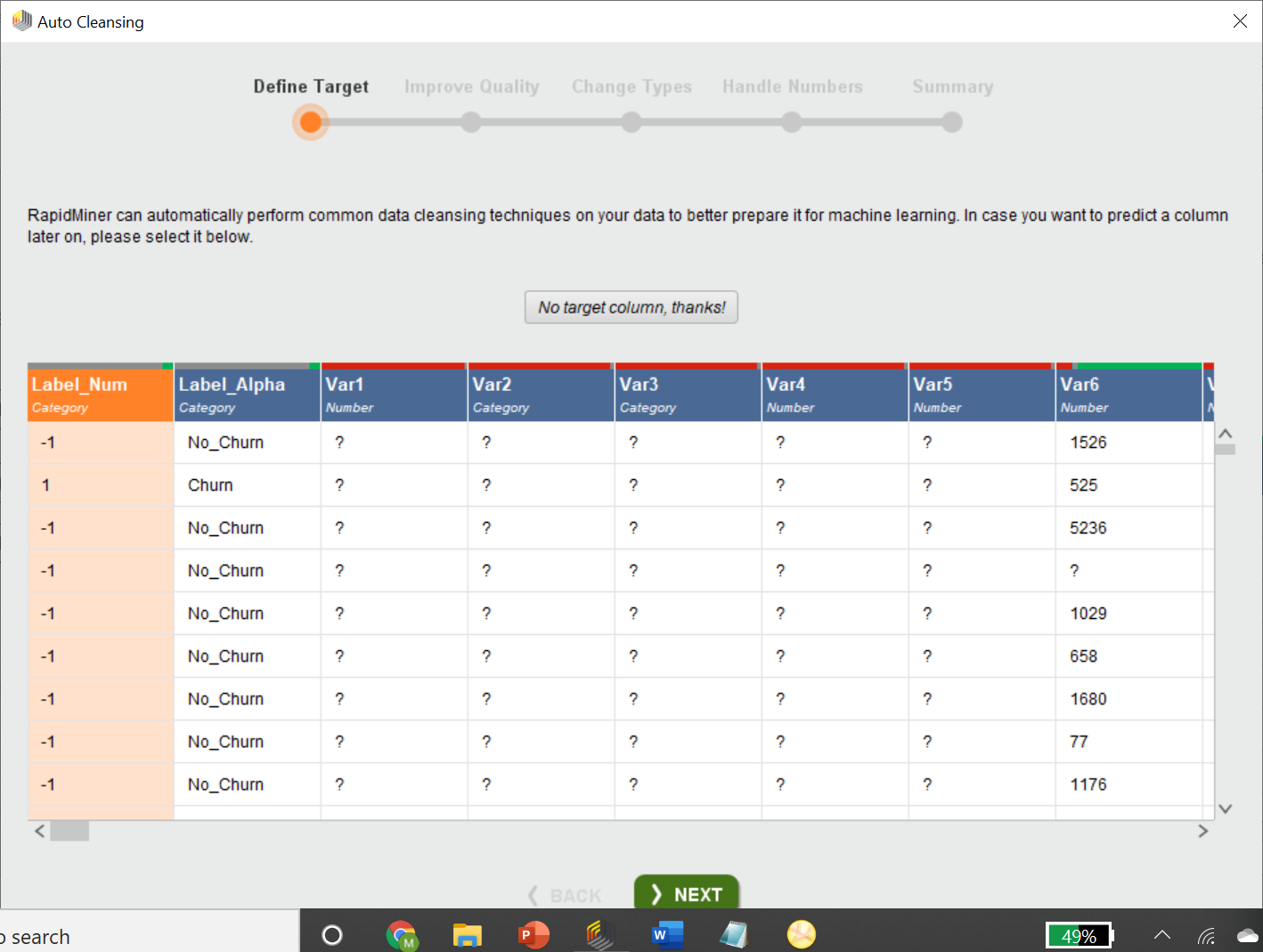
**MDS 560 Week 4 Hands-On Accelerator**

Your deliverables and hands-on activities for this week are:

