

**Special Pricing Project**

**May 5th, 2020**

**Team 6**

**Alan Market**

**Alyson Chen**

**Fairy Gandhi**

**Sumner Crosby**

**Wenbin Yang**

Contents

[1. Summary](#_heading=h.gjdgxs) 3

[Model metrics](#_heading=h.30j0zll) 5

[Economic impact summary](#_heading=h.3znysh7) 5

[Recommendations 5](#_heading=h.3dy6vkm)

[2. Data 5](#_heading=h.1t3h5sf)

[Summary 5](#_heading=h.4d34og8)

[Cleaning approach 5](#_heading=h.2s8eyo1)

[3. Recommended Price Approach 6](#_heading=h.17dp8vu)

[4. Model 1](#_heading=h.3rdcrjn) 8

[Summary](#_heading=h.26in1rg) 8

[Model to Predict Special Pricing Approval](#_heading=h.lnxbz9) 7

[Determination to Send to Pricing Managers and Economic impact model](#_heading=h.35nkun2) 7

[Learning](#_heading=h.1ksv4uv) 7

[5. Model 2 6](#_heading=h.44sinio)

[Summary 6](#_heading=h.2jxsxqh)

[Model to Predict Special Pricing Approval 6](#_heading=h.z337ya)

[Determination to Send to Pricing Managers and Economic impact model 6](#_heading=h.3j2qqm3)

[Learning 6](#_heading=h.1y810tw)

[6. Model 3](#_heading=h.4i7ojhp) 11

[Summary](#_heading=h.2xcytpi) 11

[Model to Predict Special Pricing Approval](#_heading=h.1ci93xb) 11

[Determination to Send to Pricing Managers and Economic impact model](#_heading=h.3whwml4) 12

[Learning](#_heading=h.2bn6wsx) 12

[7. Model 4 7](#_heading=h.qsh70q)

[Summary 7](#_heading=h.3as4poj)

[Model to Predict Special Pricing Approval 7](#_heading=h.1pxezwc)

[Determination to Send to Pricing Managers and Economic impact model 7](#_heading=h.49x2ik5)

[Learning 7](#_heading=h.2p2csry)

[8. Model 5](#_heading=h.147n2zr) 14

[Summary](#_heading=h.3o7alnk) 14

[Model to Predict Special Pricing Approval](#_heading=h.23ckvvd) 15

[Determination to Send to Pricing Managers and Economic impact model](#_heading=h.ihv636) 15

[Learning](#_heading=h.32hioqz) 16

# Summary

## Model metrics:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Sensitivity** | **Specificity** | **Precision** | **Accuracy** | **MSE** |
| **Neural Network** | **67.5%** | **78.3%** | **77.8%** | **72.5%** | **0.1808** |
| **Linear Regression** | **10.7%** | **93.4%** | **36.4%** | **71.6%** | **0.5329** |
| **Logistic Regression(1)** | **10.2%** | **93.3%** | **37.6%** | **69.7%** | **0.5505** |
| **Logistic Regression(2)** | **38.3%** | **90.3%** | **31.9%** | **84.9%** | **0.6831** |
| **GBM Model** | **20.3%** | **98.2%** | **48.2%** | **92.2%** | **0.0651** |

Based upon the above table, the Neural Network model seems to be performing very well. It has strong precision and sensitivity metrics. However, the GBM model seems to show strong statistics for Accuracy and Specificity. It also presents the lowest MSE value of 0.065. More results of each model and an overall recommendation will be presented below.

**Economic Impact Summary\*:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Initial Cost** | **Savings** | **Adjusted Cost** |
| **Neural Network** | **$149,757,435.73** | **$29,631,027.38** | **$120,126,408.35** |
| **Linear Regression** | **$208,796,609.31** | **$71,286,777.28** | **$137,509,832.03** |
| **Logistic Regression (1)** | **$202,812,481.05** | **$69,121,760.10** | **$133,690,720.95** |
| **Logistic Regression (2)** | **$246,176,264.70** | **$80,513,041.84** | **$165,663,222.86** |
| **GBM Model** | **$272,510,957.16** | **$99,164,278.62** | **$173,346,678.55** |

## *\*Note: Initial Cost and Saving are scaled to control for consistent data size, so they may be a bit skewed.*

As the above table shows, the neural network model seems to have the best economic benefit, which the GBM model, which seems to be the best performing in terms of key metrics, seems to be the worst, with an adjusted cost of $173,346,678.55. These results may be a bit skewed as they are scaled based upon different data subsets. Because there is a limited amount of datapoints that were actually sent to special pricing and received approval, the above metrics need to be taken with a grain of salt.

