Kimberly Cable
DSC 630 – Case Study
Oct 1, 2022

Help Desk Case Study

Introduction

A large US Corporation that manufactures, sells, and services their products of hardware devices needed help with their help desk efficiency. Specifically, it needed help to determine if a part was needed in a service call after the initial phone call support determined a service call was needed. First, customers call the help desk to determine the problem, and a support ticket is generated. A support engineer is then assigned the ticket and they try to resolve the problem over the phone. If the call isn't successful, the engineer goes to the customer's site to resolve the problem. The real problem arose when the engineer got to the customer's site and if they had the proper part needed to make the. Knowing if a part was needed for the support ticket was the question for the analysts to predict.

Data was collected from previous support tickets, and they had more than one million tickets to work with. Their dataset consisted of a unique ticket ID, date and time of the call, the business or individuals name that generated the ticket, the country where the problem occurred, any error codes the hardware was displaying, the reason for the call, any warranty information that was left on the hardware, a description from the agent about the problem the customer was having, and the outcome of the ticket along with any parts that were needed to fix the problem.

Methods and Results

To prepare the data, all the variables were examined and corrected if necessary. The date of the call was changed to just the day of the week and the business name was removed as it was not necessary for the model. The target variable was if there was a part needed or not (Yes or No). Since this was a categorial answer, this was changed to a dummy variable either as 1 for a part was needed or 0 for a part was not needed. Unfortunately, they found the distinction between yes, a part was needed or not wasn't as straightforward as was intended. In some support tickets, if a part was not needed for the repair, it was still coded as a 1, as the part was needed. In other support tickets, if a part was needed but it wasn't the part specified in the ticket, the part's needed flag was coded as 0 because the engineer closed the ticket due to the incorrect part and opened a new ticket for the correct part. The last problem they found with the part's needed flag was if the part was needed and installed the ticket was closed but later it had to be reopened when the problem returned. The part's needed flag was coded as a 1. If the problem was resolved using a different part the code remained a 1 but if the part was not needed to repair the problem, then it was changed to a 0. Ultimately, in the end, the coded values were difficult to find in the million records they had and were coded throughout the dataset, so the codes were left as they were.

The next step the analysts had to take was getting features from the call's description as it was text typed in by the support engineer's initial phone call. Problems such as misspellings and abbreviations were found. They learned the database programmers knew of this and proactively was fixing many of these errors, to the delight of the analysts. Along with domain experts and database programmers, they were able to come up with about 600 keywords and phrases to stem the data and extract the features from the descriptions.

The team chose to use a decision tree as their model for this analysis. They chose it because decision trees can handle a large number of categorical features and they can be interpreted very easily when it was completed. They tested out many different splits of proportions of training and test data sets. The team used hundreds of decision trees and different variable subsets of keywords in conjunction with different settings for prior and complexity penalties.

Upon completion, they found that about 50.5% of records fell short of the minimum value required in the initial business objective. This included a lift (the target response divided by the average response) of more than 2 over the baseline rate of 23.2%. Looking at this they found that the problem wasn't the ability of the decision tree to handle a large number of categorical features but the inability to find combinations of the features to result in a high percentage of support tickets that needed parts. The analysts found it was due to the number of dummy variables (from the transformation of categorical data) was not enough and didn't provide the model with enough information to create nodes from. This, they called sparse variables.

The team decided to go back to the drawing board. They asked themselves several questions like "What additional information is known about problems at the time tickets are submitted?" and "How would an informed person think about the data to solve the problem?" They looked at what a support engineer does when deciding what part is needed or if none is needed. They found that the engineer looks back at what they did previously, what worked and what didn't, to determine if and what part is needed for the repair. This was called "temporal information". So instead of creating dummy variables from the keywords found in the ticket's description, they used a percentage of times the keyword(s) used needed a part in the past year. It was determined that if they went further back the process was too old and couldn't be trusted. What this brought into the model was the historical data as to if a part was used or not as well as how many tickets contained those keyword(s).

The analysts used a decision tree for this model as well for the same reasons as the previous model. This model, they found, also did not get the results that were necessary for the business objective, but it did achieve a few different results that were different and encouraging. What they found in this model was that a few end nodes in most of the trees did meet the business objectives of if a part was needed or not in the support ticket. These branches also had a few different ways to predict if a part was needed and many had 6 or 7 steps to predict if a part was needed or not, but the domain experts understood its findings enough that it was a good model.

Conclusion

After creating approximately 10,000 decision trees which then created about 20,000 terminal nodes, they found they had a high percentage of branches that predicted if a ticket needed a part or not. They took each branch and created a series of rules with “and” conditions and ranked them from the highest percentage to the least. Once this was done, they created an algorithm for applying the rules to each support ticket. The company liked this idea because they could then convert the rules to SQL for easier deployment. The model went into production and was successful enough that they expanded the program ultimately causing the company to save some money in the end.

The team found that coming up with new ways to create features made for a better way to solve the parts problem for the help desk. Thinking like the help desk engineers allowed them to have more successful features for better results. They also learned that having domain experts on the team was crucial to their success. I found this study very interesting as I was once a call center agent that did technical support for a mobile phone carrier. Knowing that the agents were good experts was a good thing and something intuitively I already knew.

References

Abbott, D. (n.d.). Help Desk Case Study. In D. Abbott, *Applied Predictive Analytics* (pp. 402 - 411).