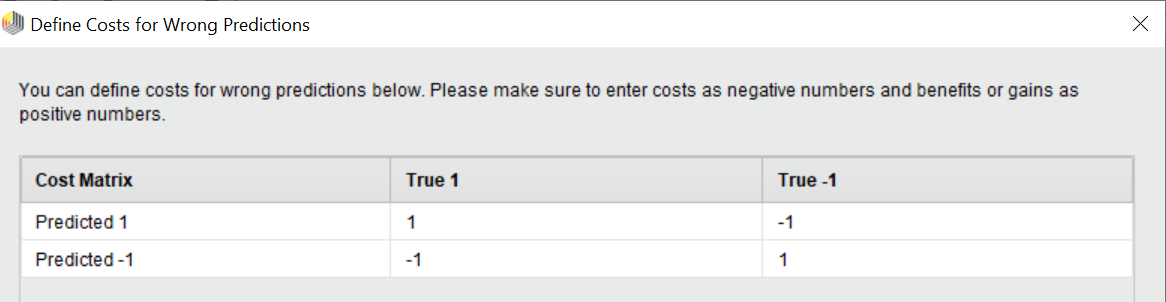
1. Load dataset 3b into Rapidminer. Cleanse the dataset using Turboprep, and then run Automodel on the cleansed dataset. Compare the performance of all model types available in Automodel. What appears to be the best performing model? Explain.

Results: I ran the auto cleanser on dataset 3B. I selected the Label\_Num as the target variable and removed the suggested columns with many missing values and high stability. I changed all categories to numeric since we are running the dataset through auto-modeler and most algorithms require numeric inputs only. I didn’t not apply any transformations to the dataset. I also removed correlated columns.

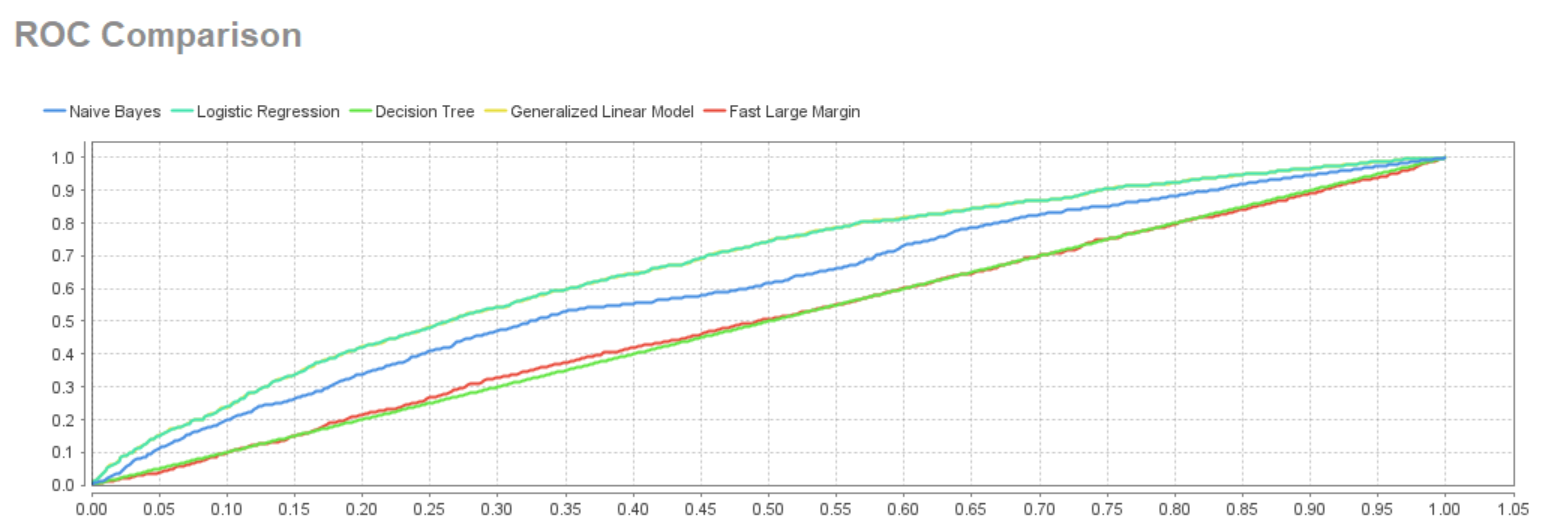


After cleansing the data, I ran the new dataset through auto modeler. I only selected the green variables auto modeler suggested to use resulting in 41 predictor variables included in the model.

When reviewing the target variable, I noticed that the classes are way imbalanced. There are more people who do not churn then churn which is a good thing for business, but not as good for our machine learning algorithms. I reviewed the costs for making the wrong predictions. It was a bit confusing since the true prediction value is 1 and the false prediction value is -1 which are also the values of the cost definitions. However, I think the default assigned values are fair. Incorrectly predicting customer churn means that we either a.) loose a customer to churn that we didn’t the chance to try to save because the customer was not predicted to churn or b.) spend resources attempting to save a customer who was expected to churn but never was going to (how do you prove this was the case though?). Alternatively, correctly predicted customers allow us to spend our resources for customers who truly would of churned and avoid misallocating resources to happy customers.



