The Price is Right…Right?

MGT-6203 – Data Analytics for Business

11/20/2022

Casey Kimes – ckimes3

Carlos Galeano – cgaleano6

Cale Williams – cwilliams461

Lauren Wilson – lwilson85

The Price is Right…Right?

Group 75 – Casey Kimes, Carlos Galeano, Cale Williams, and Lauren Wilson 11/20/22

Abstract

This analysis focused on exploring several different regression models to replicate a “black box” margin optimization model currently utilized to set weekly price margins for thousands of products sold by a single client. The methods used were Multiple Linear Regression, Regression Trees, Random Forests, and Boosting. Each model’s parameters were tuned (when applicable) using the training dataset, compared using the validation dataset, and measured for final performance using the testing dataset. Overall, the random forest and pruned regression tree models outperformed the other models, with testing R-squared values at approximately 35%. This R-squared value was lower than initially expected, and thus further investigation into the dataset was conducted to find potential causes. In addition to the regressors, an evaluation of time series and optimization models are also warranted. Suggestions for further improvement include investigation of other regressors, parameter tuning of the existing models, and a further dimensionality reduction.

Introduction

The client organization outsources the determination of product margin to a third party at the cost of $1M annually. Weekly they transmit data such as customer, sales organization, product information, and ten days of historical transactions to the supplier. Within three days, the supplier then sends the margin recommendations for each product back to the organization. These recommendations are then applied to product cost to determine the price point for the item based on geographic region, product, and a myriad of other factors. The sales staff then use these margins the following week in their sales negotiations. The team is looking to replicate this solution and bring the entire process in-house for the client. While there are numerous advantages of bringing the solution in-house, the most impactful is the reduction in annual spend. Secondary to the cost reduction is also the ability to augment and grow the solution as they deem necessary, and, finally, the last benefit is the timeliness of the recommendations. The organization is looking to have near real-time recommendations as the cost of the items is constantly in flux, and a week is often too long to sell products with upside-down margins, which could result in substantial profit losses.

This analysis aims to model a similar solution as a baseline for the client to begin exploring the previously mentioned benefits. Naturally, a complete duplication of the solution is a heavy lift, but using the data provided and the typical results, the team will be focused on a subset of products to model a target margin recommendation engine via supervised learning. Based on initial observations of the data, it is expected that a regression model will be most effective in predicting the margins based on a combination of the weekly input factors provided to the supplier. Several regression methods will be explored, including Multiple Linear Regression, Regression Trees, Random Forests, and Boosting. Model performance will be measured by comparing the Mean Squared Error (MSE) on a test dataset from each regression model. This report will explore the dataset and any unique aspects of it and discover any usable patterns for prediction using the methods listed above. Finally, the suspected causes of the quantitative results, lessons learned, and potential future improvements will be discussed in the conclusion.

Data Sources

The client provided 2 main datasets in the form of .csv files or .txt files:

- *transactions* – a record of the past week's sales, with 52 columns and 1.9 million rows. Due to the sheer size of this file, a subset of the dataset will be used during the training and testing of the model to speed up development. This dataset includes invoice details, product ID numbers, sales team information, and cost information. It also includes a section of "MarketWise" columns, which are historical margins for each item based on third-party analysis. This dataset is the main input file provided to the model that produces the desired responses.
- *REC_OUTPUT_FILE* – this is the main output file from the black box model that is trying to be replicated. Though there are several output variables, the most valuable is the "Target" column which represents the suggested target margin for each product based on the factors given in the input file.

Table 1 below lists some of the columns suspected to be the most important to the primary regression model from the provided datasets.

Attribute	Description	Data Type
TARGET	Target margin for each product.	numeric
BRANCH	Organization Branch	categorical
PROD_WHS	Product Warehouse	categorical
PRICE	Price the unit was sold at	numeric
SALES_TEAM	Sales Team Identifier	categorical
TRUCOST	True Cost of the product	numeric
BSCST	Base Cost of the product	numeric

Table 1: Dataset Attributes, Descriptions, and Data Types of transactions dataset

Sampling the Data

Through random sampling, the dataset with the independent variable(s), transactions, was reduced to 700 data points (rows). This set was joined on the *REC_OUTPUT_FILE* via the intersecting variable PROD_ID. Many PROD_ID values corresponded to multiple values in the response variable, TARGET. Each of these was included via an inner join. Thus, the resulting dataset included about 168,000 rows.

Figure 1 shows the distribution of the response variable pre- and post-reduction. The reduced set's distribution appeared to represent the entire dataset. This data reduction was then deemed acceptable and removed potential computational roadblocks in future modeling steps that arise with larger datasets.

