2019

Elo Customer Loyalty Score Prediction

BU 425

Ruslan Nikolaev (150927200)

Anton Storozhilov (150665260)

Mike Ross (205671870)

Stephany Desroches (206137302)

Contents

Business Problem and Motivation

Predicting customer loyalty and providing personalized recommendations is something that many businesses would like to do in order to make the best investments in customer loyalty. This is exactly what Elo bank is trying to do in Brazil. Elo is a pure domestic credit card brand in Brazil – meaning it only allows for transactions in Brazilian Real currency. Elo is a successful partnership of 3 of the largest banks in Brazil - Banco do Brasil, Bradesco, and CAIXA. Currently, Elo is challenging major international competitors in Brazil, seeing a steep growth rate in market share while competitors like Visa see its market share decreasing equally as quickly.

This success has come largely from Elo's investments in partnerships with local businesses. Specifically, Elo has close relationships with local clothing and food retailers in order to offer its customers promotions and discounts for visiting the stores and using an Elo card. With over 50 million cards issued, Elo has a large amount of consumer data to analyze and work with. With this data, Elo currently creates simple retail recommendations for consumers to visit one of their partner stores. For the customer, this means that Elo recommends a store they can visit for food or retail they may be interested in. If this customer visits the store and uses their Elo card they can get, for example, 2% cash back on their purchases at the store.

Elo believes that these partnerships help drive consumer loyalty to the Elo brand over intense international competitors like Visa and Mastercard, but currently does not have a solution to track or measure the resulting loyalty scores in order to see the effectiveness of these recommendations they're providing to consumers.

This is the business problem that is to be solved through this project – how can Elo predict the resulting loyalty score from these recommendations they're offering to consumers?

Dataset

The dataset provided by Elo for determining the loyalty score for each card/consumer was split into 6 relevant files outlined below, as well as a data dictionary to describe the columns. Since this is a large set of real customer financial data, it was heavily anonymized with categorical variables mapped to integers in a random order, non-categorical variables with location shifted by some constant (creating purchase amounts that are negative), and some column names anonymized to "feature X".

- Training data: (per card ID, shape (202k, 6))
    - Card ID
    - Anonymized features 1,2,3
    - Target loyalty score
    - First active month of card
- Test Data (per card ID, shape (124k, 5))
    - Card ID
    - Anonymized features 1,2,3
    - First active month of card
- Merchant information (per merchant, shape (335k, 22))
    - Merchant ID
    - Group ID, Category, subsector ID,
    - Anonymized continuous and categorical columns
    - Purchases and sales lag (average of 3, 6, 12 months divided by last active month)
    - Location: city, state (anonymized)
- Historical transactions (per purchase, shape (29.11m, 14))
    - Card ID
    - Purchase date
    - Anonymized categories
    - Purchase amount
    - Number of installments of purchase
    - Merchant ID
    - Location: city, state (anonymized)
- Additional (historical) transactions (per purchase, shape (1.96m, 14))
    - Card ID
    - Purchase date
    - Anonymized categories
    - Purchase amount
    - Number of installments of purchase
    - Merchant ID
    - Location: city, state (anonymized)

Impacts of anonymization quirks within the dataset

As previously mentioned, the dataset was heavily anonymized to ensure that no real-world customer data was leaked through the Kaggle challenge. As such, several interesting quirks were observed in the dataset that needed to be analyzed to see if they would impact the results of the model.

From the performed analysis, most of the numerical continuous variables seemed to have some sort of shift and scale operation performed on them. The variable on which this became most apparent was in the purchase amount of historical transactions, where the average purchase amount was negative. For a credit card company like Elo, having a negative average purchase value is nonsensical as this would mean consumers have made more returns on Elo's credit cards than they have purchased.

For the linear regression and other models, simple data scaling should not have an impact on the results, particularly when determining statistical significance, since R can appropriately scale the associated β value for that feature in the model. If instead of having customer loyalty score as the dependent variable, we were using purchase amount as our dependent variable, then we would likely need to figure out how the anonymized data would affect any business conclusions such as expected spend per month.

While some Kaggle challenge participants have attempted to reverse engineer the real values of some anonymized variables (Barušauskas, 2019), this was not deemed necessary for the purposes of the linear and neural network analysis performed in this report.

Exploratory Analysis

The analysis was performed using two of the main methodologies introduced in class. The first part of the analysis was exploring the data with simple statistical techniques and techniques from linear regression. This analysis was performed to better understand the problem, the data being analyzed, and to establish a baseline level of prediction quality for the later analyses using a neural network and gradient boosted decision tree.

Data clean-up & Outlier Analysis

To better understand the data, an outlier analysis was conducted. This analysis included detecting any data collection errors and finding any data points that could have a disproportionate impact on the model. Several methods were considered including building an interquartile range, isolation forest, Cook's distance, Mahalanobis distance, and Z-score. For an initial exploration, a Z-score analysis was chosen for all non-categorical features across the Historical Transactions, Merchants, Train data sets.

This method was chosen over the others as it was one of the best methods for being both directly understandable and easiest to scale. This was helpful to achieve a quick initial look at a data set of over 30 million samples. The trade-offs involved with this method were the assumption of normality of the distribution of the data, and only being able to consider one feature at a time rather than a multivariate outlier analysis.

For the transaction history outliers across month lag, purchase amounts, and purchase installments were observed. The corresponding means for each were found to be around -4.1, -0.21, 0.65 indicating there are transformations in scale and location present. Across them, a total of 190,689 possible outlying samples were found. For the merchant data the sales, purchases, and active months "lag" features were observed, and 14,437 possible outliers were found. Finally, for the training data target scores was observed and 2,262 outliers were found.

