Ecommerce Customer Retention using Machine Learning: Predictive Analytics to Determine Customers Who May Churn

Margaret Catherman
April 16, 2022

Abstract

Ecommerce websites have seen significant growth over the past ten years; with a notable spike in growth since 2020 due to the pandemic, which caused many consumers to forsake in person shopping at store front retailers and instead shop online. Customer retention is integral to online retailers' success; therefore, these businesses conduct extensive research on predicting, in advance, which customers might leave. These findings are used in general business metrics, such as forecasting of Return on Investment, monthly recuring revenue, among others; as well as for the purpose of identifying customers to target with marketing campaign to attempt to prevent attrition. Ecommerce websites are rich with big data offering a plethora of metrics on customer's behavior, making machine learning an integral tool to their analysis. While providing a rich source for analysis, these data sets have inherent challenges: of the many metrics, some will be irrelevant, others may have high correlation with metrics other than Churn; some may have missing values; their scale may be vastly different; or some may be skewed. These data attributes must be considered in selection of the best machine learning models to evaluate the ecommerce site, with special emphasis on parameter tuning and feature selection for optimal model performance.

Introduction

The Objective: This analysis will explore several models to see which one is best at predicting churn with the ecommerce data. The models will be optimized with parameter tuning to enhance performance. Two approaches for dealing with missing values will be evaluated: imputation and omission, to determine which approach increases the model's performance. Finally, features that are most influential in predicting if a customer will Churn will be identified. Reducing features to the most influential may assist in improving model performance for some models. Identifying influential features also has application to broader ecommerce business strategies, as will be discuss in the "Conclusion."

.

The Data:

This is an ecommerce data set, reflecting customer behaviors and characteristics. The original data set is available on Kaggle, and is comprised of 5630 rows and 20 columns. [1] For this analysis, Churn is the binomial response variable. The data is comprised of one dependent and 19 predictors; of which eleven are categorical. The original variables and their description are seen in Figure 1, Appendix.

Domain Knowledge:

Churn: Its meaning and relevance to an ecommerce site: Churn in an ecommerce setting refers to customers defection, attrition or turnover. It is the opposite of retaining a customer. It occurs when a customer stops using the website for a period of time or cancels their subscription. Online retailers rely on repeat business for a significant portion of revenue. A high customer churn rate impedes profits and decreases a business's return on investment (ROI). It may be affected by seasonality, and is generally measured in monthly and annual rates. Churn has a direct impact on other business metrics, such as monthly recurring revenue (MRR), customer lifetime value, and customer acquisition costs.

What metrics might corelate with Churn? Let us first consider the causes of churn. Research has shown there are three primary reasons for customer churn: 1) Poor onboarding; that is, confusion in how to

navigate the site; 2) Weak customer relations: 56% of customers churn is due to poor treatment and 3) Poor customer service, accounting for 82% of churn. [2]

We will keep this summary in mind as we examine the ecommerce data set. It appears the data may be lacking in some measurement of metrics relevant to churn listed above, although perhaps the combined parameters of our data set will suffice. The "Satisfaction" score with a scale of 1 to 5 should be helpful, as might "Complain" (0,1). Additional metrics to track poor onboarding, such as web site page navigation and the buyer's journey would be helpful. It is also not clear when and what the "Satisfaction" and "Complain" metrics of the ecommerce data are referencing; more information on these would be helpful as well.

Data Exploration, Analysis & Preparation

Data Exploration & Analysis:

Charts are helpful at visualizing general relationships between variables. For example, in Figure A, the bar chart for Complain, at left, shows a larger percent of customers that complain churn. We can see from the Gender/Churn chart, at right, that a larger percent of men churn than woman: about 25% vs. 20%.

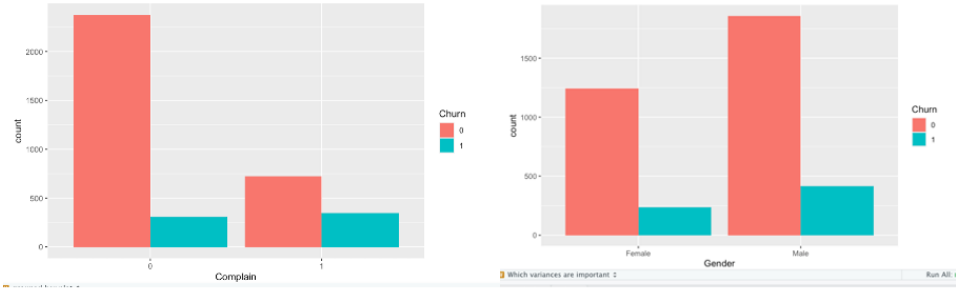

Figure A. Bar chart for Complain, at left, Gender/Churn chart, right.

In Figure B, Pearson's Correlation of highly correlated features reveals challenges of the ecommerce data due to the high number of predictors, made even higher by adding dummy variables for categorical variables. Introducing so many dummy variables increase the total number of variables, many of low relevance to predicting Churn. The chart reflects all correlations with a significance over .50. Note Churn does not appear here, indicating there are no predictors with a significant correlation to Churn. It could be argued, however, that this is incorrect, and there are predictors of significance to predict Churn. The problem is that the inclusion of highly corelated plus low relevance variables combine to dilute the correlation chart, leaving it of little value. Fortunately, Random Forest, Boosting and Stepwise Regression offer methods to assist with feature importance, which we will explore shortly. Note this chart indicate high correlation between several features, which may cause collinearity, which we will which we will discuss in more detail in "Feature Selection". Predictors highly corelated with each other are: "Order Count" & "Coupon Used"; Marital Status "Single" & "Married'; also Preferred Device "Mobile" & "Computer." For each of these pairs, one member should be removed when using models to predict Churn. We will keep this in mind as we perform feature selection, discussed shortly.

var1 <chr>	var2 <chr>	value <dbl>	abs_cor <dbl>
OrderCount	CouponUsed	0.7735315	0.7735315
MaritalStatus.Single	MaritalStatus.Married	-0.7330222	0.7330222
PreferredLoginDevice.Mobile.Phone	PreferredLoginDevice.Computer	-0.6303958	0.6303958
PreferedOrderCat.Others	CashbackAmount	0.5629242	0.5629242
PreferredPaymentMode.E.wallet	CityTier	0.5304556	0.5304556
PreferedOrderCat.Grocery	CashbackAmount	0.5257094	0.5257094
DaySinceLastOrder	OrderCount	0.5209692	0.5209692
PreferredLoginDevice.Phone	PreferredLoginDevice.Mobile.Phone	-0.5188943	0.5188943
PreferredPaymentMode.Debit.Card	PreferredPaymentMode.Credit.Card	-0.5029775	0.5029775

Figure B. Pearson's Correlation of highly correlated features.

In Figure C, a bar chart illustrates the Skewed Feature Ranking for the ecommerce data. Positively skewed features have values greater than 0; negatively skewed have values less than 0. The farther the absolute value is from 0, the greater the feature is skewed. A skew value of 0 indicates a normal distribution. This is relevant to certain models, as some do not work well with skewed data. See Appendix Figure 2, for histograms of features, offering a visual illustration of metrics in this graph.