## Recommended Price

In the Arrow project, we have information about only 2 parts (customers, product) out of 3 parts (Customer Segment). We have very little information on the customer segment. So, we will use those two parts of information in creating a recommended price to compare against the list price. We have two more files available, ‘Product.csv’ and ‘Customer.csv’.

In the customer file, ‘revenue’ and ‘cost’ is the mock data derived from another distributor. ‘Gross profit’ is calculated as ‘revenue’ – ‘cost’. ‘Total lines’ represent a number of lines across the quote ids for the particular ‘customer number’. ‘Gross profit’ column by default is in factor format, converted it to numeric format using ‘as.numeric (as.character (x))’ function. Then I sorted the ‘customer’ dataset by the ‘gross profit’ column in descending order. I created a plot of the ‘discount’ column (y-axis) and the ‘gross profit’ column (x-axis). The plot clearly says that customers that generate more profit are given further discounts. The plot also depicts that customers that generate less than six thousand dollars profit are given zero discount.

Coming to the product file, the ‘list price’ and ‘list cost’ are derived from Arrow data. ‘Cumulative lines’ are calculated by adding previous ‘lines’ number + current ‘lines’ number. ‘Discount’ is calculated based on ‘cumulative line percentage’. If you sort the products from most lines to the least number of lines, you will get Products that make up 80% of the lines are considered 'A' category. Products that make up the next 15% (95% total) of the lines are considered 'B' categories. Products that make up the next 4% (99% total) of the lines are considered the 'C' category. Products that make up the next 1% (100% total) of the lines are considered 'D' categories. ‘Lines’ column by default is in factor format, converted it to numeric format using ‘as.numeric (as.character (x))’ function. Removed comma (,) from the numeric values in the ‘lines’ column using gsub() function. I created a plot of the ‘discount’ column (y-axis) and the ‘cumulative lines’ column (x-axis).

Our approach for calculating the recommended price was as follows: ‘Customer’ dataset was merged with ‘Arrow’ data (main) with a common key, ‘customer number’. ‘Product’ dataset was merged with ‘Arrow’ data (main) with a common key, ‘manufacturer part number’. Both the ‘customer’ discount and ‘product’ discount are added and labeled as ‘total discount’. The ‘total discount’ is then applied to the list price of the ‘arrow’ dataset. That price is named as ‘recommended price’ (a new column is created for that). ‘Recommended price’ is then subtracted from ‘List cost’ to check if there is any negative profit. My way of interpolating was if there is any negative profit, I changed it to the original ‘list price’.

**Recommendations**  
Price Recommendation

Our way of approaching the price recommendation section was as follows:

We compared the recommended price with the actual price by calculating the difference between recommended price and the actual price (i.e., the list price). The ‘price difference’ was then converted to percentage. We grouped the count of the ‘price difference’ based on the percentages. You can find more details in the below table.

|  |  |
| --- | --- |
| **Delta Between Recommended and Actual** | **Count** |
| 1% | 11,621 |
| 2% | 22,235 |
| 3% | 31,763 |
| 4% | 58,710 |
| 5% | 71,329 |
| 6% | 85,513 |
| 7% | 102,551 |
| 8% | 134,859 |
| 9% | 168,078 |
| 10% | 188,689 |
| 11% | 212,386 |
| 12% | 243,412 |
| 13% | 279,162 |
| 14% | 352,117 |
| 15% | 400,489 |

As seen from the table above, there are more price differences that are higher than 15 percent. For those, we can recommend the special price. However, for percentages from 1 to 5, there’s not much price difference between recommended and actual price, hence we would recommend the recommended price.