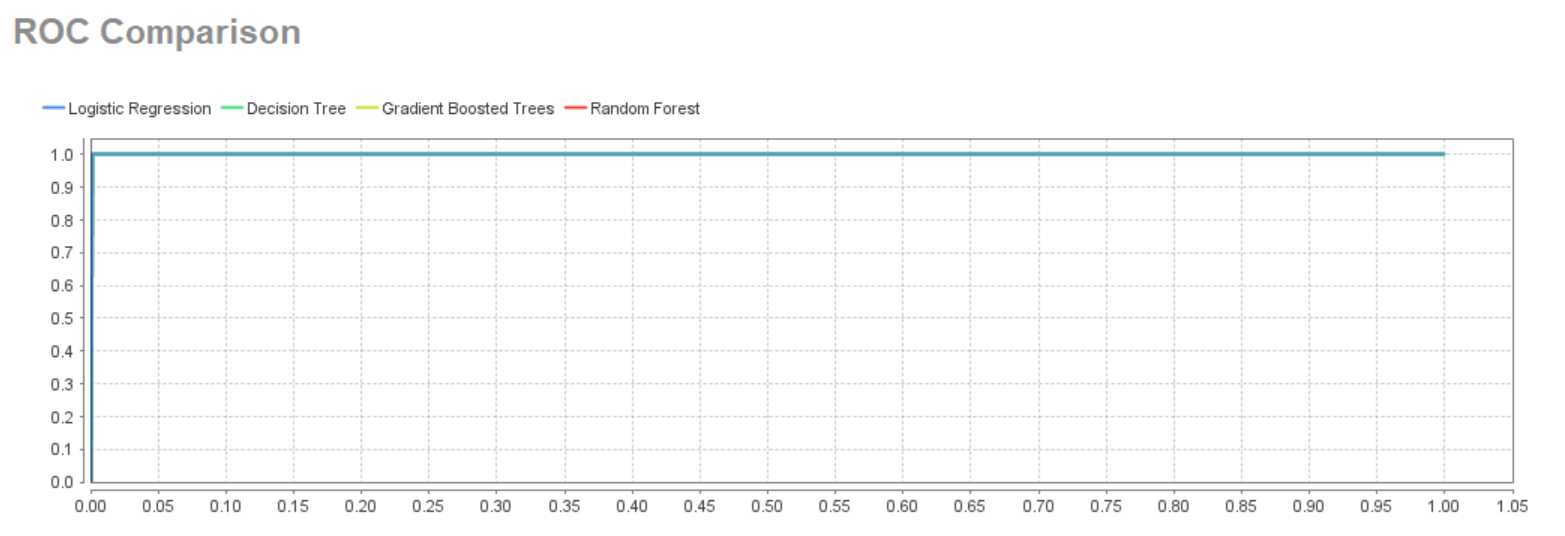
None of the models performed very well. The Generalized Linear Model and Fast Large Margin model performed about as well as guessing would have. It looks like overall, decision tree was able to provide the best overall performance by having the maximum AUC.