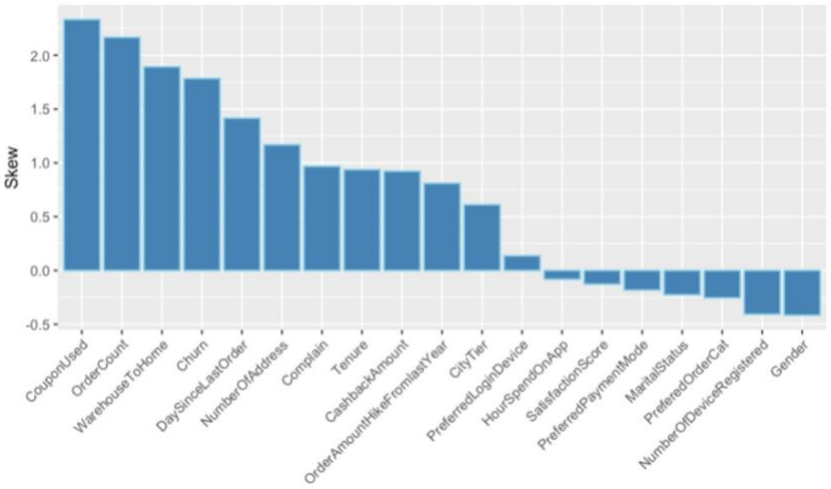

Figure C. Skewed Feature Ranking of the ecommerce data.

Data Preparation:

- **Dummy Variables** were created for the five categorical character columns, resulting in a data frame 1856 x 36, or 16 new columns. Again, some of the models used in this analysis can accommodate numeric categorical variables; however, as others do not, dummy variables were created. Dummy variable assignment was made with a new variable for each category, as is recommended for classification, not category minus one as is preferable for other situations.
- **Training Data/Testing Data Division:** Observations were randomly selected for each. The training data set was 77.5% of the original data less the missing values; the testing data set was the remaining 33.5%. Data was split before the assignment of missing values and standardization to prevent leakage, or over influence of the training data on the test data.
- **Missing Values:** There were 1856 rows with missing values, of the 5630 original rows. This is a large percentage; however, for this assignment, all rows with missing values have been removed, leaving 1856 rows and 20 columns. It should be noted that Random Forest can accommodate missing values; however, as the other methods used do not, we will omit missing values. See more information on the missing value analysis, below, as well as in Figures 3, Appendix.
- **Standardize Data:** The data is scaled to a fixed range of 0 to 1. This results in smaller standard deviations, helps to suppress effect of outliers, and prevents the model from assigning heavier importance to variables whose values are at a much larger scale than others. (See Figure 4, Appendix)

Missing values

As noted earlier, one of the objectives of this project was to compare the effect two different approaches to missing values have on the models' performance. The first approach will be to omit observations with missing values, the second will be to apply imputation (replace values) with missForest, prior to standardization. Thus, the five models used, Random Forest, Boosting, LDA, Naïve Bayes, Classification Tree, used for this Project will be run once with missing values omitted and a second time with the imputed values. To determine the impact on model performance, error rates will be compared using the test data. A detailed review of the results is in "Findings", below.

Overview of methods to address missing values: Broadly stated, missing data may be delt with in four ways: 1) Discard the observations: which is acceptable if the percent of missing data is small; to be avoided otherwise; 2) Count on the algorithm to deal with the missing values; 3) Impute missing values before training; [3] or 4) Create a third category of "missing".

The two methods explored in this project, omission and imputation, have inherent risks to the accuracy of the analysis. Omitting observations means a reduction of relevant data, which is particularly problematic if the data set is small. On the other hand, with imputation, estimates are made and assigned, but these estimates may be inaccurate. Thus, both approaches may cause poor model performance, either due to inadequate or inaccurate values.

If imputation is chosen, values may be assigned systematically, such as the mean; or the same value as the one before or after the missing value may be assigned. Fortunately, there are also more sophisticated options for imputation provided in statistical software packages, such as R and Python. These options are compelling, as they include analysis of the entire data set and apply statistical methods used for prediction to determine imputation, such as logistic regression, linear regression, bootstrapping, random forest, among others. Attributes of the methods to be used should be considered when selecting an imputation method, as some are parametric, that is, have assumptions and requirements about the data, such as requiring it to be normally distributed; others are non-parametric, thus able to take all data types.

As was observed in the data exploration and analysis, most of the variables in the ecommerce data are non-parametric, and either positively or negatively skewed, or lacking normal distribution. About half of the predictors are categorical, the other half numeric. The response is binomial, thus a classification analysis. Furthermore, there is large variance among the variables. With this in mind, we evaluated several imputation options to determine which to use.

Several imputation methods were considered: Amelia II, Hmisc and Multiple Imputation with diagnostics (MICE), but rejected either due to incompatibility with the ecommerce data set or for other reasons. A brief summary of these methods and the reason for their omission is in the Appendix, Figure 7.

missForest (Stekhoven & Buhlmann, 2011) was selected for imputation for this project, as it works well with mixed-type data, such as categorical, numeric, non-linear and nonparametric. It is therefore well suited for the ecommerce data, with its mix of categorical and continuous variables, most of which have skewed distribution. [12] missForest trains a random forest algorithm on the training data, using all features to predict the missing ones. It iterates through the data set, stopping as soon as the error rate increases over the past iteration, then provides the next to last run of imputated values. Cross-validation is not necessary, as it is built into the random forest process. As with Random Forest, missForest allows for parameter tuning of mtyr, ntree, nodesize, maxnodes. [4]

Also similar to Random Forest, missForest provides out of bag error (OOB) estimates of imputation, for features individually as well as the set as a whole. For categorical missing values, the OOB error is the proportion of falsely classified (PFC) missing values. For continuous missing values, the OOB error estimate is the normalized root mean squared error (NRMSE) defined as:

$$\sqrt{\frac{mean((X_{true} - X_{imp})^2)}{var(X_{true})}}$$

where Xtrue is the complete data matrix, Ximp the imputed data matrix. Here, “mean/variance” represents the empirical mean and variance computed over the continuous missing values only. [4]

The ecommerce data has seven predictors with missing values (See Figure 3, Appendix). Unfortunately, it was not possible to get the imputation error rate below 12.24 % for the training data, and 15.55 % for the test data, as can be seen in the chart in Figure D. In this chart, “OOBerror.fit”, reflect errors during the fitting of the missForest model on synthetically generated missing values, so while it illustrates the model’s general performance, is not directly applicable to this analysis. “OOBerror.tr” and “OOBerror.te” reflect the error rate on imputed values for the training and test data, respectively. This high error rate for

imputation meant omission was more effective at enhancing model performance. See “Findings & Results” for more details on the impact of imputation on model performance.