Special Price Recommendation

Based upon the below models, our group would recommend using the GBM model to assist Arrow in forecasting which orders will receive special pricing, based on a number of inputs. While the neural net model exhibits a higher precision rate, the GBM seems the most scalable and was cross validated with 10 different data subsets, indicating strong confidence in the result. The GBM was able to predict almost 50% of actual special pricing dynamics, and would save Arrow money and time. In addition, the GBM model takes much less computing power and resources, making it easier to test, adjust, and implement. The model was fairly simple, so it would not be hard for a staff member to adjust or add to the model for even better predictive power. The neural network did exhibit better cost metrics in the above table, but this team feels this result is skewed and needs to be taken with a grain of salt. Overall, the GBM performed quite well and it is the recommendation of this report that Arrow Electronics implement a similar model to predict which orders will receive special pricing before sending order to the Pricing Manager.

# Data

## Summary:

The main dataset that we used for our analysis was provided to us by Arrow and contained information on orders from Q1 2018 including: Part number (hashed), Customer name and number (hashed), list price, list cost, whether the order was sent to a product manager, whether the order received autocost special pricing, whether the order received traditional special pricing, the ICC and taxonomy of the product being ordered, and the industry of the customer. There were around 4.5 million observations in this data set.

Additionally, we were given two supplementary datasets to augment our analysis, and these data sets contained imputed details on the size and value of the customers which were calculated by taking data from a comparable distributor and scaling it to reflect the size of Arrow. In addition to providing mock customer level data for revenue, gross margin, and average line value, we also were given information on likely discounts based on the size of given customers.

## Cleaning approach:

## As with most modeling applications, data preparation was a crucial part of forming the following models. In the data initialization code, the data is loaded for further analysis and some primary cleaning, such as simple character vector restructuring and character to numeric transformations, is performed. In addition, some columns are forced to be a logical type, in the form TRUE or FALSE. Three datasets are used for this modeling exercise, as highlighted above, so combining this data into one usable set was an important piece of data cleaning. The three datasets were combined using the merge function across similar customer and product type indexes, creating one centralized dataset of over 4 million data points.

## 

## The last step was to drop some unnecessary columns and to fix the sent\_to\_PM category. When special pricing might be supported, the order is sent to a pricing manager, who then decides whether or not the order will receive a special price. According to the data, there were a number of orders that were not sent to the PM but received special pricing, so to normalize the data, every order that received special pricing was altered to ensure it was sent to the PM.

# Recommended Price Approach

# Model 1: Neural Network (Alan Market)

## Summary:

Neural networks excel in being able to make accurate predictions in complicated systems and are capable of picking up on nuanced patterns in intricate data. Because of the potential upside and predictive capabilities of a neural net, we decided that it should be one of the predictive techniques that we used to predict special pricing probabilities.

To build our neural network (NN) we used the NeuralNetwork package in r. The architecture of the hidden layers in the final nn was three nodes, two nodes, one node, and then was fed into the output layer. Our tuning parameter for training the model was the sum of squared error, and our train-test split was 5-95. This very small training set proportion was one of the major limitations of this model. Training the nn is very computationally intense, and my local machine was not able to handle loads larger than this. With that said, if Arrow has access to distributed computing methods, this model could be improved by altering the train-test split to something more aggressive such as a 60-40 split.

Despite the low training percentage, the performance of the model was very decent. The accuracy of the model was 72.5%, and the true negative/specificity was 78.3%. Furthermore, there is a small advantage of the low training-test ratio, and that is the model is very unlikely to be overfitting the data.

## Economic impact model:

While the model performs well from a predictive standpoint, it is not yet at the stage where it could be deployed in the field and expected to save Arrow money. Due to the high cost associated with false negatives (predicting an order will not get special pricing when it actually would have), the overall impact deploying the model would be a loss in profits due to a reduction in win rate across orders which would have received special pricing. In the test set, the model caught 222,066 true negative orders but also flagged 104,135 false-negative orders. The savings from the true negatives amounts to $5,551,650 whereas the cost of potential lost profits totals over $14,000,000. Ultimately, the model would generate a loss of $16,262,922 when all of the costs and benefits are accounted for.