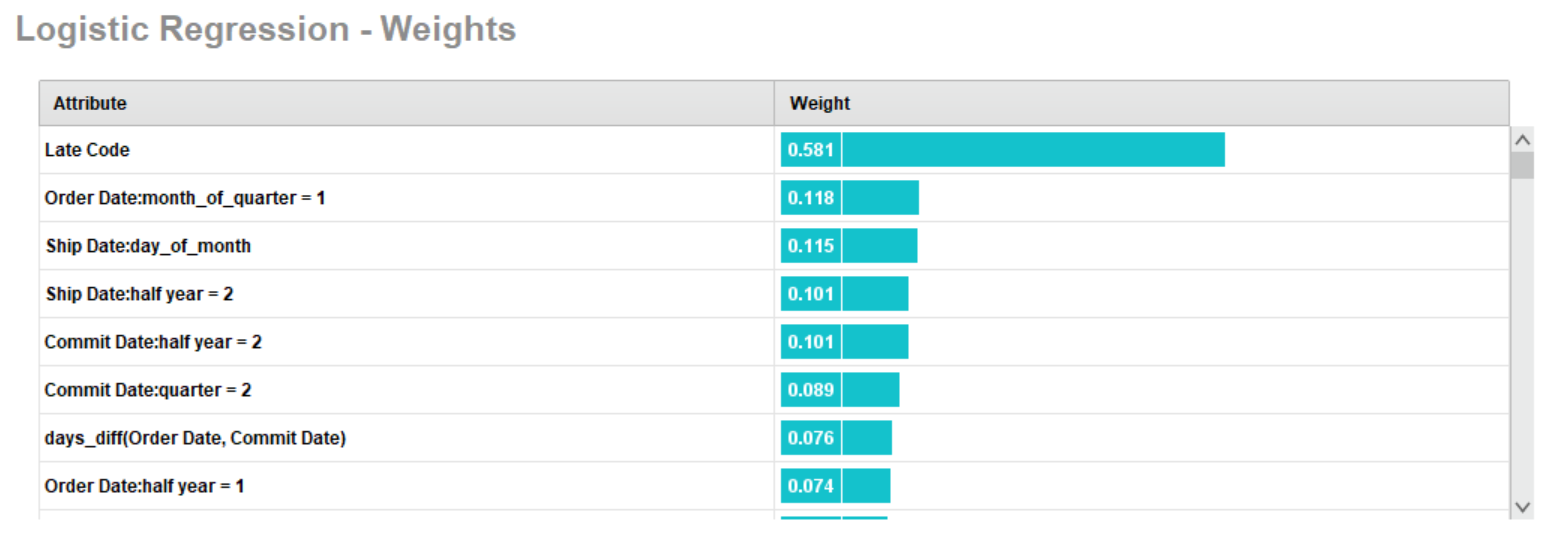
It looks like each model was not able to identify churn cases at all. Each model has a low sensitivity. We could probably get better results if we balanced the classes before training. It looks like the Decision Tree was just better at predicting negative classes of churn which isn’t saying much since that is most of the observations. For this reason, I would say Decision Tree is the best, but I would be very unsatisfied with these results overall and wouldn’t use them to make actionable decisions.

2. Load dataset 4 into Rapidminer and perform predictive modeling using automodel for gaining insight to “Late Flag” as the label for classification. What appear to be the most important predictors of late delivery? What are the implications for action? Examine ROC curves of different models to compare the model performance of gradient boosting, random forest, decision trees and logistic regression, Explain your results.

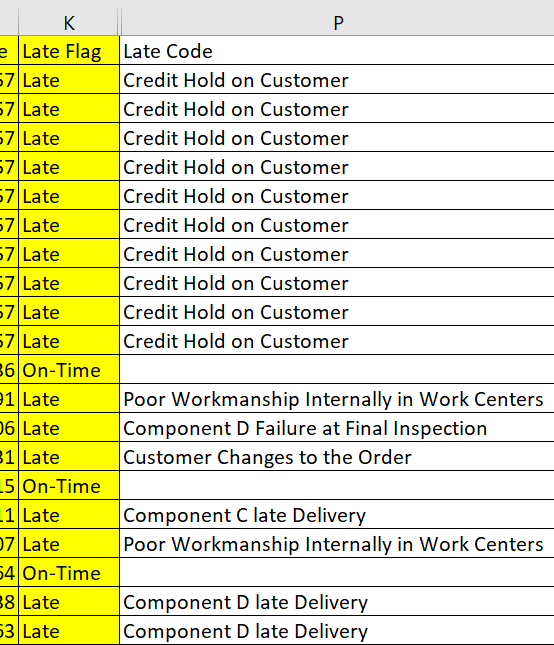
Results: The ROC Curve for these four models are below. Each have an AUC 1 which is perfect performance of classifying late deliveries. This leads me to believe that there is some data leakage in the predictor variables.



Since Late Code has the highest predictor variable weight AND contains “Late” in the name, I am suspicious that this could be the variable that tips the algorithm off on if it was classified as Late or not.