Feature <chr>	OOBerror.fit <chr>	OOBerror.tr <chr>	OOBerror.te <chr>
Mean_OOB_Error	8.33	12.24	15.55
WarehouseToHome	64.3033	44.397	53.7915
Tenure	35.335	24.3127	33.5442
OrderAmountHikeFromlastYear	10.8096	7.9946	10.3297
DaySinceLastOrder	6.3672	4.9928	6.6171
OrderCount	2.0307	1.9133	2.7575
CouponUsed	1.4483	1.8747	1.5364
HourSpendOnApp	0.2553	0.2025	0.2634
Churn	0	0	0
CityTier	0.4471	0	0

Figure D. The imputation error rate. Training data: ‘tr’; test data: ‘te.’

.

Methods

The analysis began by fitting both random forests and boosting models on the training data, with the objective to ultimately increase the models’ predictive performance on the testing data. After several rounds of parameter tuning, an optimal model was selected for each method, as measured by a decrease in the error rate on the training data. Predictor importance ranking by the optimized random forest, boosting and a logistic regression model were synthesized to determine the most significant predictors of the response variable, Churn. Finally, the top nine most significant predictors were applied to the five models in this study. The models were then fit to the test data set, the error rates were determined and compared. Model results, comparisons and conclusions follow in subsequent sections. Following please find a brief summary of the models used, followed by a detailed account of the parameter tuning and predictor selection process.

Overview of Models Used

Random Forest is an ensemble of decision trees; the algorithm predicts new data by aggregating the predictions of the trees. It picks a bootstrap sample of input variables from the training data set. From these, it makes a tree of size square root of p variables for classification or p/3 for regression (default values). [8]

The **Boosting Algorithm** applies logistic regression techniques to the AdaBoost method by minimizing the logistic loss. Unlike random forests, that builds ensembles deep independent trees, here the ensembles are of weak, shallow and successive, with the trees improving and learning from the previous. However, together, these trees provide a strong committee that out performs many other algorithms. One of the main ideas is the sequential addition of new models. Each new weak, base model as it is added is trained with the error learned by the ensemble thus far. [6]

Stepwise algorithm with Logistic Regression: This was used to assist with feature selection only. We will use the stepwise function in the R package, which combines both forward and backward stepwise: that is, at each step it considers either adding or dropping a variable, based on the one that lowers the AIC score. While we will be using this to assist in model selection, it should be noted that this is a *controversial practice*, as the AIC errors used are not accurate, as they do not account for the search process (ESL, 3.3.2, p. 60; suggests consider bootstrap for these situations.) We therefore will proceed with caution, and use it as one of several references, along with boosting and random forest, and

synthesize the findings to create the optimal model. Stepwise does not work well with missing values; thus, they have been removed from the Ecommerce data. [7, 7a]

Linear Discriminant Analysis (LDA): A Linear Classification method developed by Fisher (1936), LDA tries to approximate the Naïve Bayes method. It assigns an observation to the class for which the posterior probability *pk*(X) is greatest. So, for a binomial response such as Churn, the observations will be assigned to the default class if Pr (Chrun =Yes|X = x) > 0.5 [8]

Naive Bayes: Based or Bayes theorem, this is a probabilistic classifier with emphasis on the assumption of independence between the features. [9] A weakness of NB: it assumes, often incorrectly, that features have equal importance and are non-corelated; this can lead increase in bias of the posterior probabilities.) Naive Bayes introduces some bias, but reduces variance, does well in with a variety of data sets, even those where n, the number of observations, is small relative to the number of p, predictors. [10]

Classification Tree algorithm recursively partitions the predictor space into subsets in which the distribution of the dependent variable is successively more homogeneous.

Parameter Tuning

Parameter tuning and selection were important considerations for optimizing the model performance. It involved trying a range of values to determine the characteristics that best classify the response variable, in this case, Churn. Following is a detailed explanation of how this was accomplished and why certain parameters were chosen for each model, using the ecommerce training data set.

Random Forest Parameter Tuning: Tuning parameters involved trying various combinations of values for each parameter. For Random Forest, this included selecting the number of trees to grow. Initially Random Forest 1, with all parameters set on their default values, was run. mtrym, or the number of variables randomly sampled as candidates at each split, was set at 5.83, or the square root of p, where p = 34 = the number of predictors in this case. The default value for nodesize is 1 for classification; ntree, or number of trees grown is 500 by default. [8] The goal of parameter tuning is to try adjusting the model parameters to see if this error rate could be improved on, or decreased. [5] The Random Forest 1 model resulted in a testing error rate of 0.0381558.

Initial tuning with Random Forest: Ultimately a complex hyperparameter grid with a loop to determine the optimal values for the parameters would be used. However, initially the built-in function from the randomForest library was run to see what optimal value for mtry would be generated. For this analysis, the parameters were set as follows: ntreeTry to 500; mtryStart to 4; stepfactor to 1.5; improve to .01; trace to FALSE and type to class for classification. This generated an optimal number for mytry of 19, as this s the point at which the value for error is lowest, as illustrated in Figure E. This value was used to guide the selection of the range of values for mtry to consider in the hyperparameter grid.

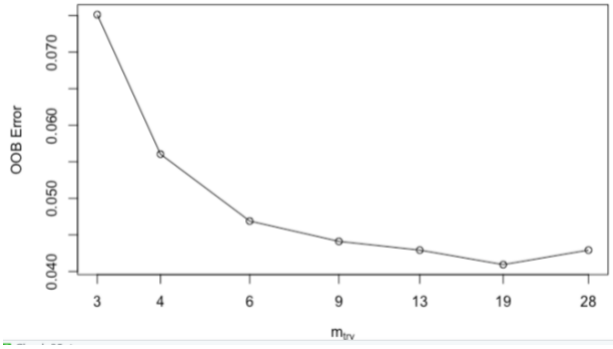

Figure E. Random Forest: Mtry Values for Optimal Error Rate

Detailed grid search: To consider a range of values for each parameter, with the ultimate goal of determining which combination of values resulted in the best model performance, a hyperparameter grid using the ranger library was used. The library chosen was ranger, a C++ implementation of Brieman's random forest algorithm, because its processing time is superior to that of randomForest. [1a] The grid contained the various parameters with a range of values for each, to be tried in a loop, for a total of 195 combinations of values. Parameters and their value ranges were entered as follows:

- num.trees: The number of trees to grow. The default is 500; a sequence of 500 to 1500, by 500, was set. (While the optimal was at 1500 trees, this may be at the cost of creating an unnecessarily long run time.)
- mtry: The number of variables randomly sampled as candidates at each split. For classification, the default is the square root of p, where p is the number of predictors (p/3 for regression)**.** For this analysis there are 34 predictors; the square root of 34 is 5.83. However, as the optimal value for mtry was determined to be 19, as described above, a range of values was explored on either side of 19, therefore setting mtry to a sequence from 17 to 23 by 2's. While 19 generated the lowest error, this range represented values creating the next lowest error rates, see graph, above.
- node_size, minimum number of samples in the terminal nodes. This is a bias/variance trade off. The default value is 1 for classification and 5 for regression. I set a sequence from 3 to 9, by 2's.
- sample_size: The number of samples to train on. The default value is 63.25% of the training set. Lower sample size may increase bias, while lowering run time. Increasing the sample size may help performance, although there is a risk of overfitting, as it introduces more variance. Generally, the sample size is in the 60-80% range. For these reasons, a range of .55, .632, .70, .80 was entered.
- Seed set at 123. [5]