**Confusion Matrix:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** | **Total** |
| **Actual Positive** | **215,840** | **104,135** | **319,975** |
| **Actual Negative** | **61,606** | **222,066** | **283,672** |
| **Total** | **277,446** | **326,201** | **603,647** |

**Confusion Matrix Costs & Benefits:**

|  |  |  |
| --- | --- | --- |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** |
| Actual Positive | $25.00 | $67.00 |
| Actual Negative | $25.00 | -$25.00 |

**Confusion Matrix Total Costs:Ｑ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** | **Total** |
| Actual Positive | $5,396,000 | $14,878,422 | $20,274,422 |
| Actual Negative | $1,540,150 | -$5,551,650 | -$4,011,500 |
| **Total** | **$6,936,150** | **$9,326,772** | **$16,262,922** |

## Learning:

My main takeaway from developing this model was that even if a model is performing well on paper, it might not be performing well enough to satisfy the business objective it is being developed for. I do believe that a nn would be a great tool to use for flagging orders, but to build it properly I would need to have more computing resources available to me. It may be the case that, while a nn could work for this predictive task, a simpler and less computationally intense model may actually be better suited to the job at hand.

# Model 2: Linear Regression (Alyson Chen)

## Summary

## Model to Predict Special Pricing Approval

In Statistics, lm is a linear approach to investigate and model the relationship between a dependent variable and one or more than one variable. However, there are some pros and cons while using lm as a prediction model. Start with advantages. Firstly, lm have better performance when the dataset is linearly separable. Second, lm is easy to implement especially to cope with large scale datasets. In this particular project, we have a large dataset. Hence, using the lm model would help us to improve the working efficiency. Lastly, lm is prone to overfitting but can be avoided using some reduction technique which can reduce the working time and improve working efficiency.

To start our prediction model using linear regression in R, we set out our prediction target- to see if special pricing supported our outcome, y variable and as for x variables we total line, discount, and costs for our dependent variables. After that, we split the training and testing set into 50/50. Eventually, we have a 0.7159493 accuracy rate.

## 

## Economic impact model

In our confusion matrix, we have 64006 in the test set for true negative orders and have 535,791 for false negative orders. and we have sensitivity 10.7%, specificity 93.4%, precision 36.4% and accuracy 71.6%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Confusion** | **Matrix** |  |  |  |  |  |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** | **Total** |  | Sensitivity | 10.7% |
| Actual Positive | 64,004 | 535,791 | 599,795 |  | Specificity | 93.4% |
| Actual Negative | 111,820 | 1,572,583 | 1,684,403 |  | Precision | 36.4% |
| **Total** | **175,824** | **2,108,374** | **2,284,198** |  | Accuracy | 71.6% |
|  |  |  |  |  |  |  |
| **Confusion Matrix Costs** | |  |  |  |  |  |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** |  |  |  |  |
| Actual Positive | $25.00 | $67.00 |  |  |  |  |
| Actual Negative | $25.00 | -$25.00 |  |  |  |  |
|  |  |  |  |  |  |  |
| **Confusion Matrix Total** | | **Costs** |  |  |  |  |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** | **Total** |  |  |  |
| Actual Positive | $1,600,100 | $105,363,061 | $106,963,161 |  |  |  |
| Actual Negative | $2,795,500 | -$39,314,575 | -$36,519,075 |  |  |  |
| **Total** | **$4,395,600** | **$66,048,486** | **$70,444,086** |  |  |  |

## 

## Learning

After finishing up this particular project, I have gained lots of knowledge related to analytics, pricing forecasting and prediction. Moreover, by using a real-world dataset, it gave me practical experience and helped me to prepare and enhance my skills in my future works. In my mind, the most challenging part of this project is to find out the most useful columns and merge them all together. For me, it is quite challenging to see what are the “useful” and “not useful” columns. However, I have a great time working with my teammates and also able to practice my communication skills by meeting them online.

# Model 3: Logistic Regression (Fairy Gandhi)

## Summary (Model to Predict Special Pricing Approval) We decided to use logistic regression as one of the five modeling techniques to predict ‘special pricing’ using ‘gross\_margin’, ‘taxonomy’, ‘gross\_profit’, ‘total\_lines’, and ‘customer\_number’ variables. The training and test set was split using an 80:20 ratio. We can even use glm () function (‘generalized linear models’) instead of lm () because glm fits a class of models that includes logistic regression. The syntax of the ‘generalized linear models’ function is quite similar to that of the lm () function. The only thing we need to be careful about is that we must pass in the argument “family=binomial” in order to tell R to run a ‘logistic regression’ rather than some other type of generalized linear model.

