**Case Study: Help Desk**

Thip Rattanavilay

Bellevue University DSC-630

**Overview**

In this case study for help desk operations is developed. Help desk personnel answer calls and emails relating to customer or client complaints. They address technical issues as needed, providing high levels of customer service. The role often requires both software and hardware knowledge to address issues that arise. This study relates in predicting if a part would be needed to resolve the problem. Since incident estimation for hundreds of products is time-consuming, we use cluster analysis to group similarly behaving products in clusters, for which we then estimate incidents based on the representative product in the cluster. Incidents are predicted using a model. The time to resolve the incidents is predicted using historical labor data for the resolution of incidents. Cluster analysis is used to group products with similar help desk incident characteristics. We use Principal Components Analysis to determine one product per cluster for the prediction of incidents for all cases of the cluster, so as to reduce estimation time spent if parts are needed to fix the issue. We were able to predict incidents for a cluster based on this product alone and do so successfully for all clusters with accuracy comparable to making predictions for each product in the portfolio. Decision trees and Linear regression is used with parts data for the resolution of incidents to relate incident predictions for help desk.

**Business Understanding**

The features in the dataset describe the characteristics of the support ticket with each row containing a unique ticket ID and columns contained descriptions of the ticket. The dataset contains more than 1 million tickets.

Each support ticket, the following variables were available as candidate inputs for models:

1. Date and time of the call
2. Business/individual name that generated the ticket
3. Country of origin
4. Device error codes for the hardware
5. The reason for the call that generated the ticket: regularly scheduled
6. maintenance call or a hardware problem
7. Warranty information
8. The problem description (text transcribed from the call)
9. The outcome of the ticket: ticket closed, solution found, and so on
10. The part(s) needed to close the ticket

The objective variable had two sections: Did the ticket require a Part to close, and what part or parts were expected to close the ticket. Just the Parts Used objective variable models are portrayed for this case study.

**Defining a problem**

Part was used to fix a problem that was not the one specified on the ticket. If this situation was discovered, the PartsUsed flag was set to 0 because a part the engineer specified to close the ticket was not actually needed to fix the problem on the ticket, and the engineer should have opened a new ticket for the second problem. These ambiguities were difficult to discover, however, and it is presumed that these problems persisted in the data.

Data used from Help Desk Text, Keywork STICK included all the variants listed. Created a 1/0 dummy variable indicating if the ticket required a part to complete the repair. The target variable was labeled the PartsUsed flag.

PartsUsed

* ID
* WORD
* KEYWORD
* COUNT

**Problem 1**, if a part was not necessary for the repair, although it helped the repair, it was still coded as a part being used.

**Problem 2**, a part was used to fix a problem that was not the one specified on the ticket. If this situation was discovered, the PartsUsed flag was set to **0** because a part the engineer specified to close the ticket was not actually needed to fix the problem. Open new ticket for second issue.

**Problem 3**, when a fi x was made using a part and the ticket was closed, only to reopen later when the problem reappeared. If the problem was subsequently solved using a different part, the PartsUsed flag would still be coded as a **1** (though a different part was used to fix the problem).

**Defining the Target Variable**

For this case study, the target variable is very obvious; Each keyword dummy variable was a sparse variable, populated in a small minority of tickets, and individual keywords didn’t necessarily provide enough information on their own to create good splits. Variable used for this case study

* CLASS
* NoParts
* Parts
* Total

**Data Understanding:**

In data understanding, we can observe all the features for data types. For this dataset, except for the PartsUsed column, all columns are of numeric type. The Id column is only for observation identification purposes. KEYWORDS column as two values ID and COUNT. We can check the distribution of values for each column. We can also use Feature support to find out the p-score; so that we decide to remove any feature that is not having any impact on the target variable.