A loop was used to generate 195 combinations of these parameter values. To determine the optimal model, each model was ranked in ascending order by the OOB.RMSE error rate, which is the out-of-box square root of the model's prediction error. (See Figure 8 in the Appendix for a Histogram of the OOB_RMSE frequency.) The parameter combinations with the 10 lowest error rates are illustrated in Figure F. The lowest error rate of 0.222894 was a tie, with mtry at either 19 or 21; 19 was used for the model, which was labeled Random Forest 2. The remaining optimal parameter values were: the number of trees: 1500, the terminal node size of 3 observations, and a sample size of 80% of the training data. The hypothesis was that these optimally tuned parameters in Random Forest Model 2 would improve the model's performance. Unfortunately, this was not the case. The parameter tuning of Random Forest Model 2 did not have the desired result of decreasing the error rate; see the table in "Findings", below. Continued work is needed to resolve this issue. It may be a result of improper synthesis of the two libraries used to perform the hyperparameter loop; or some other error.

	num.trees <dbl>	mtry <dbl>	node_size <dbl>	sampe_size <dbl>	OOB_RMSE <dbl>
1	1500	19	3	0.8	0.2228947
2	1500	21	3	0.8	0.2228947
3	1500	17	3	0.8	0.2237845
4	1000	19	3	0.8	0.2237845
5	1000	17	3	0.8	0.2246708
6	1000	21	3	0.8	0.2255535
7	500	21	3	0.8	0.2264329
8	500	23	3	0.8	0.2264329
9	1500	23	3	0.8	0.2264329
10	500	19	3	0.8	0.2273089

Figure F. Random Forest Classifier: Models with optimal parameter tuning combinations; ranked by the 10 lowest error rates.

Boosting Parameter Tuning: The parameter tuning process was initiated with the distribution set to Bernoulli, used for classification, and other parameters set on the default values, which are: number of trees: 5000; shrinkage: .01; interaction depth: 3; and cross validation folds: 10. Parameter tuning with Boosting is somewhat similar to that used for Random Forest. Once again, a hyperparameter grid was used with a range of values for the parameters, with a loop iterating these values. In this case there were 400 possible combinations. The built- in function in the glm library was

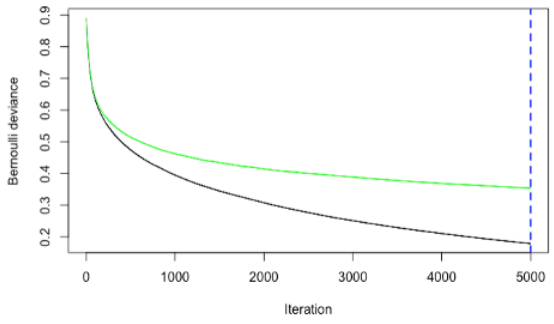

Figure F. Boosting: Optimal Iterations 5000.

used to determine the optimal number of iterations. The function returned an optimal value of 5000 iterations, or base trees, as indicated by the blue dashed line in Figure F. The black and green lines show the deviance of training and testing data. The 5000 iterations, or trees, was used as an option in our hyperparameter grid.

The next step for Boosting parameter tuning was the first round of iterations through the various values in the hyperparameter grid. The parameters tuned and the range of values tried are listed, below:

- **Distribution:** 'Bernoulli', used for classification**.**
- **Number of trees:** The total number of trees to fit, or iterations: 5000. GBMs often require many trees; however, unlike random forests, GBMs can overfit, therefore the goal is to find the optimal number of trees that minimize the loss function of interest with cross validation.
- **Depth of trees,** or interaction depth. The range of 1, 3 or 5 was entered. The number of *d* splits in each tree controls the complexity of the boosted ensemble. Often d=1works well, in which case each tree is a *stump* consisting of a single split. More commonly, d is greater than 1 but it is unlikely d>10 would be required.
- **Learning rate,** or shrinkage. The range of .01, .1, or .3 was used. The learning rate controls how quickly the algorithm proceeds down the gradient descent. Smaller values reduce the chance of overfitting, but also increases the time to find the optimal fit.
- **Subsampling,** or bag fraction, controls whether or not a fraction of the available training observations are used. The range of .65, .8, or 1 was entered into the hyperplane grid. Using less than 100% of the training observations is referred to as implementing a stochastic gradient descent. This can help minimize overfitting and prevent getting stuck in a local minimum or plateau of the loss function gradient. [6]

In Figure G we we see the top ten results of the iterations:

	shrinkage <dbl>	interaction.depth <dbl>	n.minobsinnode <dbl>	bag.fraction <dbl>	optimal_trees <dbl>	min_RMSE <dbl>
1	0.3	5	5	1.0	312	0.4958292
2	0.1	5	5	1.0	1221	0.5055278
3	0.1	5	10	1.0	857	0.5063090
4	0.3	3	5	1.0	528	0.5069493
5	0.1	5	5	0.8	1063	0.5187142
6	0.1	3	5	1.0	1475	0.5226770
7	0.3	3	10	1.0	552	0.5326193
8	0.1	3	5	0.8	1343	0.5328580
9	0.1	5	10	0.8	848	0.5367445
10	0.1	3	10	1.0	1642	0.5370613

Figure G. Boosting/Gradient Descent: Models with optimal parameter tuning combinations; ranked by the 10 lowest error rates.

Parameter Tuning Round 2: The parameters were fine-tuned with a second iteration through the hyperparameter grid, using the values from the top ten combinations in round one, as well as values slightly higher and lower. For example, in Figure G, the top ten combinations for shrinkage were with values at .1 and .3. This range will be extended slightly this time, to .08, .1, .2, .3, .4. Interaction depth performed well at 3 and 5 in round one; for round two, values of 2, 3, 5, 7 will be tried. n.minobsinnode did well at 5, 10; for round two, 4, 5, 7, 10, 11 were entered. Bag fraction did well at .8 and 1; these values will be included again, as well as .7 and .9

	num.trees <dbl>	mtry <dbl>	node_size <dbl>	sampe_size <dbl>	OOB_RMSE <dbl>
1	1500	19	3	0.8	0.2228947
2	1500	21	3	0.8	0.2228947
3	1500	17	3	0.8	0.2237845
4	1000	19	3	0.8	0.2237845
5	1000	17	3	0.8	0.2246708
6	1000	21	3	0.8	0.2255535
7	500	21	3	0.8	0.2264329
8	500	23	3	0.8	0.2264329
9	1500	23	3	0.8	0.2264329
10	500	19	3	0.8	0.2273089

Figure H. Parameter Tuning Round 2 reduced the error rate on the training data.

Parameter Tuning Round 2 produced better results, as seen

in Figure H. The final Boosting Model will use the value combination with the lowest error rate, which is: shrinkage of .3, interaction depth of 5, n.minobsinnode of 5, bag fraction of 1, iterations of 312, and 10 rounds of cross validation.