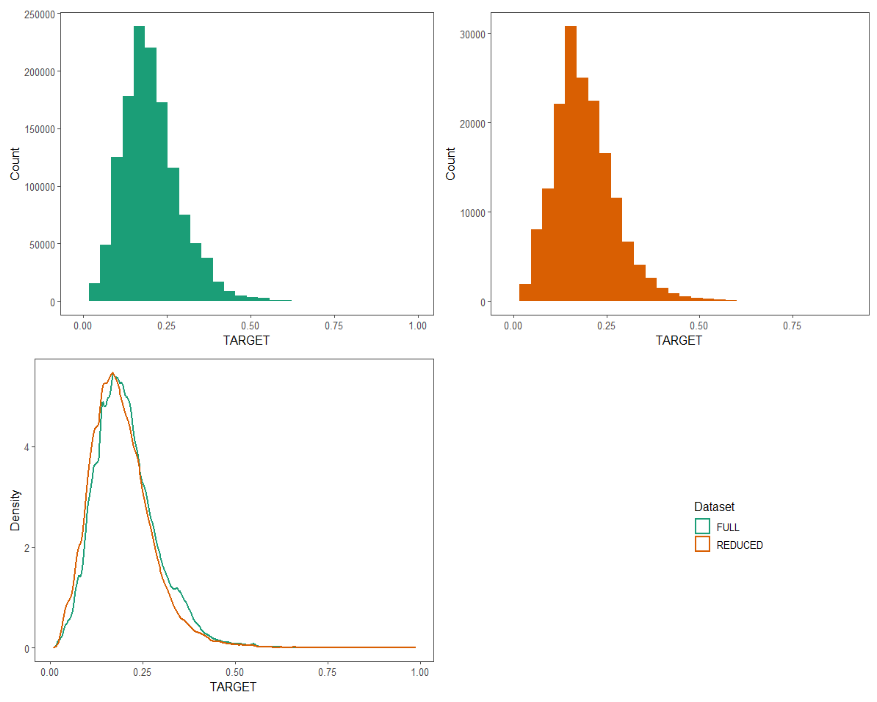


Figure 1:Response variable, TARGET, distribution before and after sampling

Exploratory Data Analysis and Preparation

Variable Distributions

Before eliminating any variables, scatterplots were built to determine if each variable was linearly related to the TARGET margin response. However, there was no clear indication that any single variable could be used as a sole predictor. This is demonstrated by plotting VENREBATES and TRUCST against TARGET in Figure 2 below.

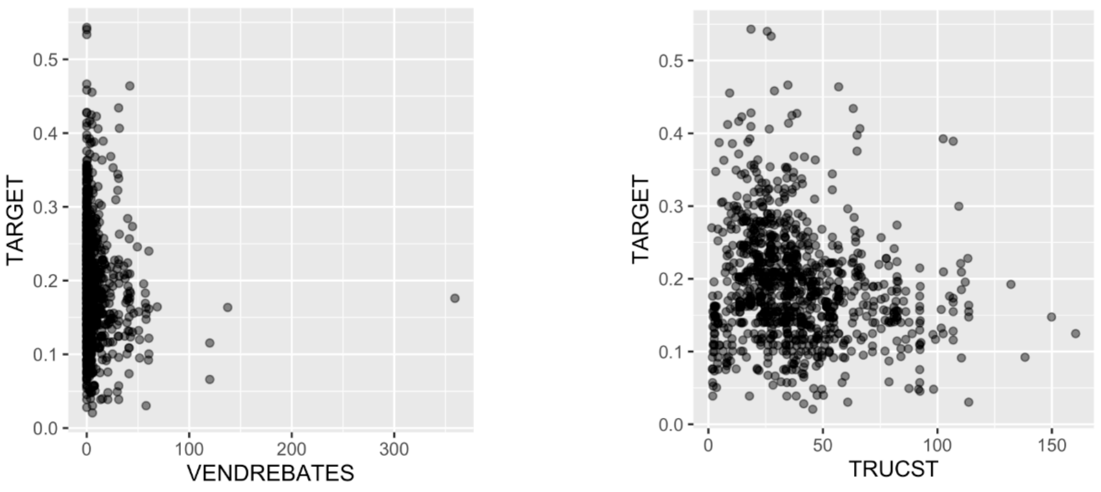


Figure 2: Scatterplots of VENDREBATES and TRUCST vs. Response Variable, TARGET

Boxplots and pair plots were built to understand predictors' response patterns. Figure 3 below shows a boxplot of BRANCH as it relates to TARGET. The current margin recommendations fall at a mean of approximately 19%, but there are instances where the margin is as high as 80%. The right-skewed nature of the plot suggested that further data reduction would be needed to achieve a more normal distribution and, by extension, better predictive models. Furthermore, the pairs plot in Figure 3 shows the initial correlation analysis of predictors. The only correlated pairs seemed to be PRICE and TRUCST, which was expected as the price is directly related to item cost. This is explored further in the next section.

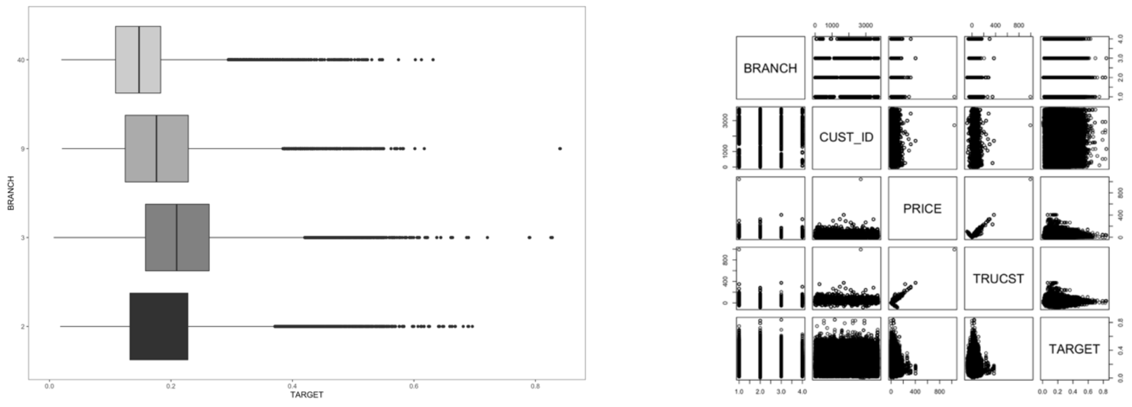


Figure 3: Boxplot of Branches and Target and Pairs Plot of Select Variables

Collinearity & Missing Data

All numeric variables were checked for collinearity before model creation. It is evident in the correlation matrix in that many variables were strongly correlated. Via manual inspection, some categorical variables were also found to be correlated. For example, some BRANCH factor levels corresponded to a single PROD_WHS factor level. The following combinations of variables strongly correlated. The bolded variable was selected in place of the remaining collinear variables.