Looking more closely at these outliers revealed the useful insight that the data contained both NaN and infinity values specifically for the months lag feature in the merchant data. The infinity values are hypothesized to appear for customers that have been active for less than a month; this is due to the fact that the calculation for lag columns are an average divided by the last month, which in this case would be 0. Thus, the conclusion made is that the missing data should be considered non-random and informative. Due to this, NaN and infinity values will be preprocessed before putting them into the model.

Linear exploratory analysis

Using techniques developed from class, the data was analyzed using linear methods to see if there were any immediately obvious relationships between any variables. As most datasets from Kaggle are motivated by messy, real-world problems, it was not expected to observe many statistically significant relationships. However, there is value in analyzing the data through a variety of different methods so the analysis was performed to see if any additional insights would be gained. The analysis was performed using linear models in both Python and R.

Loyalty score statistical properties

In order to get a sense of what the loyalty score means, the statistical properties of the score itself must be analyzed. The loyalty score variable has an average score that is slightly negative (-0.394). It can also be seen that most loyalty scores are near this average as the distribution has both the 25th-percentile (-0.883) and 75th-percentile (0.765) scores within the range of (-1,1). A large deviation from this range can only be seen near the extremes of the data. While loyalty score was the most interesting variable to analyze, a full statistical analysis of each relevant variable in our dataset has been prepared as a graphic in Exhibit 1.

The relationship between purchasing behaviour and loyalty score

The main analysis to be performed linearly is identifying any relationship between transactions made on a customer's card and their loyalty score, as this seemed like the most straightforward relationship that could be present. To create the dataset for this analysis, the historical transactions were aggregated by user (card_id) and the frequency, size, variance of purchases, and the number of merchants shopped at were analyzed. Combining this grouped data with the training set allowed the assignment a target loyalty score to each customer's purchase behaviour. From a quick visual inspection of a pair plot of the data, there is no strong correlation between any of the variables (See Exhibit 2).

The two relationships found that goes against the original hypothesis made are the relation among loyalty score, number of transactions made, and number of merchants shopped at. It was first assumed that a high

frequency of transactions would make a consumer's loyalty score higher, but this is not the case. Customers who make more transactions tend to have a loyalty score that deviates less from the average loyalty score but does not have higher scores overall. Customers who make fewer transactions have scores that tend to deviate more from the mean, either positively or negatively. As such, it seems that an increase in transactions served only to decrease the variance of loyalty score from the mean but did not actually increase the score itself. Similarly, it was also assumed that shopping at fewer merchants in the year would increase the loyalty score, but there is no correlation to suggest this. This overall relationship can be attributed to the positive relationship between the number of merchants shopped at and the number of transactions. This means that customers who increase spending do not stay loyal to just a small number of merchants but tend to diversify their transactions across merchants.

Merchant sales data relationships

Another use of linear analysis is to identify confounding variables that are correlated with each other. By removing correlated variables from the model, complexity, and potentially overfitting, can be removed.

Again, using pairplots (see Exhibit 2), a strong correlation between the "lag" family of variables was identified in the merchant data set. The last three, six, and twelves months of monthly average sales at each merchant were evidently correlated with each other, so it is logical to only include one of these in the models. Similarly, the lagging three, six, and twelve months of average monthly revenue are also correlated in a similar manner, meaning two of these highly correlated variables can be removed to reduce complexity and increase explainability of results.

Linear exploratory model

A baseline linear model was run in R to establish if there are any correlations between variables in a combined dataset of loyalty score and purchase behaviour with loyalty score as the dependent variable. Running a linear regression on all 76 columns and 325,000 records was not possible as R failed to

initialize enough memory, asking for upwards of 300GB. Instead, it was decided to analyze a subset of the columns in different regressions to see if anything meaningful could be identified.

A linear regression was first run exclusively with the training dataset with the card features (perks) as the independent variables and loyalty score as an independent variable. From this first model, no meaningful results could be identified as the r-squared value for the model is extremely low, at around 0.002. Secondly, a linear regression was built with the number of purchases in each state_id (geographical region) as the independent variables. The output of this model was another extremely small adjusted r-squared value, at 4.783e-05, indicating a poor model fit. Lastly, a model was built with purchase behaviour for each user, including independent variables such as average monthly purchases, average purchase amount, maximum and minimum purchase amount, number of authorized transactions, and average days between transactions, to name a few. This model scored slightly better on the adjusted r-squared measure at 0.0036, but the extremely low value still implies that this is a very poorly fitting model.

Based on these linear regressions, and several others explored, it is determined that the complexity of the dataset and the sheer number of features make it very difficult to make a meaningful linear model to predict loyalty score. As such, a neural network and gradient boosted decision trees are used to help analyze the problem further.

Neural Network and Gradient Boosted Tree Model

Neural network analysis/prediction

To join all the mass of transaction and merchant data to the actual training and testing data, the mass data first must be preprocessed. There are four steps performed: joining data tables, treatment of outlier/missing data, one hot encoding, and feature engineering. Due to the size of the raw historical purchase data being over 30 million data points, all this analysis also had to be done in batches of 10 million to fit in the computer working memory and then joined appropriately.

Firstly, three of the tables Merchant Information, Historical Transactions, and Additional Transactions were joined. Historical Transactions and Additional Transactions were directly concatenated as they both contained rows of the same type of data. Then the resulting table was left joined to the merchant data on Merchant IDs, so that for every transaction the relevant merchant information was added. Additionally, features were dropped to reduce model complexity. All average sales lag features were dropped due to correlation with average purchase lag features, and out of 3-month, 6-month, and 12-month, only 12-month was kept as it was hypothesized to contain the most information, as seen in Exhibit 2. Merchant Group ID was dropped as it had over 900 categories that would blow up the complexity of the model when it was hot encoded.