In summary, optimal parameter tuning using a hyperparameter grid, the ranger library & a loop for both Random Forest & Boosting, as described above, resulted in the parameters seen in Figure I. For results on how these optimized models compared against models using default values, please see "Findings & Results."

Optimal parameters Random Forest		Optimal parameters Boosting	
• mtry of	19	shrinkage at	.3
• number of trees is	1500	Interaction depth at	5
• terminal node size of observations	3	n.minobsinnode at	5
		bag fraction at	1
• sample size of the training data.	80% of	iterations at	312
		10 rounds of cross validation.	

Figure I. The optimal parameters for Random Forest and Boosting.

Feature Selection

Feature selection is a critical component of model optimization for some models. Two areas of particular importance are 1) Omitting week, irrelevant predictors; 2) Collinearity: Identifying, and removing, a parameter that is highly corelated with another.

For this project, a five-step approach was used to choose the top nine features: a consensus among metrics provided by 1) Random Forest, 2) Boosting, 2) Stepwise, along with 4) A check of correlation between the features, to prevent collinearity; and finally, 5) Domain knowledge, or common sense, as needed, to break any ties or give further thought to model generated rankings. Details on these metrics and findings are provided, below.

The Random Forest and Boosting models each provide a summary ranking the relative importance of features. The top ten variables most influential from each model are reflected in Figure J. There is a consensus on nine features. Two features were not listed in both methods: "Number of Devices Registered from RF", and "Marital Status from Boosting", so we will omit these. In Figure J, the general check of the Stepwise ranking reveals these nine have p-values of significance, as lower values indicate greater relevance. The correlation chart in Figure X shows no two of these nine features are highly corelated. Finally, we reflect on our Domain Knowledge: it seems reasonable that these features will be influential on Churn, and are unlikely to exhibit collinearity. In this manner, the nine predictors are selected. A second run of the models will be made with only these nine features to see if this will improve performance.

Random Forest		Boosting	
Tenure	249.94130	Tenure	37.05453919
CashbackAmount	70.09870	CashbackAmount	9.06248470
NumberOfAddress	61.09667	Complain	7.88826591
WarehouseToHome	57.07053	NumberOfAddress	7.30042381
DaySinceLastOrder	53.77676	DaySinceLastOrder	5.46938052
Complain	50.84332	WarehouseToHome	5.22657380
OrderAmountHikeFromlastYear	41.67930	SatisfactionScore	4.16949403
SatisfactionScore	39.49538	OrderAmountHikeFromlastYear	3.37558393
NumberOfDeviceRegistered	27.81674	OrderCount	2.69099570
OrderCount	22.87587	MaritalStatus.Single	2.47959887

Figure J. Features ranked by influence

The top nine predictors were: Tenure, Number of Address, Cash back Amount, Warehouse to Home, Day Since Last Order, Complain, Order Amount Hike From Last Year, Satisfaction Score, Number of Devises Registered. (See also Figures 5 & 6 in Appendix for bar charts of these findings.)

>Stepwise Parameter Ranking by p-value: The table in Figure K ranks the nine features with the lowest p-value. Observations of interest: Tenure scored significantly higher in terms of performance in both Random Forest and Boosting, and had the lowest p-value in the Stepwise analysis. It would be interesting to run the models again, with Tenure as the sole predictor, as its rank is significantly and consistently the highest by all three models. This high ranking does make me wonder: how are the values for Tenure and Churn arrived at? Could there be collinearity between the two? Tenure's extremely low p-value of 1.291526e-67, almost zero, further supports this possibility. Additional research is needed on how Churn and Tenure are derived, although a definitive answer may not be possible without access to in-house domain knowledge.

Predictor <chr>	p.value <dbl>
Tenure	1.291526e-67
Complain	5.250898e-35
NumberOfAddress	5.707663e-13
NumberOfDeviceRegistered	5.527669e-12
SatisfactionScore	7.722864e-08
WarehouseToHome	1.642439e-03
DaySinceLastOrder	2.253379e-03
PreferredLoginDeviceComputer	7.023591e-03
(Intercept)	7.591534e-01

Figure K. Stepwise Ranking of Nine Predictors by p-value, ascending order.

With this consensus for the top nine predictors out of a total of 34, these nine predictors were used to evaluate the remaining models. The other fourteen, lower ranked predictors were omitted. This was done because decreasing less relevant predictors can help improve a model's performance. See Findings and Results for performance analysis. Unfortunately, imputation had mixes results, as discussed, below.

Findings & Results

Which model yields the best performance in term of smallest testing error on the ecommerce data? Test errors for each model are illustrated in the Figures L and M. Lower error rates reflect better model performance. Overall, the best preforming model was optimized Random Forest on all 34 predictors, with missing values omitted; feature reduction made little difference. *Analysis of parameter tuning:* Boosting showed a slight improvement with parameter tuning (Bst1 vs Bst2) particularly when missing values were omitted. Random Forest was virtually unchanged. *Analysis of Imputation:* NB was the only model where performance was improved with imputation, but only when features were not reduced.

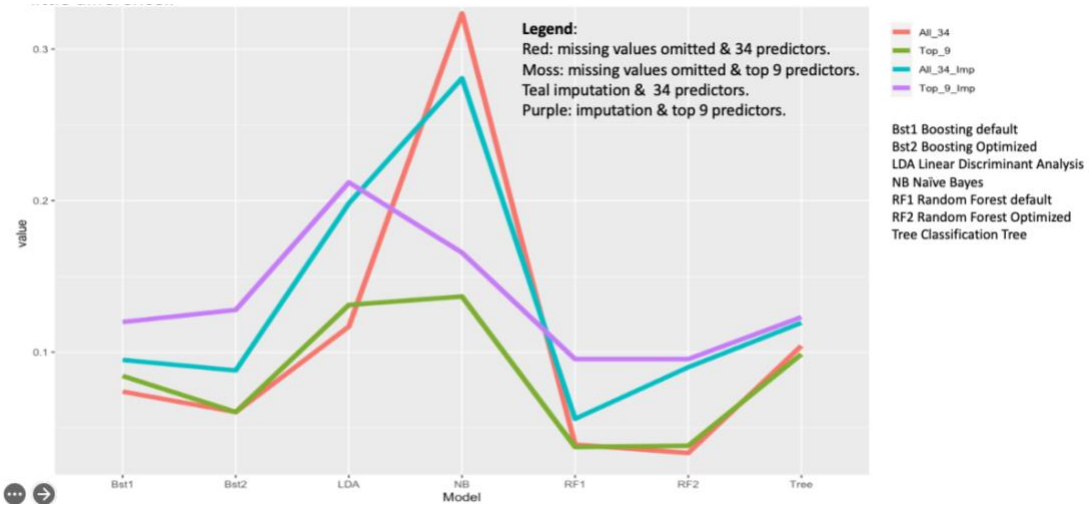

Figure L. Line graph of table in Figure M. The model with the lowest error rate was optimized Random Forest with all 34 predictors, with missing values omitted, seen as "RF2" in red.

