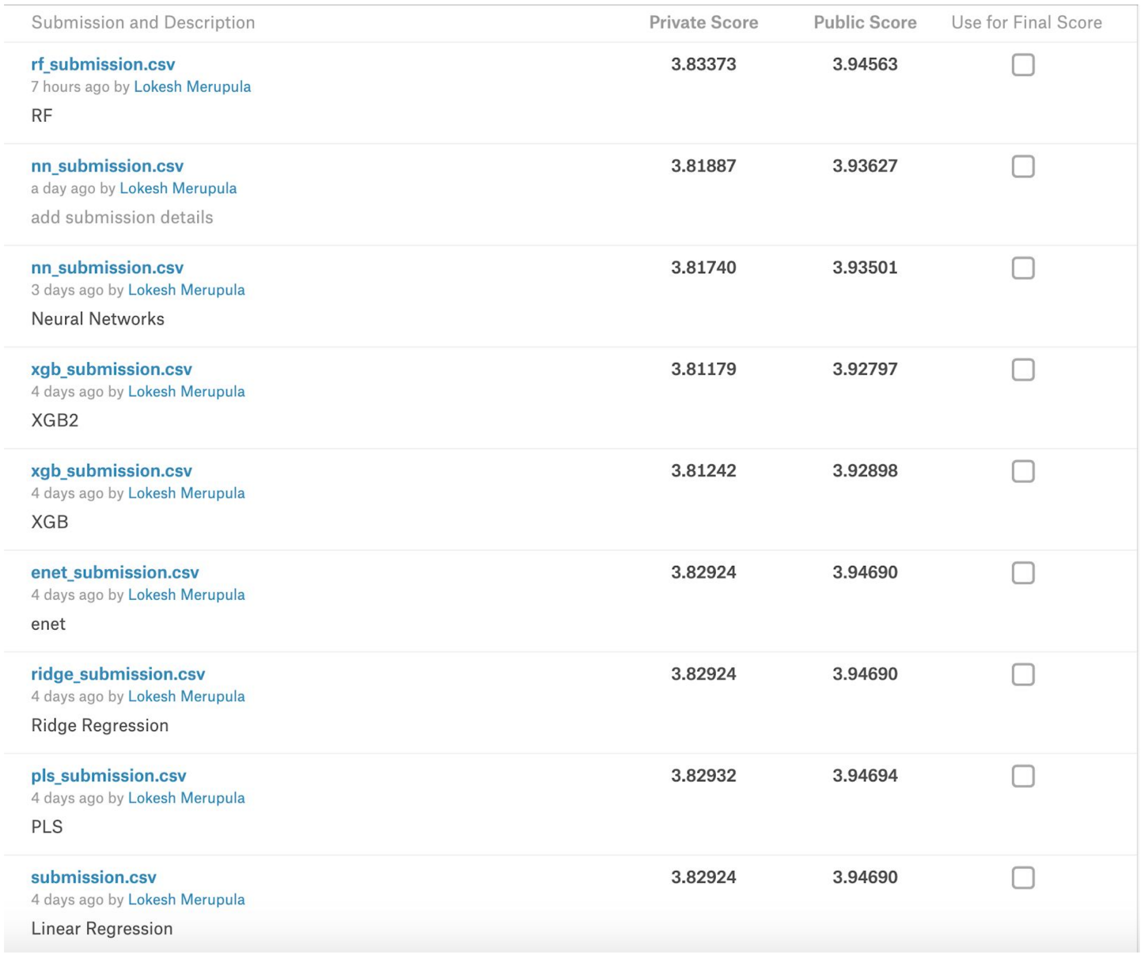
These features might have better predictability.

- Duration since active.
- Features based on transaction date.
- More features based on which merchant the customer shops at.

SVM is an optimization model and was taking forever to converge. Troubleshooting and using GPU might help the model to converge.

SCORES

Kaggle competition winning private score was 3.57875 and public score was 3.61285. Our best model RMSE private was 3.92797 and public score was 3.81179. Below is the screenshot of Kaggle submissions scored.



Submission and Description	Private Score	Public Score	Use for Final Score
rf_submission.csv 7 hours ago by Lokesh Merupula RF	3.83373	3.94563	☐
nn_submission.csv a day ago by Lokesh Merupula add submission details	3.81887	3.93627	☐
nn_submission.csv 3 days ago by Lokesh Merupula Neural Networks	3.81740	3.93501	☐
xgb_submission.csv 4 days ago by Lokesh Merupula XGB2	3.81179	3.92797	☐
xgb_submission.csv 4 days ago by Lokesh Merupula XGB	3.81242	3.92898	☐
enet_submission.csv 4 days ago by Lokesh Merupula enet	3.82924	3.94690	☐
ridge_submission.csv 4 days ago by Lokesh Merupula Ridge Regression	3.82924	3.94690	☐
pls_submission.csv 4 days ago by Lokesh Merupula PLS	3.82932	3.94694	☐
submission.csv 4 days ago by Lokesh Merupula Linear Regression	3.82924	3.94690	☐

Figure [11]. Kaggle Submissions scores for each model.

RUNNING THE CODE

1. Download the data from Kaggle - https://www.kaggle.com/c/elo-merchant-category-recommendation/data
2. Download the code (preprocess.R and analysis_and_models.R) into the same directory.
3. Update the line 13 in preprocess.R to set current working directory to the downloaded directory.
4. Run preprocess.R - it takes a long time to load the data into memory. As we saved data after pre-process, this step can be skipped in order to fit models.
5. If step #3 isn't done and takes too long, download the pre-processed data from Bb - train_dataset.csv and test_dataset.csv
6. Update the line 20 on analysis_and_models.R to the download directory.
7. Run analysis_and_models.R - Neural Networks, Random Forest take at least 48 hours to run,