Next, missing data was processed. As mentioned in the outlier analysis, dropping variables with missing data should be avoided since it is hypothesized these occurrences are non-random. Furthermore, if transactions with missing data were to be omitted then it might result in Card IDs with no transaction data at all. This should be avoided since it is desirable to keep all Card IDs within the domain of the machine learning model. The missing categorical data were imputed by replacing with 'Z'. For non-categorical variables, missing values were replaced with -1 for features which were always sampled positive, and by the mean for features that contained both negative and positive values.

For a model to be able to work with categorical variables one hot encoding should be used to create binary dummy variable columns for every category of every categorical feature. For example, if a category was color and a sample was blue then its blue column would have a value of 1 and the red and green columns would have a value of 0.

Finally, the training and test set samples are indexed by unique IDs so the supplementary transaction data which contains multiple transactions per ID must be converged to a single set of features per ID. For the feature engineer process key summary statistics were applied to the sets of purchase data for each ID – optimizing the trade-off between capturing as much information as possible while avoiding increasing the model complexity through too many features. For all one-hot encoded transactions, the

summary statistics were a count of the number of times each category appeared in the transactions. Anonymized features and poorly interpretable features such as average month lag was summarized by a mean and standard deviation as well as a min and max. Features that could be meaningfully interpreted such as purchase dates had additional statistics such as days between purchases that could be related to linked. A count of the number of purchases, from which all the other statistics are derived, was added as a feature capturing information that would be necessary for normalization.

To see the measured importance of each of the used variables refer to Exhibit 3. It is interesting to note the unexpected results of the importance graph where the features included in the training data are at the bottom of importance, while seemingly unrelated features like authorized_unauthorized_ratio at the top.

Building the Model

After preprocessing the data, different models were set up, trained, and then tested. Before getting into the models themselves the training versus testing data must be determined. Initially, when trying out the models there was a split of the provided "training data" into a training set and testing/validation set which was split 90:10 training to validation. This resulted in 181,800 testing samples and 20,200 validation samples. However, later in the process, there was a switch to using all the training data for training and testing it on the Kaggle website.

Before being fed into any model the processed data batches must be merged and then left joined with the training data to get training data indexed by IDs with both anonymized features and supplementary features from all the processed data. Refer to Exhibit 4 for overview of data processing. Any training IDs that did not have corresponding processed data were dropped because there is no way to mitigate that much missing data without degrading the model. The final step is to shuffle the data to avoid being impacted by any order-based patterns as the models iterate over sections of data in epochs.

After this set up the optimal model began being built, with a total of 27 model iterations before reaching the final one (summarized in Exhibit 5). The first model tried was putting 542 features into a

Neural Network with 1 layer of 256 neurons and 1 output layer of 1 neuron linear activation function running for 30 epochs. This gave an initial Mean Squared Error (MSE) test loss of 11.29. Through further experimentation, the Root Mean Squared Error (RMSE) test loss was able to be reduced to 1.11 by dropping to only categorical features and select non-categorical features such as purchase amount, installments, and month lag. Unfortunately, after comparing with top submissions on Kaggle that were much higher than this value, it was quickly realized that this model was not producing the right output.

This is when feature engineering as described above was introduced into our model to summarize the transaction history for a card ID within a single set of features. After doing this the MSE training loss became relevant at 3.7241 and the test loss grew to 3.9312, as tested on Kaggle. Based on the gap between the training and test loss, the new goal became increasing generalizability of the model. To do this a regularization technique of adding a dropout layer as the second layer was employed. After a couple of iterations with a 0.5 dropout, test loss was reduced to 3.9307. However, further modifications to the model only increased overfitting and were not able to make meaningful progress.

At this point there was a realization that an assumption had been made - this was that a Neural Network was the best model due to the 30 million data points found in original transaction data. However, after collapsing it to the 202,000 data points in the training data, a Neural Net was too complex of a model for data. This led to the pivot of a new model being a Gradient Boosted Decision Tree. This model with 500 estimators, max depth of 4, and min sample split of 2. This immediately gave a significantly lower test score of 3.9153.

After detecting NaNs in the processed data and replacing them, only training error was able to be reduced but the test error increased. Looking at a graph of test versus train error the test error is flat, indicating overfitting once again. From this, the samples split, estimators, and max depth were iterated upon to get down to the best model that scored 3.91345.

Results

As a result of this analysis, this model (GBT) can be stated as capable of predicting target customer loyalty score with reasonable accuracy, where RMSE: 3.91345. This model also scores well relative to the best submission on Kaggle where RMSE: 3.61285. As a whole, this model has ΔRMSE of 0.3006 from the best submission, which makes it a feasible predictor for customer loyalty. Finally, a decision tree derived from a Gradient Boosted Tree model will be useful in business contexts to add explainability to the model.

Future Work

There are several possible model improvements that were inspired by discussions on Kaggle. The first is further data preprocessing. With the capabilities of decision trees to output feature importance, further feature optimization can be performed to develop better summary statistics on historical transactions. The second is developing separate predictors for outlier data and non-outlier data. This would require building a classifier which can detect between outlier and non-outlier data based on provided summarized historical transactions and using separately trained predictors, where one is trained on outlier data and the other on non-outlier data. This approach has the potential to prevent model overfitting, which is a problem with the current model.

Recommendation

Elo's goal of predicting customer loyalty is now within reach as the model discussed has demonstrated strong predictive capability from the provided data. As mentioned by Elo in the presented dataset, previously there did not exist a method that could predict customer loyalty and, in many cases, Elo was overspending on their customer retention strategies. With this new prediction capability, Elo will be able to serve targeted retention campaigns for its customers, which will decrease its marketing spend and increase customer retention and loyalty scores. Note, that many solutions proposed on Kaggle used neural network approach to the problem, which has little explainability and does not present no feature

importance for the model. In our case, due to the final choice to use a Gradient Boosted Tree model, a visual diagram was created of the model's decision-making process which is valuable in a business context to add explainability to predictions (Refer to Exhibit 6). Finally, based on existing feature importance diagram (refer to Exhibit 3), Elo's data-scientists can extrapolate which other features could be useful to understand customer loyalty and start collecting this additional data to further improve model accuracy in the future, which is not possible in case of a neural network.