Both Figures L and M compare the test error rate among models in terms of feature reduction & missing values omitted vs. those imputed. The greatest error reduction occurred when features were reduced for Naïve Bayes: (-57%) for omitted MV; second was Naive Bayes (-40%) with imputation. Errors were reduced slightly when features were reduced & missing values were omitted for Classification Tree (-5%)

and RF1 (-.4%). For all other imputed models, reducing features *increased* the error rate: for RF1 by 70%; Boosting 2 by 45%; Boosting 1 by 26%. The reason for this discrepancy between the models will be expanded on further in the conclusion. However, here I will address an unexpected factor that accounts for this, unique to this analysis: Recall, the imputation error rate was 12.2% and 15.55% for testing & training data, respectively. Among the top nine features, those with imputed values account for 4 of the 9, vs. only 7 of the full 34 features. (See below, and Appendix.) As a higher portion of the features in the top nine are imputed, the imputation error rates outweigh the benefits of narrowing features down to most relevant features. The top nine features; ones with those imputed, in bold:

X <chr>	All_34 <dbl>	Top_9 <dbl>	Delta <dbl>	Delta_Percent <dbl>	All_34_Imp <dbl>	Top_9_Imp <dbl>	Delta_Imp <dbl>	Delta_Percent_Imp <dbl>
Bst1	0.07392687	0.08426073	0.0103	0.1393	0.09483218	0.11987214	0.0250	0.2636
Bst2	0.06041335	0.06041335	0.0000	0.0000	0.08790623	0.12786361	0.0400	0.4550
LDA	0.11685215	0.13116057	0.0143	0.1224	0.19818860	0.21204049	0.0139	0.0701
NB	0.32352941	0.13672496	-0.1868	-0.5774	0.28076718	0.16568993	-0.1151	-0.4099
RF1	0.03895072	0.03736089	-0.0016	-0.0411	0.05594033	0.09536494	0.0394	0.7043
RF2	0.03338633	0.03815580	0.0048	0.1438	0.09003729	0.09536494	0.0053	0.0589
Tree	0.10413355	0.09856916	-0.0056	-0.0538	0.11933937	0.12306873	0.0037	0.0310

Columns 1 to 4: missing values were **omitted**;
Columns 5 to 8: imputation.
Delta: (Top_9 – All_34): the amount the error rate changes due to feature reduction.
Delta_Percent (Delta as a percent of All_34) the percent change to the error rate due to feature reduction.
Delta_Imp and **Delta_Percent _Imp**: similar to Delta & Delta-Percent, for imputed models.

Missing Values = MV
All_34: MV omitted & 34 predictor
Top_9: MV omitted & 9 predictors
All_34_Imp: imputation & 34 predictors.
Top_9: Imp: imputation & top 9 predictors.

Figure M. Table comparing the test error rate among models in terms of feature reduction & missing values omitted vs. those imputed. Values are illustrated in line graph in Figure L.

> **Tenure**, Number of address, Cash back Amount, **Warehouse to Home**, **Day Since Last Order,** Complain, **Order Amount Hike from Last Year**, Satisfaction Score, Number of Devises Registered

Conclusion

The Random Forest method is known for superior performance for classification, and with the ecommerce data it did not disappoint, as the Random Forest Models held the lowest error rate on the test data, compared to other models. Boosting, also known as a strong model, held its own, coming in second to Random Forest for lowest error prediction. Regarding whether parameter tuning could effectively improve performance, Boosting showed a slight improvement with parameter tuning, particularly when missing values were omitted. Random Forest was virtually unchanged. I attribute this to Random Forest's suitability for the non-parametric nature of the ecommerce data set in its default value. Its error rate was very low prior to tuning, as .0389, thus leaving little room for improvement.

Regarding the benefit of omission of missing values vs imputation, an important consideration is the accuracy of the imputation model. As we could not reduce the error rate below 12% for the training data and 15% for the test data, therefore imputation decreased model performance, for all models except for Naïve Bayes with all 34 predictors.

We were able to identify the top nine influential features. For some more traditional models, reducing features to the strong predictors should increase a models' performance. This was true for Naïve Bayes, which saw the most dramatic increase in performance from feature reduction among the models both with omitted and imputed values. This was not the case with any of the other, newer models studied, such as Random Forest, Boosting, Classification Tree and LDA, due to the fact their algorithms are designed to be well suited to large, non-parametric data: here the models' performance was virtually unchanged due to feature reduction. Furthermore, as the proportion of erroneous features increased, from decreasing

features from 34 to 9, the models just noticed performed worse. This indicates Naïve Bayes is less sensitive to data's inaccuracies than the other methods.

Finally, and perhaps most importantly, how could these findings be used by an ecommerce business to reduce customer attrition and help improve the bottom line? The identification of the highly influential features on Churn can be used to guide an ecommerce business in metrics to measure and threshold values to triggers customers likely to Churn. This customer segment can then be contacted in various ways an effort to retain their business, such as with special promotions, coupons, or even be contacted by phone by customer service. These top features can also be incorporated in analysis used to compute broader business metrics, such as forecasting, monthly recurring revenue (MRR), customer lifetime value, and customer acquisition costs.

Future Study

1. Imputation: continue to explore tuning missForest to reduce the error rate.
2. Research causes of possible collinearity between Tenure and Churn.

Appendix

A. Charts & Graphs: Exploratory Data Analysis

Figure 1: Data Dictionary of the Ecommerce data set.

A tibble: 20 × 3

Data <chr>	Variable <chr>	Discerption <chr>
E Co…	CustomerID	Unique customer ID
E Co…	Churn	Churn Flag
E Co…	Tenure	Tenure of customer in organization
E Co…	PreferredLoginDevice	Preferred login device of customer
E Co…	CityTier	City tier
E Co…	WarehouseToHome	Distance in between warehouse to home of customer
E Co…	PreferredPaymentMode	Preferred payment method of customer
E Co…	Gender	Gender of customer
E Co…	HourSpendOnApp	Number of hours spend on mobile application or website
E Co…	NumberOfDeviceRegistered	Total number of deceives is registered on particular customer
E Co…	PreferedOrderCat	Preferred order category of customer in last month
E Co…	SatisfactionScore	Satisfactory score of customer on service
E Co…	MaritalStatus	Marital status of customer
E Co…	NumberOfAddress	Total number of added added on particular customer
E Co…	Complain	Any complaint has been raised in last month
E Co…	OrderAmountHikeFromlastYear	Percentage increases in order from last year
E Co…	CouponUsed	Total number of coupon has been used in last month
E Co…	OrderCount	Total number of orders has been places in last month
E Co…	DaySinceLastOrder	Day Since last order by customer
E Co…	CashbackAmount	Average cashback in last month

20 rows

Figure 2: Histogram of Churn and 18 predictors of Ecommerce data set, after missing values removed. Note the absence of normal distribution of many predictors. Scale will help with this, though only slightly. Random forest will have an advantage over Naïve Bayes in this situation. I've encounter mixed reviews on whether adjustments should be made to create normal distribution. I've decided to opt for leaving the variables skewed. See Figure X, below, that offers a visualization of the numeric skew values for each feature.