- PRICE, **TRUCST**, INVCST, BSCST, FINCST, MKTWS_BASE, MKTWS_BASE_PLUS, MKTWS_BOTTOM_OF_TARGET, MKTWS_TARGET, MKTWS_TOP_OF_TARGET, MKTWS_LIST with each other
- **TOTALBILLBACKS** with EXTBILLBACKS
- SALES_TEAM, INVTYPE, PROD_WHS, **BRANCH** with each other
- **TRANSTYPE** with ORDER_SOURCE

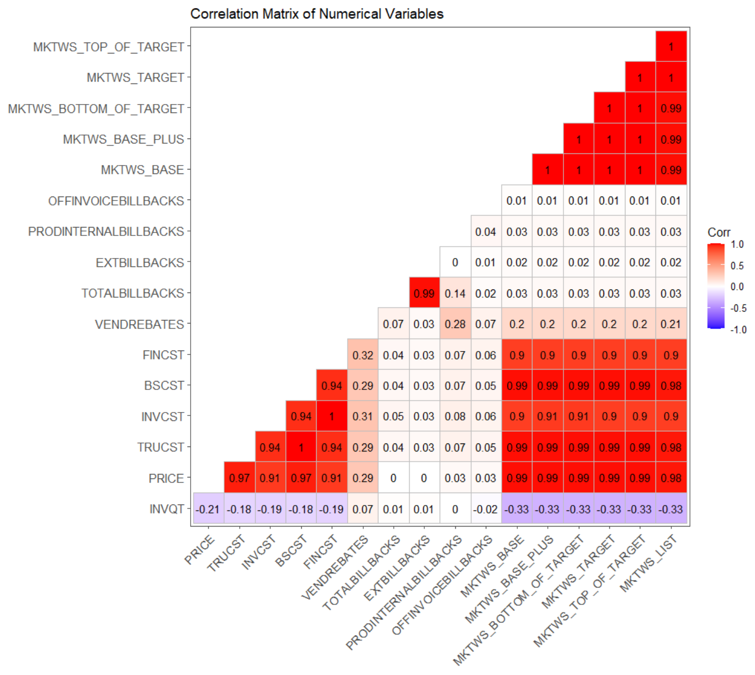


Figure 4: Correlation Matrix of Numeric Independent Variables in transaction Dataset

After the removal of collinear variables, missing data was investigated. TRANSTYPE was the only variable not fully populated, missing 14% of its data. These missing rows were inconsistent with another predictor's values, implying no cause or bias for the missing data. Thus, it was deemed acceptable to remove rows in which TRANSTYPE was missing.

Following the exploratory data analysis phase and the data preparation steps described, the resulting dataset included 145,055 data points with the variables in below. This was the input dataset for variable selection.

Attribute	Description	Data Type
BRANCH	Branch	Factor w/ 4 levels
INVDTE	Invoice Date	Date range from 09/05/2020 to 09/21/2020
TRUCST	True Cost / Rep Floor	Numeric ranging from -49.6 to 372.8
IDC	Invoice Debit Credit	Factor w/ 2 levels
PriceAgreementMethod	Agreement Method	Factor w/ 7 levels
VENDREBATES	Vendor Rebates	Numeric ranging from 0 to 114
TOTALBILLBACKS	Total Billbacks	Numeric ranging from 0.0 to 460.1
PRODINTERNALBILLBACKS	Internal Billbacks	Numeric ranging from 0.0 to 16.7
OFFINVOICEBILLBACKS	Off Invoice Billbacks	Numeric ranging from -0.5 to 69.5
TRANSTYPE	Transaction Type	Factor w/ 5 levels
TARGET	Margin (Response)	Numeric ranging from 0.01 to 0.91

Table 2: Model Input Dataset Summary

Methodology

Splitting the Data

The data were divided into training, validation, and test sets to build the regression models. This was done by randomly sampling 70% of the dataset in the training set and the remaining 30% to be split evenly between the validation and test sets. This distribution is common as it is beneficial to have a larger training set than validation and test sets to reduce overfitting.

Variable Selection

Three variable selection regression methods were employed to improve and simplify models by reducing the number of predictors: Stepwise, Lasso, and Elastic Net.