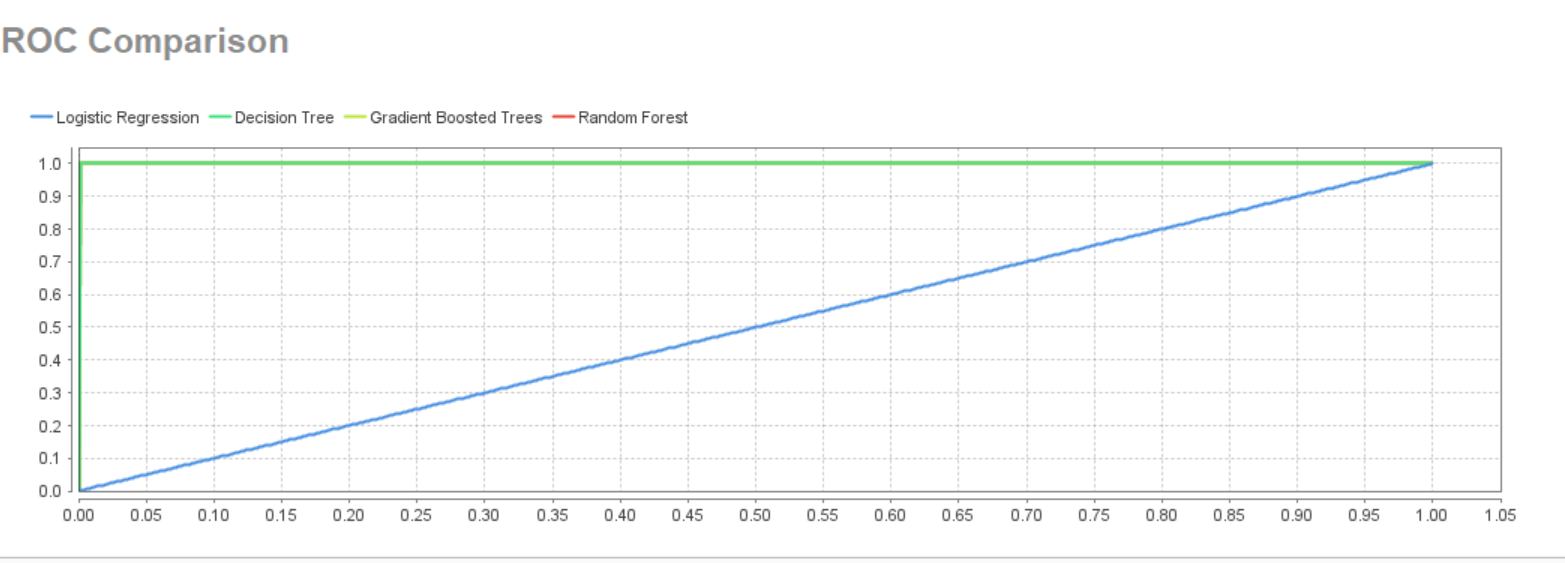


When I look at the data in Excel, it looks like Late Code might be specifying why the shipment is late. When the shipment is classified as on time, it appears this column is blank. I’m not sure that this data is available at the time the order is placed. At the same time, this information might be valuable in figuring out why the shipment was late. There might not be actionable guidance if the customer is on a credit hold, but there would be if a component was frequently late.