1. Date and time of the call
2. Business/individual name that generated the ticket
3. Country of origin
4. Device error codes for the hardware
5. The reason for the call that generated the ticket: regularly scheduled
6. maintenance call or a hardware problem
7. Warranty information
8. The problem description (text transcribed from the call)
9. The outcome of the ticket: ticket closed, solution found, and so on
10. The part(s) needed to close the ticket

**Data Preparation**

This dataset needs very little data preparation. We can remove the id column as it is only for identification purposes. Before removing the identity property, we can check for duplicate records and get rid of them. We can check for the data type of each column value. If needed, we can convert them to the numeric type for COUNT. We can transform the PartsUsed column to binary values for ID. We will replace the value with 0 and the value with 1. We will inspect the distribution for all columns and will remove records with outliers if there are any. We can replace all missing values with Iterative Imputers. The Iterative Imputer will observe all other column values and compare them with the observation with the missing value. Imputer then decides the appropriate value to replace for the absent value STICK. We can use the standard scaler to scale values as values are having different ranges. The Standard Scaling will reduce the skew impact on algorithms like the linear regressions, Random Forest, and decision Tree.

**Feature Selection**

We can use SelectKBest from Scikit learn to find out P-Scores for all features. According to the p-scores, we can get rid of some of the features that do not have a major impact on the target variable.

**Modeling**

Before modeling, we will split our dataset into training and test data sets. The column name for the PartsUsed target variable in the tree is Parts. Terminal nodes had the percentage of records needing a part (Class equal to Parts) ranging from a low of 1.5 percent in Terminal Node 2 to 50.5 percent in Terminal Node 8. 50.5 percent fell far short of the minimum value required in the business objectives, even though the 50.5 percent Parts rate represented a lift of more than two over the baseline rate of 23.2 percent. This tree and the hundreds of others were clearly insufficient to achieve the goals of the model. That is why we can use Stratified splitting based on the target variable to split the dataset as there are fewer records of the dataset as compared to the 600 keyword and phrases. We can build several prediction models for this problem. We can start with linear regression. Then we can train other models like decision tree classifier, Random Forest classifier, K-Neighbors classifier, and XGB classifiers, etc. While preparing the model, we can use k-fold cross-validation as the dataset is small. For the cross-validation split, we can use a Stratified split to remove bias due to one type of observation.

Diagram

Description automatically generated

**Model interpretation**

We will compare all trained models for accuracy as well as the confusion matrix by predicting values for the test dataset. As previously mentioned, the model with the small Type II error we need to select. We can finalize a model with high accuracy and small Type II error as our final model for deployment. The comparison of the ROC curve can help us deciding the best model selection.

**Model Deployment**

For all data preparation, we will use the sklearn pipeline. The pipelines will help in applying the same operations on unseen data before applying model prediction. Because this model is used as a medical application, we must observe the model results frequently. We also need to retrain the model regularly as we are starting with a small dataset. It will also help us in accommodating characteristic changes with time.

**Conclusions**

As the hardware fleet scales and the complexity in the hardware repairs increases, rule-based diagnosis can be lagging for the new failure modes. In this case study, we present a prediction that bridges the gap by predicting the proper repair actions based on the human repair behaviors observed. Using decision trees allows the modeling to proceed quickly because trees handle wide data so easily and efficiently. adjusting learning parameters for the trees produced thousands of trees and interesting terminal nodes. In addition to providing diagnoses to undiagnosed repair tickets, this framework can also correct misdiagnosed repair tickets when the predicted repair is different from the rule-based diagnosis but has very high confidence. This will ensure cost saving to the company.

**References**

<https://hbr.org/1964/07/decision-trees-for-decision-making>

<https://towardsdatascience.com/predict-it-support-tickets-with-machine-learning-and-nlp-a87ee1cb66fc>

<https://www.sciencedirect.com/science/article/pii/S1319157819300515>

<https://machinelearningmastery.com/linear-regression-for-machine-learning/>

<https://towardsdatascience.com/understanding-random-forest-58381e0602d2>

https://www.kaggle.com/general/206853

Chapters 12 ("Applied Predictive Analytics") and 13 ("Help Desk Case Study") of *Applied Predictive Analytics: Principles and Techniques for the Professional Data Analyst*