- *Stepwise Regression-* A method that combines backward elimination and forward selection to perform a step-by-step iterative construction of a regression model for variable selection. Stepwise regression starts with no predictors, then sequentially adds the most contributive predictors (like forward selection). After adding each new variable, it removes any variables that no longer improve the model fit (like backward selection). In R, the algorithm removes predictors by analyzing the Akaike Information Criterion (AIC) of each model built. Once the AIC stops changing, the algorithm stops.
- *Lasso Regression*- An optimization technique that adds a restraint to the standard regression equation. The objective is to find the model that minimizes the sum of squared errors. This approach adds the restriction that the sum of the coefficients cannot be too large. It regularizes the data, penalizing high-value regression coefficients to avoid overfitting. In R, the `cv.glmnet()` function uses a tuning parameter, lambda, to perform the penalizations. As lambda increases, shrinkage occurs to eliminate variables.
- Elastic Net Regression- A similar approach to Lasso but rather than just constraining the absolute value of the magnitude of the coefficients, it also constrains their squares.
- An optimization technique that constrains the absolute value of the magnitude of the coefficients and their squares. In R, this is done with the caret package and `cv.glmnet()`. An extra tuning parameter, alpha, determines how much to penalize the model.

Regression Models

The response variable, TARGET, is a continuous value, meaning that regression models are the best models to consider for predicting TARGET accurately. Several potential models could be used; however, based on the characteristics observed during the Exploratory Data Analysis phase, linear regression, regression trees with bagging, random forest, and boosting were selected for this analysis.

- *Multiple Linear Regression* - A linear approach to evaluating relationships between predicting variables and the response. For multiple linear regression situations, where there is more than one predicting variable, a hyperplane is drawn to fit the data and used to predict future responses.
- *Regression Trees* – Regression trees work by iterating through points where the data could potentially be split and calculating the mean square error at each location for each variable. Then, the minimum MSE is chosen for that node, and the process is repeated. The goal is to minimize the overall MSE of the model while ensuring the model does not become overfit. Regression trees are beneficial when there is a nonlinear relationship between the variables.
- *Regression Trees with Bagging* – Bagging is a procedure used for reducing the variance in a statistical learning method such as regression trees. Bagging constructs regression trees using numerous different

bootstrapped (re-sampled with replacement from training dataset on each iteration) and averages the results, which reduces the variance. Bagged trees are a special case of random forest models.

- *Random Forest* – An ensemble method built upon decision trees; Like bagging, random forest models create numerous bootstrapped trees with different decision criteria to consider as many results as possible. The results of the collection of trees are then combined to generate the estimate of the response. Additionally, random forests also select subsets of features used in each data sample. Random Forests are known to be more accurate than decision trees due to the amount of variability represented in the model. However, unlike decision trees, they are difficult to explain as they do not produce a single visible output.
- *Boosting* – boosting is like random forests in that it involves creating many trees; however, instead of the trees being created independently in parallel, they are built sequentially. This means that each tree is grown with knowledge from the previous trees grown, with the goal of correcting the errors from the prior tree.

Analysis & Results

Variable Selection

Stepwise Regression

The stepwise regression model was built using the stepAIC function in R, and the results are summarized in Figure 5.

Figure 5 shows that all the variables have an AIC of approximately -235,000, indicating that the information loss from removing any variables would be minimal. Moreover, the model did not eliminate any variables, and as such, this technique was not used to make variable selection recommendations.

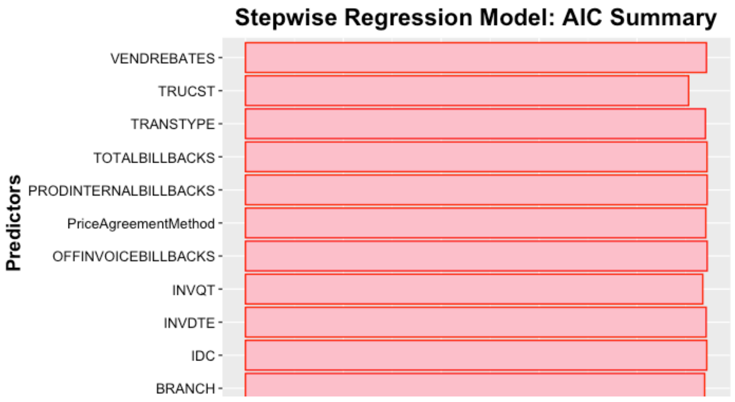


Figure 5: Plot of predictors against their AIC values

Lasso and Elastic Net

The lasso analysis started by building a regression model using the `cv.glmnet()` function in R with MSE as the evaluation measure and eight folds for cross-validation. Figure 6 shows the log(lambda) versus the MSE of the Lasso model. The top values represent the number of predictors. The leftmost dotted line represents the lambda min at which the MSE is minimized. The second dotted line represents lambda 1se, which is the lambda that gives the most regularized model such that the cross-validated error is within one standard error of the minimum. The narrow range of MSE values indicated that reducing the variables would not significantly affect the model's performance. However, since the difference in the MSE for lambda min and lambda 1se is close to zero, there was an indication that a simpler model with only nine predictors could be used without sacrificing performance.

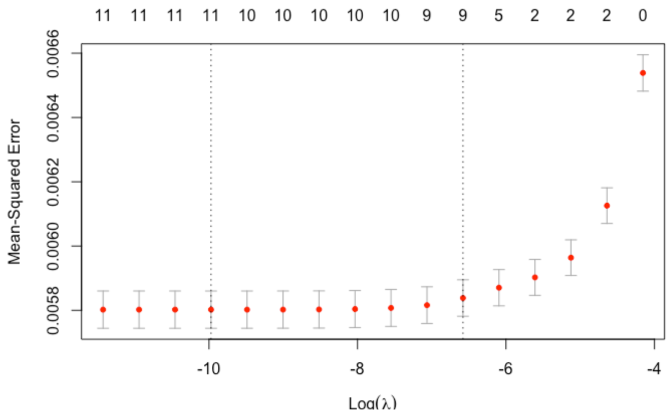


Figure 6: Lasso plot of log(lambda) versus MSE per number of predictors

The first elastic net model was also built using `cv.glmnet()` using the same number of folds and employing MSE as a measure.. However, it was iterated over 20 alpha values ranging from 0.05 to 0.095. Figure X below shows similar results to Lasso and further reinforces the idea of potential model simplification by removing two predictors.