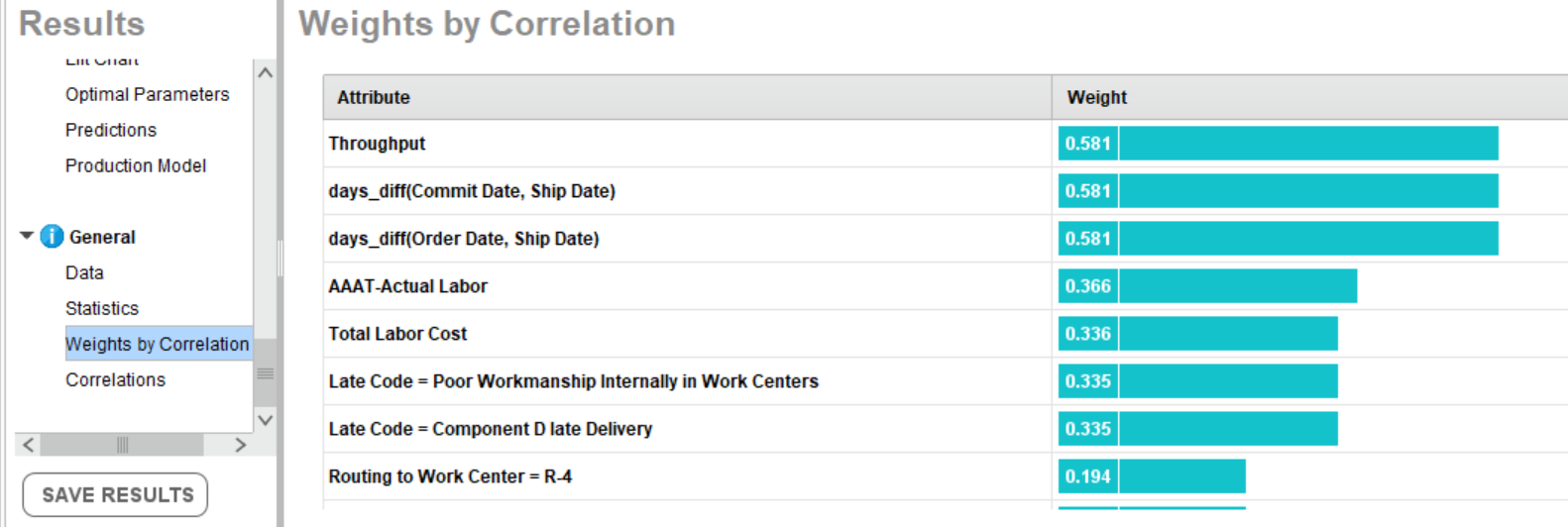


I also noticed high correlations of the predictor variables throughout the data set. As a result, I went to turboprep and decided to remove the correlation and dummy encode the Late Code column. I didn’t remove Late Code column even though I think there is data leakage going on there since we are trying to identify what drives late shipment and not necessarily trying to predict it ahead of time.

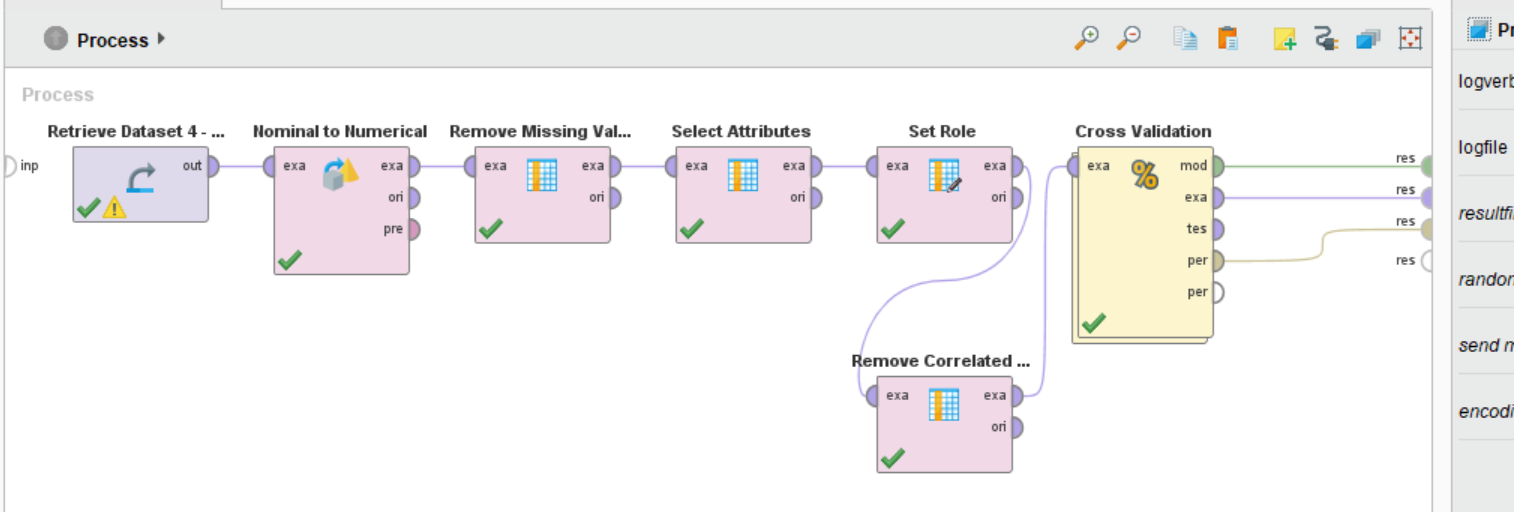
As a result, Logistic Regression significantly under performs now, but the rest of the models maintain a high AUC:

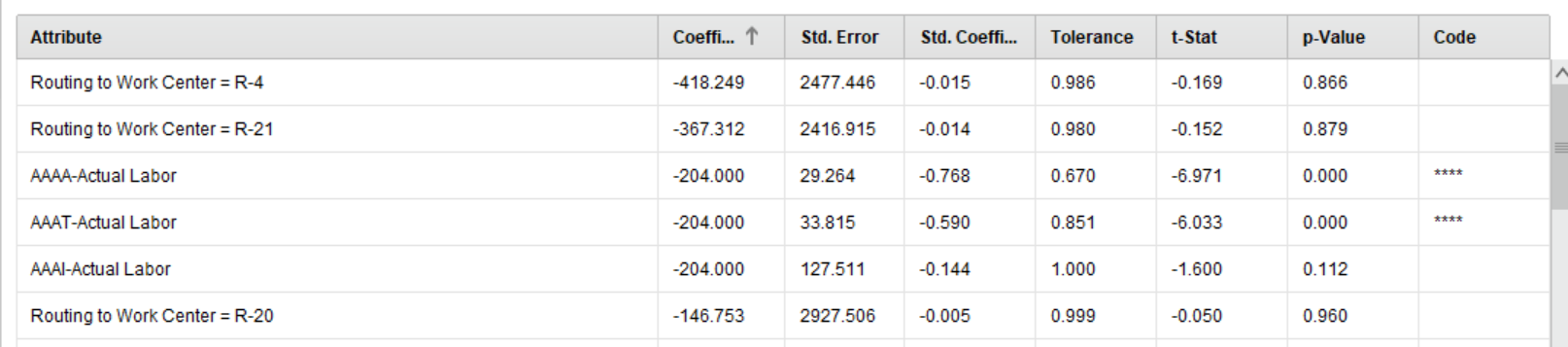


The below are the general weights by correlation produced by the auto model. This gives us some insight on what factors contribute to late deliveries. Some of these aren’t surprising. For instance, a higher throughput represents more orders. It could be that their production isn’t able to scale up well. This could be because there’s a bottleneck in the supply chain that is accentuated when the number of orders increase. In addition, the days between order date & ship date as well as the days between commit date and ship date are significant. This makes sense because the longer amount of time between these two events increase the overall timeframe from the customer’s order to delivery. Again, we are looking for the actual bottlenecks, not the result of those bottlenecks. I believe some of the answers are found in the lower correlations in the Late Codes. I would start with identifying the reasons why Component D is late. Do we need to switch vendors? What is going on with Work Center R-4? Are they underperforming? What is the correlation between Work Center R-4 and Late Code = Poor Workmanship Internally in Work Centers? Are there other work centers that frequently have products late because the order wasn’t produced correctly? There are all good places to start, ask questions, gather data, and repeat the process.



3. Continuing to use Dataset 4, perform predictive modeling using regression on Gross Profit as the dependent variable. What appear to be the most important predictors of Gross Profit? What are the implications for action by the business?

Results: 



The only significant contributors to gross margin was the label costs associated to AAAA and AAAT. This is difficult to assess because I believe there is a sweet spot with labor costs where you want to have highly trained, well performing employees that reduce mistakes and improve quality. I don’t think the answer would be to cut employment but maybe review the work these groups are doing and find out if it can be automated, made more efficient, or if somebody at a lower pay grade could perform the tasks.

4. Open the Superstore example in Tableau. Review the KPI layout and comment on how predictive analytics can help drive KPI performance?

Results: Data Visualizations such as the Superstore Dashboards provide a starting point in asking questions from your data. Similar to visualizing a process using event log data, plotting data points that represent how well an organization is succeeding at their KPI metrics, is a starting point to creating hypothesis from their data. For instance, the On-Time Shipment Trends tab has an area chart that quickly shows the percentage of items shipped on late, on time, and early. In the first half of the chart, the area that the shipped late takes up is less than the latter half. Why? Did they change vendors? Encounter a natural disaster, bad weather, etc. that delayed shipping? This is just one example of how data visualizations assist in the business intelligences and data analysis process. However, a visualization will not give you guidance on how to improve the KPI metric. This is where predictive analytics is more suited. The KPI that is measured (such as on-time delivery) can be made the target variable. Other factors of the organization and external stakeholders can be included to identify which predictive factors influence on time delivery the most (both positively and negatively)? Once this has been identified, the guidance can be used to strengthen the KPI metric.

5. Watch Trevor Hastie’s Video on Trees and Boosting for machine learning. <https://www.bing.com/videos/search?q=gradient+boosting+video&view=detail&mid=BC6160518DEF1ECD2379BC6160518DEF1ECD2379&FORM=VIRE>

Results:

The following are notes and concepts I took away from Trevor Hastie's discussion over Gradient Boosting Machine Learning. I found it very helpful in solidifying my understanding over this group of techniques.

Decision Trees:

Pros: Can handle large datasets, can contain both quantitative and qualitative predictors, ignores redundant variables, handles missing values, small trees easily interpretable.

Cons: large trees cannot be interpreted well, prediction performance is usually poor.

Bagging, Boosting, Random Forests all attempt to improve decision trees by some variation of model averaging. As a result of producing many trees and averaging the results, the variance is reduced.

# order by complexity/expected performance (1 lowest, 3 highest):

1. Bagging - Averages the results of many large trees built on resampled versions of the training data & classify by majority vote.