Exhibits

Exhibit 1 – Statistical properties of variables

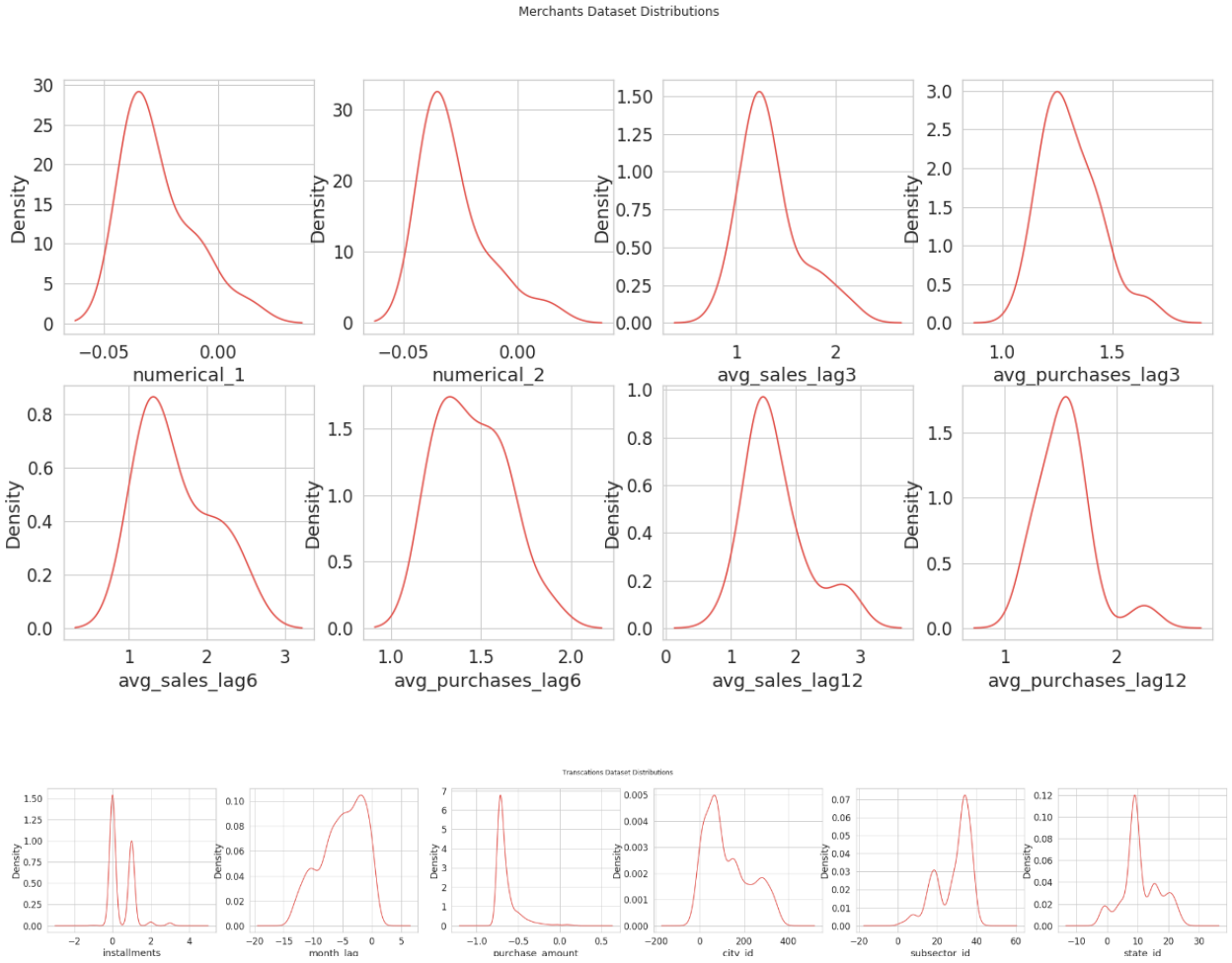
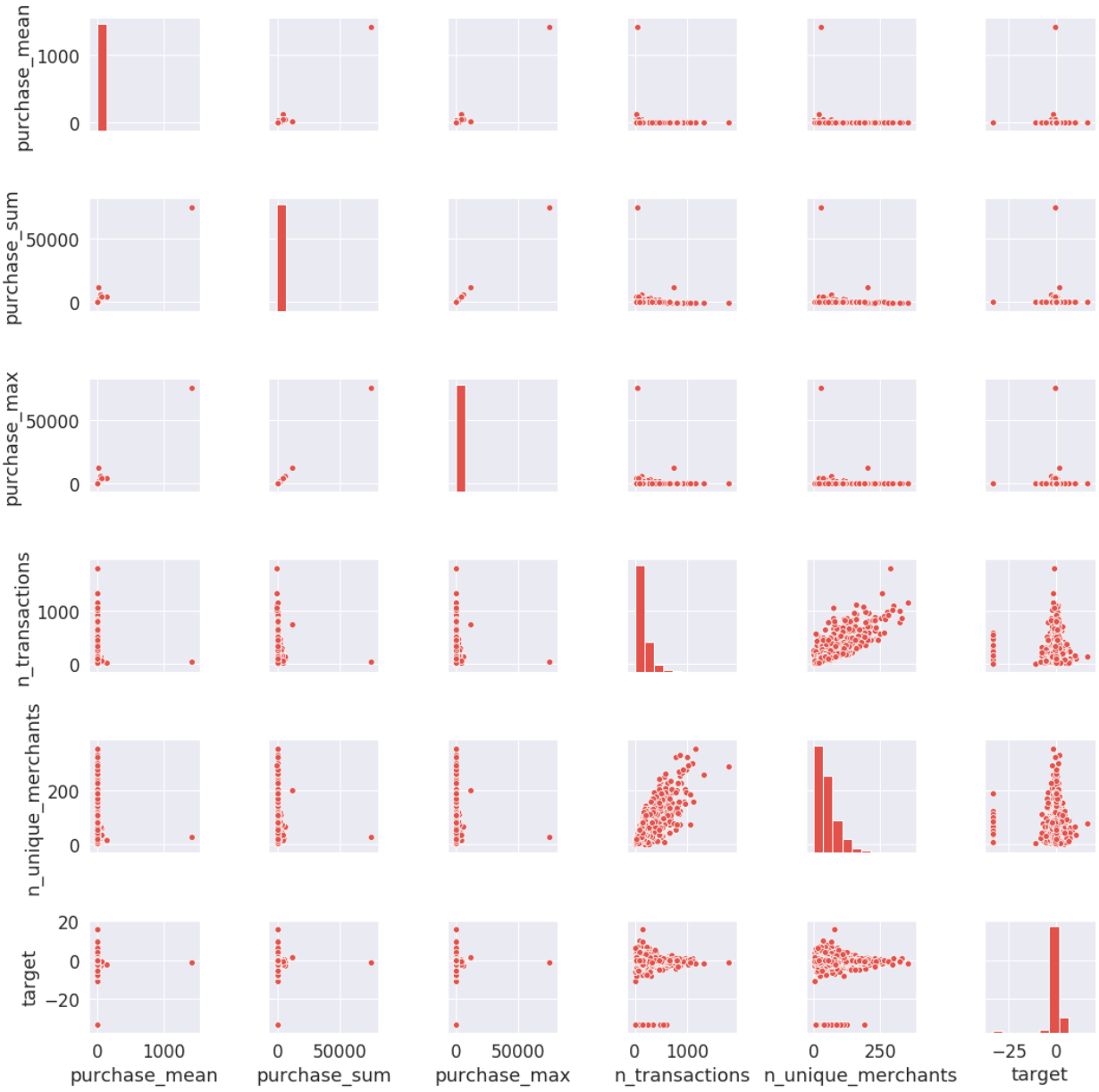