A second elastic net model was then built using the caret package, which invokes `glmnet()` but allows for further tuning of alpha and lambda using repeated cross-validation. In this case, the model tested for 25 different alpha and lambda values combinations. Furthermore, each of the three models was rebuilt using validation data. Performance for all models is summarized below.

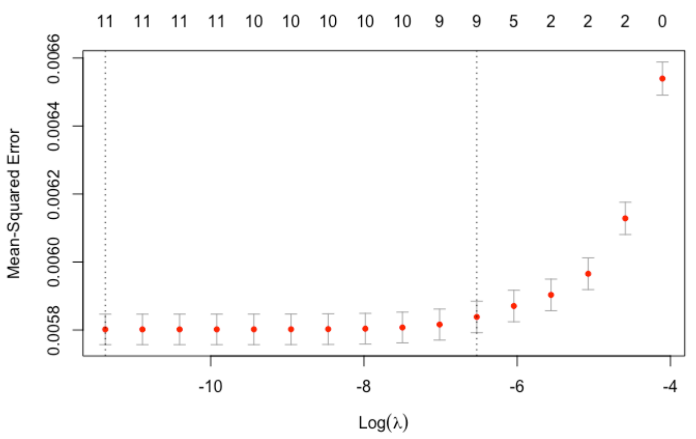


Figure 7: Elastic Net plot of log(lambda) versus MSE per number of predictors

Model	Lambda Training (Optimal)	Lambda Validation (Optimal)	Alpha Training (Optimal)	Alpha Validation (Optimal)	Training MSE	Validation MSE	Training R-squared	Validation R-squared
Lasso	4.67E-05	1.11E-05	1	1	5.802E-03	5.843E-03	11.34%	11.13%
Elastic Net	3.22E-05	2.01E-05	0.5	0.9	**5.802E-03**	5.843E-03	**11.34%**	11.27%
Elastic Net Tuned	1.31E-03	1.31E-03	0.085851	0.085851	5.803E-03	5.803E-03	11.32%	11.32%

Table 3: Results of Variable Selection models with Mean Squared Error and R-Squared Values

Table 4 shows little variation among MSE and R-squared values, implying variable selection from any models could be recommended for the next steps without sacrificing performance. Lastly, Table 4 below shows penalization values for each model.

Model	Lasso	Elastic Net	Elastic Net Tuned
Intercept	-1.34E+01	-1.89E+01	1.89E-01
BRANCH	1.41E-05		
INVDTE	7.31E-04	2.76E-03	3.58E-03
INVQT	-1.64E-03	-2.08E-02	-2.17E-02
TRUCST	-8.36E-04	-2.16E-02	-2.29E-02
IDC	2.18E-02	2.81E-03	4.03E-03
PriceAgreementMethod	2.21E-03	2.76E-03	4.73E-03
VENDREBATES	4.44E-04	2.22E-03	3.91E-03
TOTALBILLBACKS	-2.41E-04	-1.77E-03	-2.58E-03
PRODINTERNALBILLBACKS	-1.34E-03	-1.02E-03	-2.00E-03
OFFINVOICEBILLBACKS	-8.22E-04		-6.47E-04
TRANSTYPE	-4.29E-03	-2.44E-03	-4.58E-03

Table 4: Variable Selection Model Penalty Results

Elastic Net completely shrinks the BRANCH and OFFINVOICEBILLBACKS to zero, implying they can be eliminated from future models. In both Lasso and Elastic Net Tuned, the BRANCH and OFFINVOICEBILLBACKS coefficients are among the smallest, with BRANCH being eliminated completely from Elastic Net Tuned. Considering this information and due to the Elastic Net model showing a slightly lower MSE of 5.802E-03 and a higher R-squared of 11.34%, these two predictors could be potentially excluded from different summaries.

Regression Summary

Due to the continuous nature of the response variable, the model's accuracy is being measured via the Mean Squared Error (MSE) and R-squared values. The training and validation MSEs and the R-squared values for each model are shown in Table 5 below. The goal is to find the model that minimizes MSE and maximizes R-squared. As shown in the table, based on the validation MSE, the model with the lowest error and highest R-squared value was the random forest model, followed by the pruned regression tree at 35.14% and 34.97%, respectively. There is a big gap in accuracy between these two models and the remaining ones created, with the next best models being the simple and bagged trees with an R-squared value of 18.49% (for both models). The multiple linear regression model failed to produce comparable results; however, an extensive investigation of the model assumptions and variations was completed and detailed in the next section.

Model	Training MSE	Validation MSE	R-Squared
Random Forest	0.004208	0.004267	35.14%
Pruned Regression Tree	0.004213	0.004278	34.97%
Simple Regression Tree	0.005306	0.005362	18.49%
Bagged Tree	0.005304	0.005362	18.49%
Boosting	0.005564	0.005605	14.79%
Multiple Linear Regression	Invalid Results	Invalid Results	Invalid Results

Table 5: Results of Regression Models with Mean Squared Error and R-Squared Values

Multiple Linear Regression

Multiple Linear Regression was run iteratively, starting with a base model and solving issues one a time with a new model at each step.