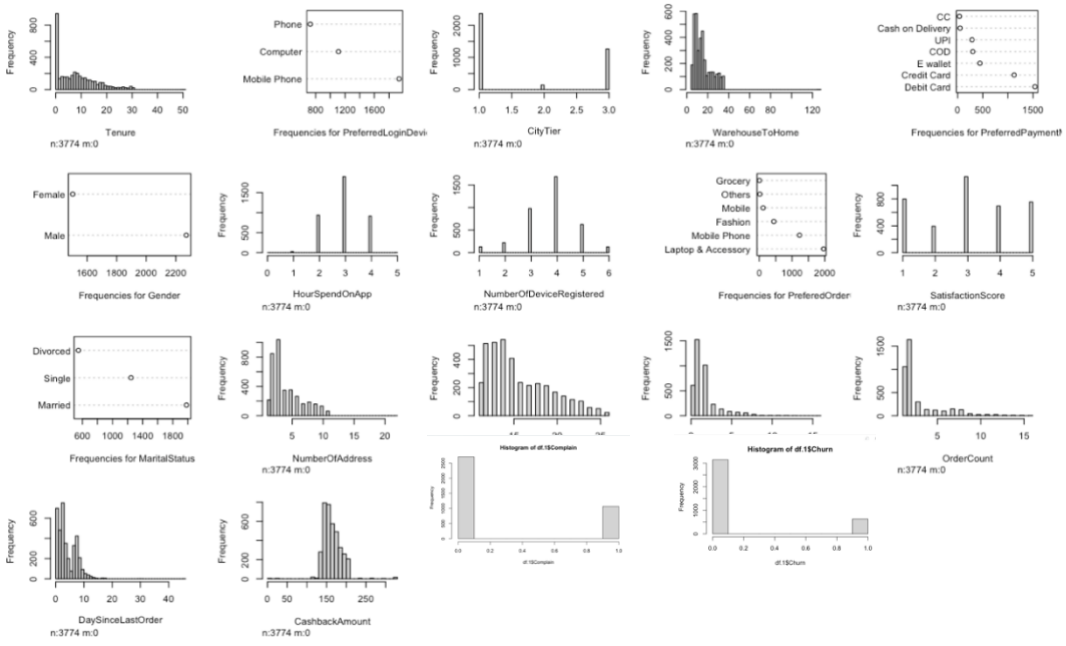

Figure 3: The Seven Missing Values of the ecommerce data, noted by a number in the n-missing column in the chart, below. Thus, we see 7 of the original 20 features had missing values. Note that no character values have missing values, so when they were made into dummy variables, creating 34 features, we still had only 7 features with missing values.

	skim_type <chr>	skim_variable <chr>	n_missing <int>	complete_rate <dbl>
1	character	PreferredLoginDevice	0	1.0000000
2	character	PreferredPaymentMode	0	1.0000000
3	character	Gender	0	1.0000000
4	character	PreferedOrderCat	0	1.0000000
5	character	MaritalStatus	0	1.0000000
6	numeric	CustomerID	0	1.0000000
7	numeric	Churn	0	1.0000000
8	numeric	Tenure	264	0.9531083
9	numeric	CityTier	0	1.0000000
10	numeric	WarehouseToHome	251	0.9554174
11	numeric	HourSpendOnApp	255	0.9547069
12	numeric	NumberOfDeviceRegistered	0	1.0000000
13	numeric	SatisfactionScore	0	1.0000000
14	numeric	NumberOfAddress	0	1.0000000
15	numeric	Complain	0	1.0000000
16	numeric	OrderAmountHikeFromlastYear	265	0.9529307
17	numeric	CouponUsed	256	0.9545293
18	numeric	OrderCount	258	0.9541741
19	numeric	DaySinceLastOrder	307	0.9454707
20	numeric	CashbackAmount	0	1.0000000

Figure 4: Data Pre-Processing: Standardization of variables, after dummy variables have been created, and factors have been assigned (0,1) values. This will help with uneven distributions.

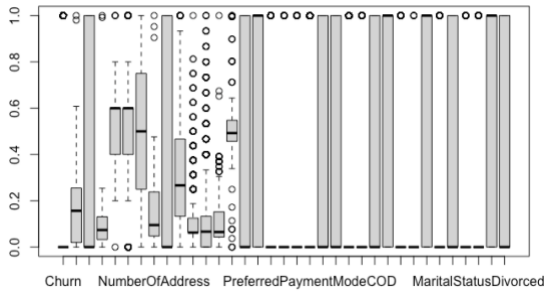

Figure 5: Random Forest Ranking of Influential Predictors of the Ecommerce Data set

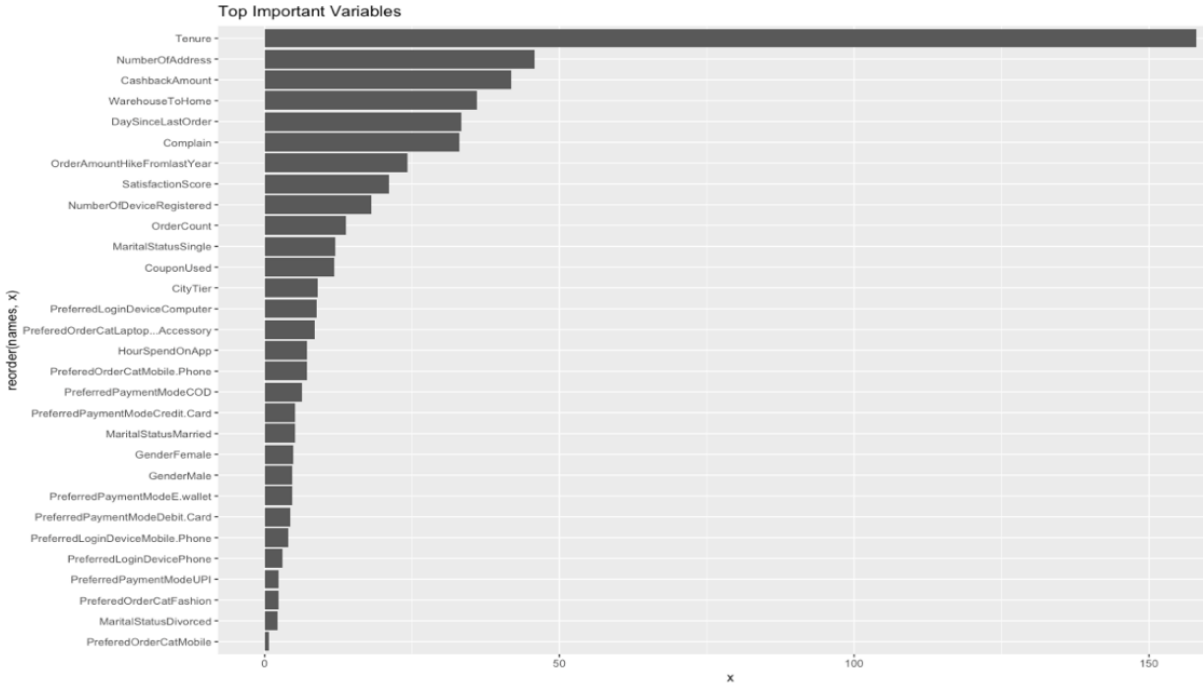

Figure 6: Boosting Ranking of Influential Predictors of the Ecommerce Data set, after parameter tuning and optimization.