Exhibit 2 – Pairplot for Historical Transactions Dataset



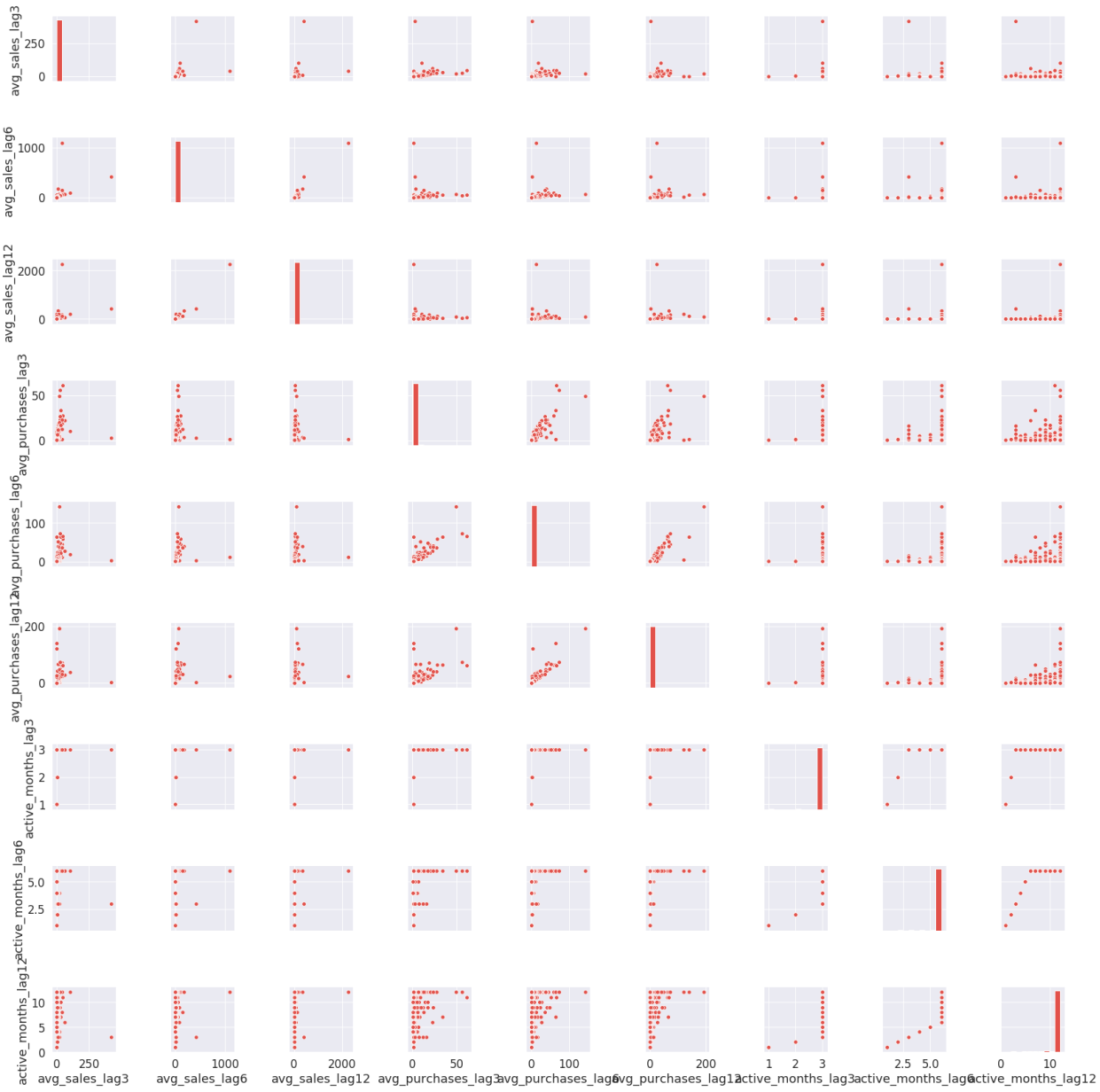

avg_sales_lag3
avg_sales_lag6
avg_sales_lag12
avg_purchases_lag3
avg_purchases_lag6
avg_purchases_lag12
active_months_lag3
active_months_lag6
active_months_lag12