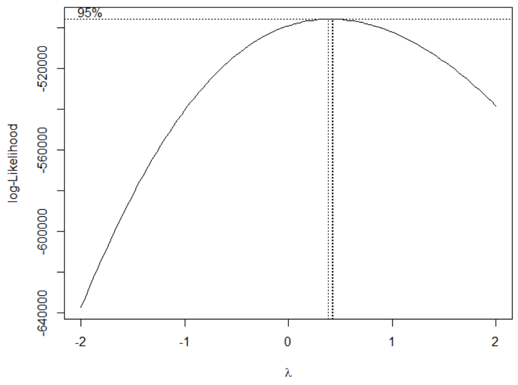
The initial model, Model 1, was run regressing TARGET on all variables post-variable selection. This list of variables includes BRANCH, INVDTE, INVQT, TRUCST, IDC, PriceAgreementMethod, VENDREBATES, TOTALBILLBACKS, PRODINTERNALBILLBACKS, and TRANSTYPE. The variance inflation factor for PriceAgreementMethod and TRANSTYPE was high, implying multicollinearity between the two variables. Therefore, PriceAgreementMethod was removed in subsequent models to be run. Additionally, Model 1 violated constant variance and normality assumptions. As the fitted values become larger, so do the standardized residuals.

Model 2 was run with identical parameters as Model 1, without the collinear variable PriceAgreementMethod. As expected, there was no multicollinearity found via VIF. However, Model 2 violated the constant variance and normality assumptions still.

To mitigate this, a Box-Cox transformation was performed to build Model 3. The optimal transformation is found as follows:

$$\lambda = 0.5 \Rightarrow \sqrt{TARGET}$$

As shown in Figure 9, normality was improved between Model 2 and Model 3. However, the constant variance assumption was still violated.



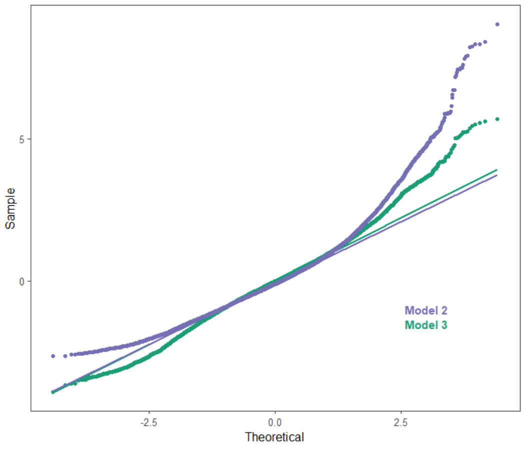
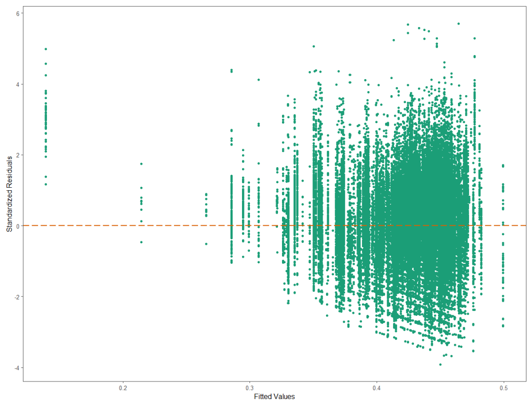


Figure 9: Box-Cox Optimal Lambda (left) and QQ Plot Pre/Post-Transformation (right)

For the final model, Model 4, an interaction term was created by multiplying INVQT by TRUCST. It was thought that perhaps profit margin (TARGET) could be dependent on not order quantities (INVQT) or costs (TRUCST) but the interaction between the two predicting variables. For clarity, Model 4's code line is below:

$$\sqrt{TARGET} \sim INVQT \times TRUCST + BRANCH + INVDTE + IDC + VENDREBATES + TOTALBILLBACKS + PRODINTERNALBILLBACKS + TRANSTYPE$$

This produced the highest coefficient of determination of the models at $R^2 = 0.14$. However, as shown in Figure 10, the constant variance assumption was still violated, and the results were deemed unreliable.



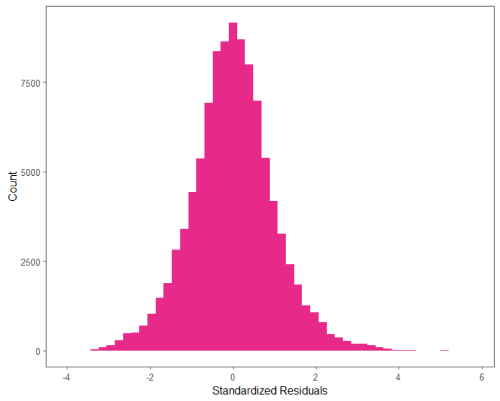


Figure 10: Model 4 Constant Variance (left) and Normality (right) Assumption Checks

Regression Trees and Bagging

A few variations of the regression tree models were explored during this analysis to find which one had the best fit. First, a simple regression tree was made with all the standard parameters. The output of this simple tree is shown in Figure 11 below. This model was created to understand the outcome if simplicity was used as the priority in the model. However, it is known that more complex, more accurate models can be obtained by manipulating the parameters. Thus, the pruned model was created by tuning the complexity parameter (cp) to the value that minimized the model's error and pruning the full tree using this parameter. The difference is shown in

Figure 11 below. Though illegible, the sheer size difference in this tree demonstrates the change in detailed decision-making in the pruned tree versus the simple tree. This led the pruned tree to have a lower MSE value and, thus, a significantly higher R-squared value than the simple tree. In fact, the pruned tree had the second-highest R-squared value of all the models, at 34.97%, slightly lower than the random forest model.