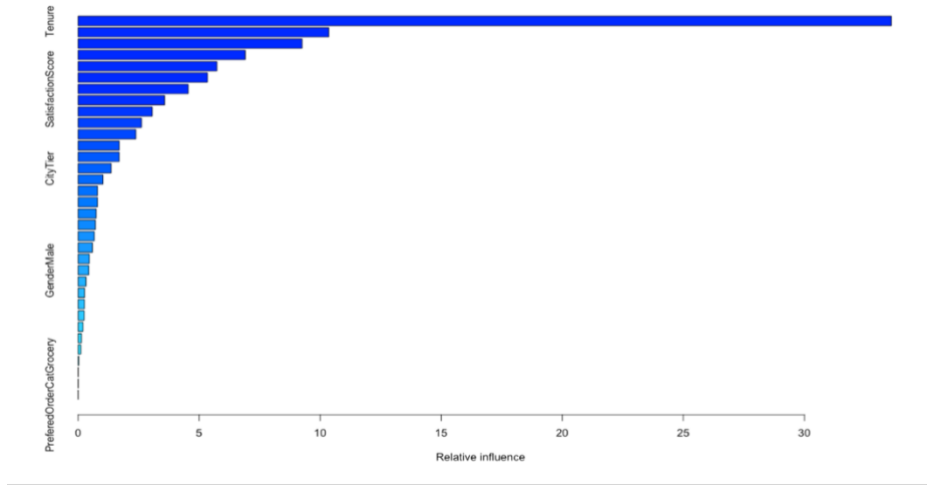

Figure 7: Other Imputation Methods Considered:

MICE (Multivariate Imputation via Chained Equations) was used in the Kaggle Analysis; we will therefore not use it here, although it is well suited for our data. It assumes data is MAR, missing at random. It creates multiple imputations, which helps reduce the bias and increase efficiency, as compared to a single imputation such as calculating the mean based on other values for that feature. It uses logistic regression for categorical missing values; linear regression for continuous ones. For example, say we have variable X1, X2, X3. To calculated missing values for X2, it will be regressed on variables X1 and X3. It will thus produce multiple data sets. MICE has four parameters to select, depending on the nature of the variable: numeric, binomial, factor (greater than two levels), proportional odds model. Other parameters include number of data sets, number of iterations, method. It produces a visual chart to illustrate the concentration of missing data in the data set. You can get a consolidated data set that pools the data sets generated for each missing variable. Mice would work well with the ecommerce data; however, since it was used in the Kaggle analysis, we will use a different analysis for comparison. [11]

Amelia II, named for Amelia Earhart, the famous aviator that went missing over the Pacific in 1937, also uses multiple imputation considering the entire data set, along with bootstrap based EMB algorithm that increases its processing speed. Missing data is random in nature (Missing at Random, MAR.) Variables are multivariant normal distribution (MVN); **so not appropriate for the Ecommerce data set.** Amelia takes m bootstrap samples, applies the EMB algorithm to each, estimating m mean and variance. **As the Ecommerce data is non-parametric with its skewed distribution, this method would be a poor choice for our analysis.** [11]

Hmisc is a statistical software package offering numerous applications such as advanced tables, models, diagnostics as well as imputation. It offers several approaches to missing values: either basic statistics, such as mean, max, median; or a more advanced using bootstrapping, predictive mean matching and additive regression. It assumes linearity in the variables being predicted; Fisher's optimum scoring is used for predicting curatorial variables. A summary shows R^2 values for predicted values. [11]

Multiple Imputation with diagnostics (mi) package: As with some other observations, it builds multiple imputation models; uses predictive mean matching (PMM) which finds other observations with the closest predictive mean to the one with the missing value to create a "match." The observed value is then use to impute the missing value. MI allows for graphical diagnosis, as does MICE; it uses Bayesian regression models, model specification is similar to regression output; automatically detects high collinearity (might make it a contender for the Ecommerce data

set); it adds noise to solve for additive constraints. Parameters that may be tuned include selection of method, such as bootstrap, number of multiple imputations, and number of iterations. [11]

B. References

1. The Ecommerce Data set: https://www.kaggle.com/datasets/ankitverma2010/ecommerce-customer-churn-analysis-and-prediction

1a. Ecommerce Customer Churn Analysis and Prediction. Kaggle. https://www.kaggle.com/datasets/ankitverma2010/ecommerce-customer-churn-analysis-and-prediction

2. How to Calculate Customer Churn Rate (Formula + KPI Example). (2020), Onix. https://onix-systems.com/blog/customer-churn-rate-and-its-impact-on-business-performance

3. Hastie, T., Tibshirani, R., and Friedman, J. (2009), *The Elements of Statistical Learning, Data Mining, Inference, and Prediction, Second Edition.* Springer Science + Business Media, LLC, New York, NY, p. 333. https://hastie.su.domains/Papers/ESLII.pdf

4. Package 'missForest'. Cran.r-project. https://cran.rproject.org/web/packages/missForest/missForest.pdf

5. Random Forests. UC business analytics programming guide. https://uc-r.github.io/random_forests

6. Gradient Boosting Machines. UC business analytics programming guide. http://uc-r.github.io/gbm_regression

7. Choose a model by AIC in a stepwise algorithm. RDocumentation. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/step

7a. Stepwise model selection. RStudio pubs. https://rstudio-pubs-static.s3.amazonaws.com/2899_a9129debf6bd47d2a0501de9c0dc583d.html

8. James, G., Witten, D., Hastie, T., and Tibshirani, R. (2021), An *Introduction to Statistical Learning, Second Edition*, *with Applications in R*. Springer Science + Business Media, LLC, New York, NY, pp. 142, 358. https://hastie.su.domains/ISLR2/ISLRv2_website.pdf

9. Naive Bayes classifiers. Wikipedia. https://en.wikipedia.org/wiki/Naive_Bayes_classifier

10. Naïve Bayes Classifier. UC Business Analytics R Programming Guide. https://uc-r.github.io/naive_bayes

11. Analytics Vidhya, (2020), Tutorial on 5 Powerful R Packages used for imputing missing values**.** https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/

12. Stekhoven, D.J. and Buehlmann, P., (2012), MissForest - nonparametric missing value imputation for mixed-type data. Bioinformatics, 28(1) 2012, 112-118, Doi: 10.1093/bioinformatics/btr597 https://academic.oup.com/bioinformatics/article/28/1/112/219101

C. Code https://github.com/catherman/Predicting-churn-with-machine-learning-

D. Visualization https://public.tableau.com/app/profile/margaret.catherman