## After running the glm () function, we can use the predict function to get the probability about the special price. If you put the type="response" option in the predict () function, it tells ‘R’ to output probabilities of the form P (Y = 1|X). By default, it takes a ‘training’ dataset (used to fit the logistic regression model) to compute the probability if no dataset is specified. We can also convert the predicted probabilities into class labels (It is totally optional). Given these predictions, we then used a table function to produce a confusion matrix to determine how many observations were correctly or incorrectly (False positive, False negative, etc.) grouped.

## The advantage of using logistic regression through generalized linear models is its fast computation. It does a good job even for complex problems considering only meaningful variables are included.

## Economic impact model The model shows low precision but decent accuracy. Low precision indicates that the proportion of the data points our model says were relevant actually were not that relevant.

## Accuracy, on the other hand, states how close a measured data point is to the true value. Being accurate doesn't mean that we are precise too.

## As you can see from the confusion matrix, “False negatives” are equal to 48,790. The cost of each False negatives is $67 (from 'Confusion Matrix Costs' table) which brings the total cost of “False Negatives” to be equal to $41,381,813. It's almost more than double the total cost of “True negatives”.

## 

|  |  |
| --- | --- |
| **Sensitivity** | **38.8%** |
| **Specificity** | **90.3%** |
| **Precision** | **31.9%** |
| **Accuracy** | **84.9%** |

## **Confusion Matrix:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** | **Total** |
| **Actual Positive** | **30,988** | **48,790** | **79,778** |
| **Actual Negative** | **66,140** | **617,639** | **683,779** |
| **Total** | **97,128** | **666,429** | **763,557** |

## **Confusion Matrix Costs & Benefits:**

|  |  |  |
| --- | --- | --- |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** |
| **Actual Positive** | **$25.00** | **$67.00** |
| **Actual Negative** | **$25.00** | **-$25.00** |

## **Confusion Matrix Total Costs:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** | **Total** |
| **Actual Positive** | **$774,700** | **$41,381,813** | **$42,156,513** |
| **Actual Negative** | **$1,653,500** | **-$15,440,975** | **-$13,787,475** |
| **Total** | **$2,428,200** | **$25,940,838** | **$28,369,038** |

**Learning**

This project has taught me that how working for real-time data is different from what we do in other courses. It's a great way to apply what we have learned so far. Plus, working in a team really makes a difference since everyone has different inputs to provide which eventually makes you look at more than one perspective. Technically speaking, I should have also used the knn model to compare my results with logistic regression since we are working on a classification problem. The capability of knn to outperform more powerful classifiers makes it even more interesting.

# Model 4: GLM (Wenbin Yang)

## Summary

I chose logistic regression for this prediction model this time. There are many advantages to logistic regression. First, the logistic regression algorithm is relatively mature and the prediction is more accurate; second, the coefficients obtained by the model are easy to understand and easy to explain; the most important point is that the calculation of logical regression is small, which is convenient for later Inspection and testing. Based on the above advantages, and considering the number of our data sets, we decided to split our train-test into 50-50.

Because the size of the training set can affect the accuracy of the model, and the more training sets, the higher the degree of model fitting in the later period. Due to the huge difference in values between each column of the sample, some of the columns have relatively large values, which will affect the accuracy and direction of the entire model. So we first use zero-mean normalization to normalize the columns that need to be used. In another word, Z-Score uses (x-μ) / σ to convert two or more sets of data into unitless Z-Score scores, which unifies the data standards, improves data comparability, and weakens data interpretability.

At the beginning, we took most of the data columns as independent variables, and then used the step function to sort out the columns that have the greatest impact on the model, and then repeatedly modified and fitted them to make the model more accurate. Finally, the accuracy of the logistic regression model has increased from the initial 53% to 69.7%, and Specificity is 93.3%.