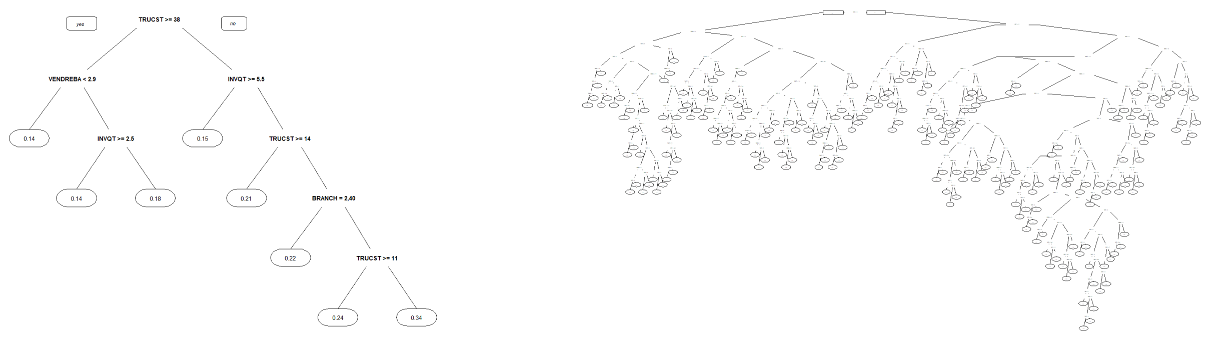


Figure 11: Simple Regression Tree (left) and Pruned Regression Tree (right)

The final attempted improvement on the decision tree models was using bagging, a technique utilizing bootstrapping to reduce the variance in the model. However, the MSE value was higher for the bagged tree models than for the pruned tree models, and thus the pruned trees were the best version of this model found.

Random Forest

Building upon the same concept as bagging, random forests also use bootstrapping to reduce variability and consider a collection of several unique trees when deciding the value of the estimated response. Because of this, it has a lower risk of overfitting and is also more robust to outliers, which are both essential features for this dataset. The main challenge of the random forest model is tuning the parameters to achieve the best possible model. In this study, two parameters were primed – `ntree`, the number of trees to grow in the forest, and `nodesize`, the minimum size of the terminal nodes. The default values for regression models are 500 and 5, respectively. A range was set for each value and run through a set of loops to test different combinations of the factors and find the values that returned the lowest training MSE. For this dataset, that was `ntree = 600` and `nodesize = 5`. When a model was created with these parameters on the training data, the training MSE was the lowest of all the models tested, at 0.004208. When running on the validation dataset, the error jumped slightly to 0.004267, which is expected. Overall, the random forest model had the highest R-squared value of all the models at 35.14%. Interestingly, the random forest did substantially better than the bagged regression trees models, which had an R-squared value of only 18.49%, as they are similar algorithmically.

One drawback of random forests is that they are hard to interpret as they do not produce one single output model. However, the model does give a glimpse into the importance of the variables in determining the response, as shown in Figure 12 below. The plot shows that TRUCST and VENDREBATES again proved to be the most crucial deciding factors, consistent with the results from the Decision Tree model.

The boosting model process is similar to the random forest in that it required tuning a few key parameters to get the best performance from the models. Using cross-validation via the `gbm.perf()` function, the optimal number of trees was found to be 500. Despite a variety of tuning methods tried on several other parameters, the highest R-squared value that could be achieved for this dataset was 14.79%, which is the lowest of all the models. Oddly, this result is so much different than the random forest, as it is typical for the sequential building of trees to benefit the model's accuracy rather than hinder it.