2. Random Forests - improves on bagging by de-correlating the trees to reduce variance. Introduces an additional randomness by selecting features at random. So many trees are produced using bootstrapped samples in addition to random features being included per decision tree made. Results are averaged.

3a. Boosting - Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote. Learns from the mistakes of previous fitted trees (this is when the weights of observations are reassigned to help correct previous mistakes).

3b. Boosting Stumps- 2-node tree, after single split. Surprisingly sometimes outperforms trees with 2+ nodes (ex: 10 or 100 nodes).

Boosting = Stagewise Additive Modeling - "stagewise" indicates that the parameters are being fit as the trees are being built and errors are being identified. It doesn't go back to optimize previous parameters. This slows down overfitting.

ex: Least Squares Boosting - a way that boosting can be applied to a regression problem. Essentially, the residuals get fitted into the problem. (Reminds me of Auto-Regression in ARIMA where the regression uses residuals to fits itself)

ex: Boosting can be applied to general loss functions = General Boosting Algorithms.

Gradient Boosting - works with variety of loss functions. Models include = regression, resistant regression, K-class classification & risk modeling.

- inherits PROS of decision trees (See above) & improves prediction performance.

- Provides variable importance plot.

Learning Ensembles - 2 steps: 1.) Build dictionary of trees 2.) Fit a model

- Result is large ensemble w/ many similar trees.

- Post-processing such as Lasso selects smaller subset of the trees and combines them.

6. Watch Salford Systems videos on decision trees for regression and classification:

<https://www.salford-systems.com/resources/introductory-videos/introduction-to-tree-based-machine-learning-regression>

<https://www.salford-systems.com/resources/introductory-videos/introduction-to-tree-based-machine-learning-classification>

Results:

The following are notes taken while watching the Salford System videos over CART Decision Trees for Regression:

CART = Classification and Regression Trees; divide X variables into different regions to better predict Y.

TERMS: Node at the top of the tree = root node. Tree Split = when an x variable is divided/partitioned. A split that divides one node into two = binary split. Terminal Nodes = node with no splits. Predicted Value = the average of Y for the records that fall into the terminal node.

Step 1: Grow a large tree

To compute the first split in CART: Calculate a split improvement for each X variable. The best and selected split is the one with the highest split improvement. To continue, compute the same steps, except on each partition of the data.

Step 2: Prune the tree

W/ Test Sample: Run test data through the large tree & subtress. Compute the error for each. The tree with the smallest test error is the pruned tree shown to the user.

Advantages: easily interpretable, variable selection, can handle redudant variables, models variable interaction, automatic nonlinear modeling, handles missing values (surrogate split) & outliers, transformations do not impact the model.

Interpretation:

Relative Error - when close to 0, CART is doing a better job than predicting the median/average for all records in the data. When it's 1+, CART is no better at predicting the average of the target variable for each record.

Relative Error = (CART Model error using Least Squares or Least Absolute Deviation)/(Error for predicting overall average for each record.)

Variable Importance: The sum of each variable's split improvement score across the splits in the tree. Increased when the variable is used to split a node or when it is the surrogate split.

Relative Importance (more easily interpretable compared to other variables): Calculated by taking each variable importance score and dividing it by the largest importance score. Then multiple by 100.

CART Tree as an equation:

Indicator Functions define a path from root node to the terminal node. If condition in Indicator Function = true then the output is 1, else 0.

Each terminal node has a function that uses an indicator function to specify the condition(s) multiplied by the predictive value. The tree function is the addition of each equation.

The following are notes taken while watching the Salford System videos over CART Decision Trees & Random Forest for Classification:

CART Decision Trees for Classification is similar to CART Decision Trees for Regression. The same terminal such as root nodes, terminal nodes, leaf nodes, etc. apply. For the splitting procedure, the data is subsetted by the features with best split improvement. When the pruning process begins, a test set or cross validation can be applied. When using a test set, the data will be evaluating by running it against the large tree and smaller trees that are the subsets of the largest. Test error is measured for each tree. The tree with the smallest test error is the selected tree. This method can also handle missing values (by a surrogate observation), nonlinear data, handle outliers, and provide interpretable results using relative cost.

Random Forests - If a variable is important then randomly rearrange then its value results in a decline in model performance.

In CART it will fit multiple trees to independent bootstrap samples of data and combine predictions. This results in reduced variance compared to a single CART decision tree.

Some features of Random Forest: variable selection, variable interaction detection, nonlinear relationship detection, missing value handling, outlier handling, and model local effects.

CON: not as interpretable as a single CART tree.

Like CART Decision trees for regression and classification, Random Forest will deliver variable importance on a relative scale.