Exhibit 3 – Gradient Boosted Tree Model Variance Importance

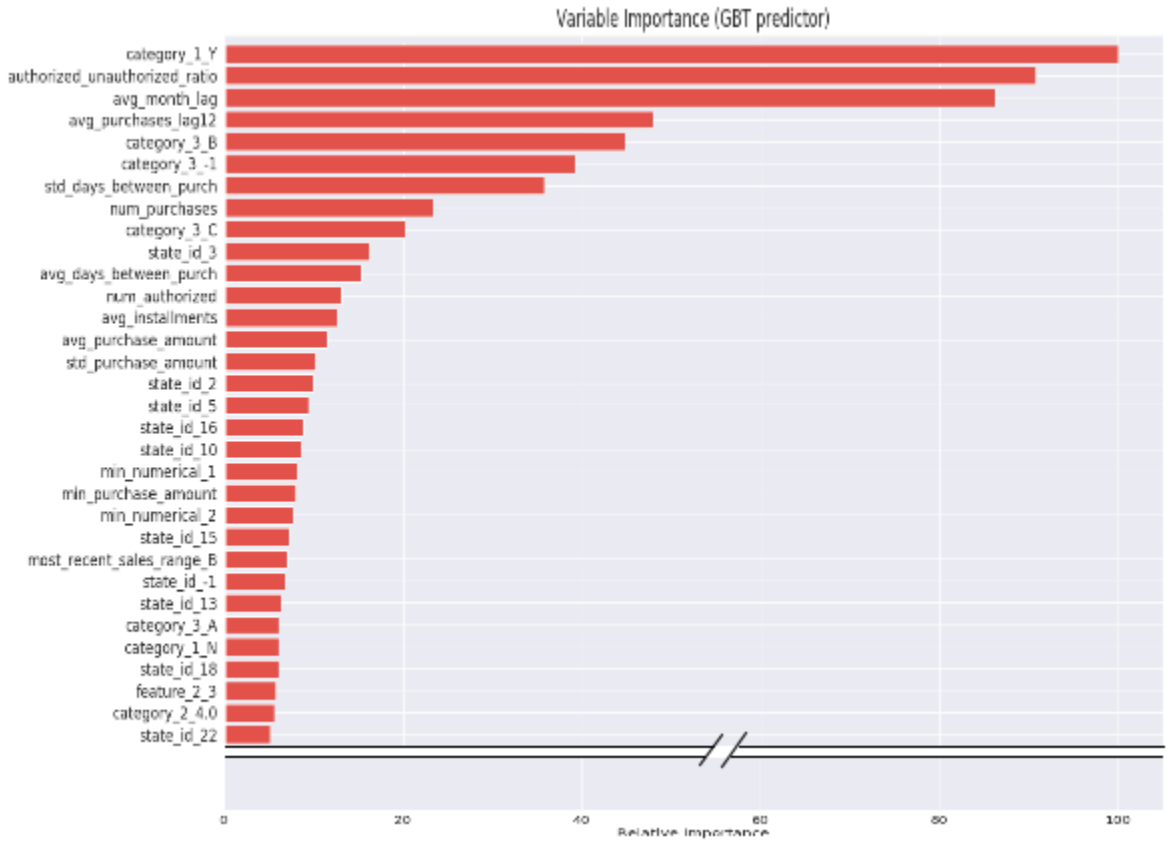
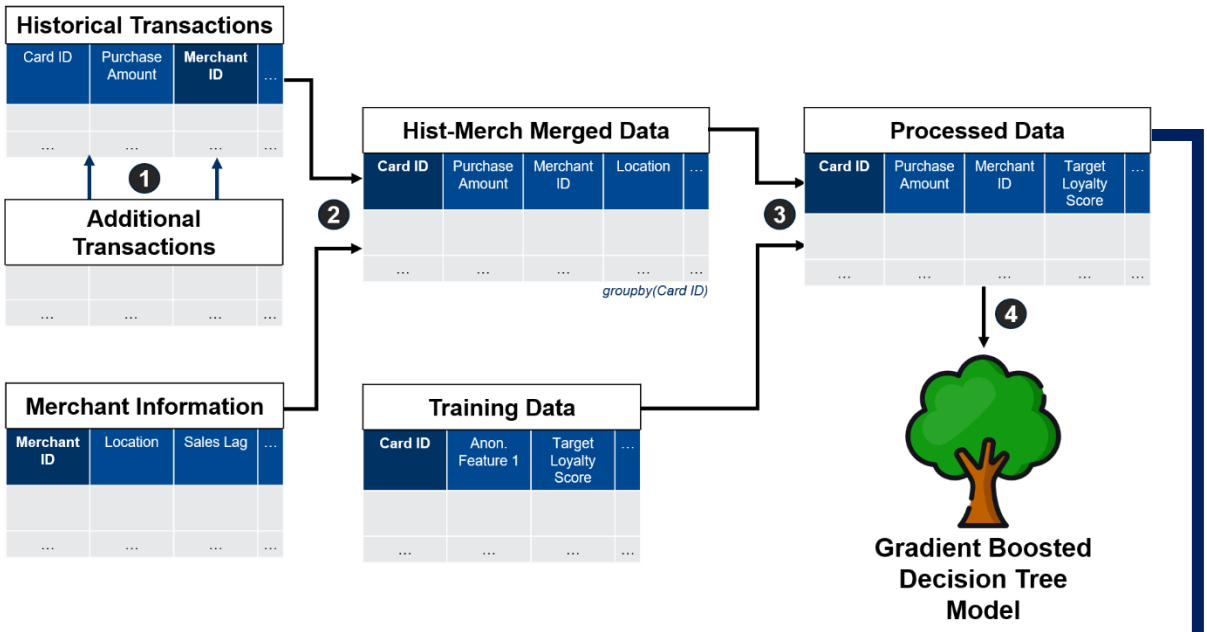