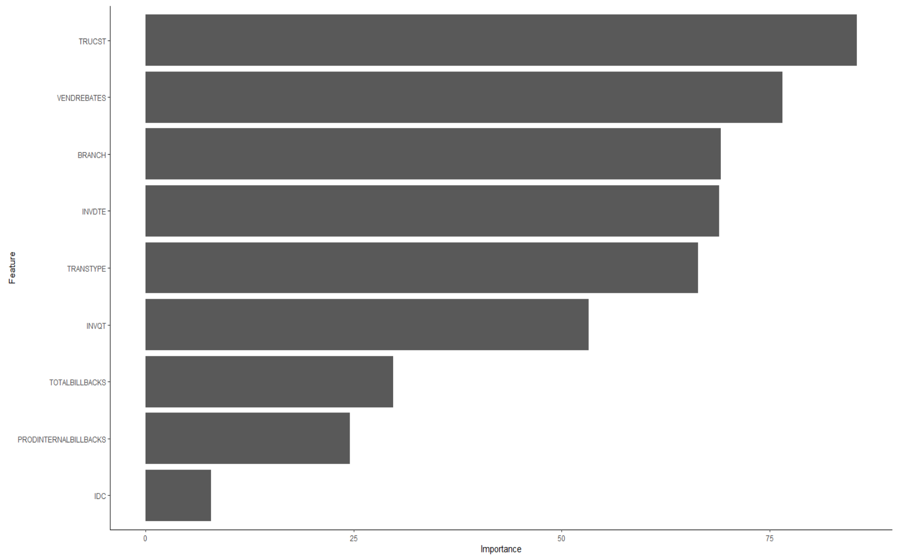


Figure 12: Predicting Variable Importance Based on Random Forest Model

Based on these results, the random forest and pruned regression tree models were selected and run on the dataset again, this time being trained on a combination of the training and validation sets and evaluated using the unseen testing data. With this, the R-squared values of the final models were 35.52% for the random forest and 35.39% for the pruned tree. The performance of regression modeling overall was lower for this dataset than initially expected. Additionally, there was unexplained variation between model performances, leading to the hypothesis that an underlying pattern in the dataset was not being captured in the regression models explored thus far. This dataset captured millions of data points from a single week of transactions. However, it makes intuitive sense that a price of an item likely changes on some relatively consistent schedule and may change differently depending on the time of year and other time-related factors. Thus, secondary research using time series analysis was also explored to see if it would be a recommended next step to improve the analysis results.

Time Series Modeling Exploration

With the less-than-stellar performance of the regression models, it was decided that further inquiry into other models was required. The team returned to the client and acquired a full six months of transaction data on a handful of products to investigate the results of a time series model for forecasting the margin for one period into the future. The data was aggregated daily using the mean actual margin from the historical transactions and converted into a time series object. That object was then passed to the `auto.arima()` function to generate the model. By using the `auto.arima()` function, the P, D, and Q parameters were automatically selected based on the data passed to the function. The final model was then passed to the

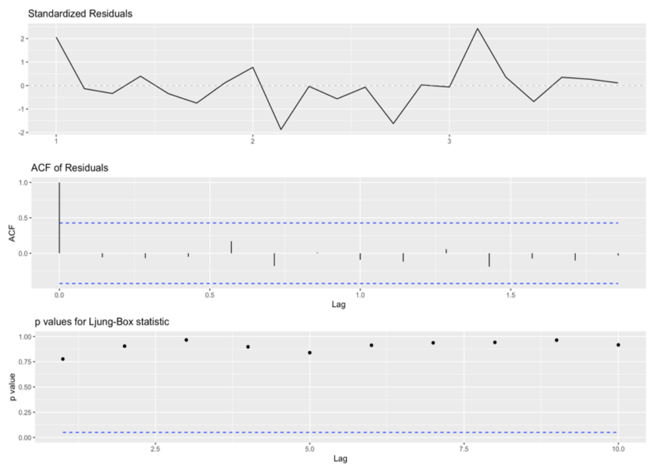


Figure 13: Time Series diagnostic plot

Box.test() function utilizing type = 'Ljung-Box' and lag = 7 arguments to test if the residuals are independent and not autocorrelated. The results provided a p-value of .9372, indicating that we can fail to reject H_0 and conclude that the values are indeed independent – a necessary prerequisite for a time series model.

After the creation and testing of the model, a forecast with a time horizon of one week and a prediction interval of .75 was generated with the following results in Table 6. The target values from the REC_OUTPUT_FILE.txt for the same PROD_ID without additional categorical factors are also included for comparison.

Period	Forecast	TARGET	Forecast High .75	Top_of_Target	Forecast Low .75	Bottom_of_Target
1	.1042518	.104255	.126746	.118307	.08139425	.089747

Table 6: Time Series initial results

The results were promising, with only a -.003% error from the TARGET; however, the high and low values within the prediction interval had wider error margins – a property that could be tuned with the level argument in the forecast function. With the encouraging results, a second PROD_ID was processed by the model, and the results were less than favorable, with error margins of up to 40%. With time to complete the project depleted, further investigation into using a forecasting model was halted.

Conclusion

Initially, the team ran into multiple difficulties with the dataset, but nothing thwarted their resolve. Exploratory Data Analysis and Variable Selection processes provided the necessary predictors eventually used within the final models. By removing null values and dropping collinear variables, the models were simplified enough to be trained, validated, and tested successfully.

Many models were evaluated, and the results were mixed. A random forest yielded the best R-squared value of 35.14%, with a pruned tree in a close second at 34.97%. Interestingly enough, because of the violations of data assumptions, the multiple linear regression results were inconclusive even after numerous attempts to transform the model. Ultimately, the team ran the random forest and pruned tree models on new unseen test data, and, once again, the random forest edged out the pruned tree with R-squared values of 35.52% and 35.39%, respectively. Some additional analysis was conducted using forecasting, but with time running out and the sheer mass of a dataset with six months of data, further progress was not possible.

While the results could have been better, the team gained valuable experience dealing with large and highly dimensional datasets. Most of the team's time was spent understanding, cleaning, and tidying the data. Proper variable selection also became paramount in training the models, which was not possible initially. With the cleaned data and appropriate variables selected, the models were successfully trained and analyzed in the final days of the project.

In terms of future enhancements to the models, additional parameter tuning, further reduction of predictors, and looking at different regressors could improve the solution's performance. In addition, constraint-based optimization should also be explored. The ultimate solution will most likely be a combination of models that lead to the replacement of the "black box" solution the client currently utilizes at a high cost.

References

[1] Kumar, Arun. “Price Prediction Using Machine Learning Regression.” *Towards Science*, 20 Mar. 2020, https://towardsdatascience.com/mercari-price-suggestion-97ff15840dbd.