## Model to Predict Special Pricing Approval

## Economic impact model

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Confusion Matrix** |  |  |  |  |  |  |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** | **Total** |  | Sensitivity | 10.2% |
| Actual Positive | 66,063 | 582,860 | 648,923 |  | Specificity | 93.3% |
| Actual Negative | 109,761 | 1,527,214 | 1,636,975 |  | Precision | 37.6% |
| **Total** | **175,824** | **2,110,074** | **2,285,898** |  | Accuracy | 69.7% |
|  |  |  |  |  |  |  |
| **Confusion Matrix Costs** | |  |  |  |  |  |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** |  |  |  |  |
| Actual Positive | $25.00 | $67.00 |  |  |  |  |
| Actual Negative | $25.00 | -$25.00 |  |  |  |  |
|  |  |  |  |  |  |  |
| **Confusion Matrix Total Costs** | |  |  |  |  |  |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** | **Total** |  |  |  |
| Actual Positive | $1,651,575 | $102,323,338 | $103,974,913 |  |  |  |
| Actual Negative | $2,744,025 | -$38,180,350 | -$35,436,325 |  |  |  |
| **Total** | **$4,395,600** | **$64,142,988** | **$68,538,588** |  |  |  |

## Learning

I actually learned a lot from this project.

First, I knew we could use Z-Score normalization to normalize the columns with a huge gap. Z-Score normalization is a good way to unifies the data standards, improves data comparability, and weakens data interpretability.

Second, I learned we can use the “step” function to pick the columns which are useful for the model.

Third, a model cannot only find out the relevant columns in it, but also need to constantly adapt and modify it to achieve a good accuracy. This requires a lot of experience and practice.

# Model 5: GBM (Sumner Crosby)

## Summary

Model 5 implements a Generalized Boosted Regression Model (GBM). GBMs are a popular machine learning algorithm with proven success across many domains and are a good alternative to generalized linear models when predictieve power is low. GBM models are similar to a random forest regression, with the main difference being that where a random forest builds a group of deep independent trees, GBMs build a system of shallow and weak successive trees with each tree learning and improving on the previous. When combined, this group of successive trees produce a powerful “committee” that is often hard to beat with other algorithms. After testing some preliminary logistic generalized linear models that exhibited low predictive power, the GBM model was explored to create a more useful predictive model. A number of modeling parameters were tested, with the below model parameters creating the most robust model.

## Model to Predict Special Pricing Approval

## After testing a number of GLM and GBM models, three input variables were used for the final model, which are as follows: total\_lines, gross\_margin, and inventory\_class. total\_lines serves as an indicator of client size, which was shown to play a significant role in predicting special pricing. gross\_margin is a statistically significant control for a customer’s gross margin across all orders, and again serves as a size and profitability indicator. Inventory\_class is a variable that splits the product into 4 classes, and this serves as a control for product type. Other variables to control for product type or statistics were added to the model, but due to the massive time increase of running the model and a low impact, these variables were omitted in the final model. To ensure the test and training subsets were not creating unbalanced results, the model was run across 10 different subsets, with a very similar result each time. Therefore, the model seems to work across the data. The GBM model takes a number of inputs, which are highlighted in the below table.

|  |
| --- |
| Parameter Value  Distribution Bernoulli  n.trees 1000  interaction.depth 1  shrinkagE 0.01  keep.data TRUE |

## Economic impact model

## Confusion Matrix Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** | **Total** |
| Actual Positive | 14,257 | 56,009 | 70,266 |
| Actual Negative | 15,342 | 828,752 | 844,094 |
| **Total** | **29,599** | **884,761** | **914,360** |

**Confusion Matrix Costs**

|  |  |  |
| --- | --- | --- |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** |
| Actual Positive | $25.00 | $67.00 |
| Actual Negative | $25.00 | -$25.00 |

**Confusion Matrix Total Costs**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual/Predicted** | **Predicted Positive** | **Predicted Negative** | **Total** |
| **Actual Positive** | **$356,425** | **$55,526,384** | **$55,882,809** |
| **Actual Negative** | **$383,550** | **-$20,718,800** | **-$20,335,250** |
| **Total** | **$739,975** | **$34,807,584** | **$35,547,559** |

## Learning

Based upon the above tables, the model performed fairly well, predicting about 48% of special pricing requests correctly, which is a large improvement over the current pricing strategy. Unfortunately, it did miss some predictions, which are costly, but it certainly will save the company money to implement such a policy. This dataset was much cleaner than I had expected, making it a bit easier to work with. That being said, I learned more about the computing limitations of my computer, which somewhat limited some modeling exploration. With more computing resource and a bit more time, this model could have performed better, but given the resources given, it performed much better than the previous project and could provide Arrow with material operations savings.

. . .