Exhibit 4 – Data Flow Overview for Final Processed Dataset



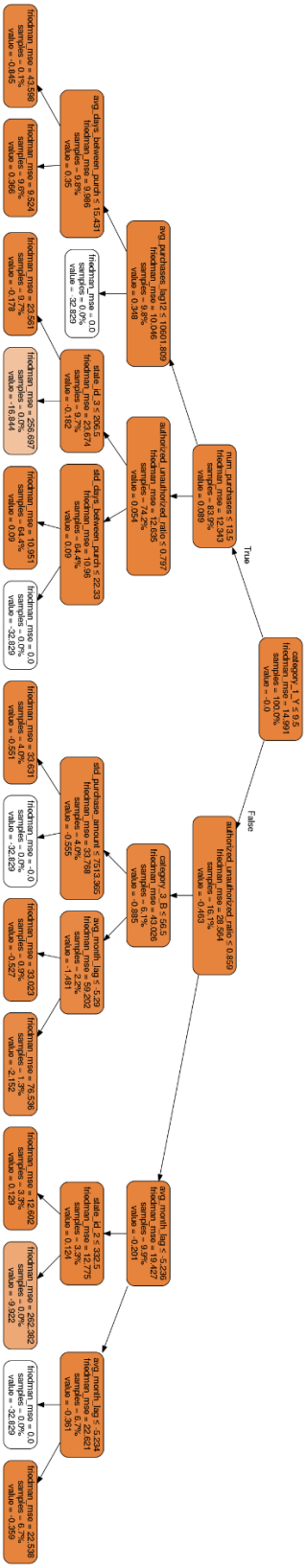
Columns	Description
Card ID	Unique card identfiier
Avg. Month Lag	Average of month lag to reference date
Avg. Installments	Average of number of installments of purchase
Avg. Days Between Purchases	Calculate date difference between purchases then take average
# Authorized Purchases	Number of purchases that were authorized
Authorized Unauthorized Ratio	Ratio of authorized to anauthorized purchases
Avg. Purchase Amount	Average of purchase amounts
Std. Purchase Amount	Standard deviation of purchases amounts
Min Purchase Amount	Minimum of purchases amounts
Max Purchase Amount	Maximum of purchases amounts
# Purchases	Number of total purchases
Avg. Numerical 1	Average of anonymized measure
Std. Numerical 1	Standard deviation of anonymized measure
Min Numerical 1	Minimum of anonymized measure
Max Numerical 1	Maximum of anonymized measure
Average Numerical 2	Average of anonymized measure
Std. Numerical 2	Standard deviation of anonymized measure
Min Numerical 2	Minimum of anonymized measure
Max Numerical 2	Maximum of anonymized measure
Avg. Active Month Lag 12	Average of quantity of active months within last 12 months
Avg. Purchases Lag 12	Average of monthly average of transactions in last 12 months divided by transactions in last active month
Encoded Category 1 (one column per category)	Counts occurences of Category 1 categories
Encoded Category 2 (one column per category)	Counts occurences of Category 1 categories
Encoded Category 3 (one column per category)	Counts occurences of Category 1 categories
Encoded Category 4 (one column per category)	Counts occurences of Category 1 categories
Encoded State ID (one column per category)	Counts occurences of Category 1 categories
Encoded Most Recent Purchases Range (one column per category)	Counts occurences of Purchases Range categories
Encoded Most Recent Sales Range (one column per category)	Counts occurences of Purchase Range categories
Encoded Feature 1 (one column per category)	Counts occurences of Feature 1 categories
Encoded Feature 2 (one column per category)	Counts occurences of Feature 2 categories
Encoded Feature 3 (one column per category)	Counts occurences of Feature 3 categories

Exhibit 5 – Summary of Model Performance and Changes

Dataset	Loss Function	# of Features	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	LR	Batch Size	Epochs	Test loss	Train Loss	Kaggle (Test) Score	Notes
Neural Network															
All variables combined	MSE	542	256, Relu	1 Linear					0.001	256	30	11.29114	0.00286		
Only categorical variables	MSE	531	256, Relu	1 Linear					0.001	256	30	4.38685	0.00202		Baseline model
Only continuous variables	MSE	11	256, Relu	1 Linear					0.001	256	30	11.29261	0.00286		
Only continuous variables + normalized	MSE	11	256, Relu	1 Linear					0.001	256	30	11.29312	0.00286		Normalized continuous variables
All variables combined + normalized	MSE	542	256, Relu	1 Linear					0.001	256	30	11.29174	0.00286		
All categorical + installments	MSE	532	256, Relu	1 Linear					0.001	256	30	4.57586	0.00062		
All Categorical + month_lag	MSE	532	256, Relu	1 Linear					0.001	256	30	3.31512	0.00077		
Found that accuracy is only valid for classification task https://stackoverflow.com/questions/52453079/what-accuracy-function-is-used-in-keras-when-using-metrics-accuracy Created pair plot of all continuous variables and observed trend of visual linearity between all of the average_sales(_purchases)_lag(3,6, 12) columns. Refer to Exhibit 1															
All categorical + installments, month_lag, purchase_amount, numerical_1, numerical_2	MSE	536	256, Relu	1 Linear					0.001	256	30	3.30549	n/a		
All categorical + installments, month_lag, purchase_amount, numerical_1, numerical_2, avg_purchases_lag12	MSE	537	256, Relu	1 Linear					0.001	256	30	2.97648	n/a		Clearly this loss is too low relative to Kaggle scores (3.759 best score), so there must be an error. Must debug and check the data .
Shuffled the data															
All categorical + installments, month_lag, purchase_amount, numerical_1, numerical_2, avg_purchases_lag12	MSE	537	256, Relu	1 Linear					0.001	256	30	2.24326	n/a		MSE is still too low, something still wrong.
All categorical – state_id + installments, month_lag, purchase_amount, numerical_1, numerical_2, avg_purchases_lag12	MSE	231	256, Relu	1 Linear					0.001	256	30	2.03846	n/a		**Hypothesis**: City_id contributed >300 columns to the training set, can remove since state_id and city_id provide similar info. To make sure categorical columns don't overshadow continuous ones
Found out that Keras by default adds an extra weight when running model. Evaluate() if an explicit evaluation function is not specified. To fix added MSE as an explicit function for model evaluation.															
All categorical – state_id + installments, month_lag, purchase_amount, numerical_1, numerical_2, avg_purchases_lag12	MSE	231	256, Relu	1 Linear					0.001	256	30	1.86761	n/a		After fixing MSE calculation, still getting loss that is too low relative to Kaggle
All categorical – state_id + installments, month_lag, purchase_amount, numerical_1, numerical_2, avg_purchases_lag12	RMSE	231	256, Relu	1 Linear					0.001	256	30	1.11223	n/a		RMSE is even lower. Need to submit to Kaggle to determine what's wrong.
Realized that transaction history contains many transactions per Card ID that are not properly interpreted when training. So the current model is not appropriate, must summarize all transactions made by each card id into a single row summary. Feature engineering on transaction history to create summary statistics for each card ID.															
New features– described in data preprocessing report section	RMSE	80	32, Relu	1, Linear					0.001	256	100		3.72410	3.93124	Clearly there is a large gap between train and test sets, so need to increase generalizability. In our case, going to increase dropout (type of regularization technique).
Preprocessed Data	RMSE	80	32, Relu	0.25 Dropout	16, Relu	1, Linear			0.001	256	100		3.72930	3.93118	
Preprocessed Data	RMSE	80	32, Relu	0.5 Dropout	16, Relu	1, Linear			0.001	256	100		3.74110	3.93078	Train error increased, and test decreased. This means we are successfully closing the gap between training and test errors, but very few things to try because the network is so small. **New hypothesis:** increasing network complexity might lower both errors
Preprocessed Data	RMSE	80	64, Relu	0.5 Dropout	32, Relu	0.5 Dropout	16, Relu	1, Linear	0.001	256	100		3.73000	3.93131	Clearly, instead of decreasing both errors, we simply increase overfitting, so need to consider a simpler network
With all of the historical transactions converging to just over ~300,000 rows instead of expected ~29,000,000 (mainly because historical transactions have been summarized with engineered summary features), it makes sense to attempt using simpler models (not NN). One of these models is Gradient Boost Trees – has been recommended on Kaggle forums by many participants.															

Dataset	Loss Function	# of Features				LR	Batch Size	Epochs	Test loss	Train Loss	Kaggle (Test) Score	Notes
Gradient Boosted Decision Tree												
			Number of Estimators	Max Depth	Min Samples Split							
Preprocessed Data	RMSE	80	500	4	2	0.01	n/a	n/a	3.82570	3.71840	3.91527	Best Kaggle score so far
Preprocessed Data - NaNs detected and replaced	RMSE	80	1000	6	2	0.01	n/a	n/a	3.99660	3.64000	3.92388	Test loss has increased while training is much lower, so overfitting again. Going to start tuning by setting a higher learning rate and trying to find optimal number of estimators
Preprocessed Data	RMSE	80	300	4	2	0.1	n/a	n/a	4.00840	3.65210		Overfitting even more now, must further decrease number of estimators.
Preprocessed Data	RMSE	80	100	4	2	0.1	n/a	n/a	3.99330	3.75940		Looking at train vs. test plot, test curve is almost flat and training one is continuously decreasing, so there is overfitting and poor learning over time. Will try to increase sample size per split.
Preprocessed Data	RMSE	80	100	4	30	0.1	n/a	n/a	3.99020	3.77620		Further increasing number of samples per split
Preprocessed Data	RMSE	80	100	4	100	0.1	n/a	n/a	3.98990	3.78340		Further increasing number of samples per split
Preprocessed Data	RMSE	80	50	4	200	0.1	n/a	n/a	3.98940	3.78590		Further increasing number of samples per split
Preprocessed Data	RMSE	80	50	8	200	0.1	n/a	n/a	3.98580	3.72840		
Preprocessed Data	RMSE	80	100	16	200	0.1	n/a	n/a	3.99990	3.25120		Depth adjustments causing overfitting
Preprocessed Data	RMSE	80	50	4	100	0.1	n/a	n/a	3.74150	3.82990	3.91345	Best model so far

Exhibit 6 – Gradient Boosted Decision Tree Diagram



References

Barušauskas, D. (2019). Towards de-anonymizing the data! Some insights. Kaggle. Retrieved March 20, 2019 from https://www.kaggle.com/raddar/towards-de-anonymizing-the-data-some-insights/