Machine Learning Engineer Nanodegree

Capstone Proposal

Daniel Kelly dan_kelly@telus.net

Predicting Budget Cost Codes Based on Purchase Order Information

Domain Background

I currently work for a Construction Management company. One critical function of a Construction Management company is creating and tracking the budget of a construction project during the entire lifecycle of the construction project. During the construction of the project, the budget information may be used by developers/clients to track the performance of the architect, construction management team, and subcontractors. It can also be used to identify when a project may go over budget and require additional financing from their lender.

By breaking down the project budget into detailed sections like concrete, appliances, windows, etc. you can identify potential performance issues with individual subcontractors, products, or design elements and take steps to correct or mitigate them moving forward.

Additionally, when a project is complete you can then use the information in the finished budget to estimate the costs of new projects. Again, by breaking the budget into more granular sections, you can get more detailed information on what the costs of new projects are likely to be and how changes to the design or materials may affect that cost.

As a result of these use cases, it is important then to have accurate cost data and ensure that all costs are allocated to their respective budget sections. To do this, you use numerical Cost Codes that are unique and mutually exclusive. One numerical cost code is defined for each line in the budget. Every cost that arises during the construction of a project, from concrete orders to safety equipment, is allocated to a cost code within the project budget.

For consistency, an organization will define a master list of every possible cost code that can be used in a budget, then a project will use only a subset of these codes that are relevant to the type project.

The projects that my company manages range in budget from between $1,000,000 and $200,000,000 with between 100 and 900 cost codes depending on their size and scope.

Problem Statement

Currently, Project Coordinators on the construction site must review and assign a cost code to each item on a purchase order so it can be accounted for in the correct location of the construction project's budget. The Project Manager then reviews and confirms or modifies the correct cost code is selected for an item before committing it to the budget.

This is a time consuming and therefore expensive process for both the Project Coordinator and Project Manager that must be completed manually at least once per month for every purchase order item before the supplier of the purchase order can be paid and the costs of the project can be entered to the budget. A completed project will have between 1,000 and 10,000 purchase order items that will have needed to be manually coded.

This process cannot feasibly be accomplished by a one-to-one mapping of products to cost codes because products are continuously added or changed, different projects may use different vendors, vendors are continuously being added, one product may be associated with different cost codes depending on how it is used, and many other factors.

I propose that one way to reduce the amount of time and the expense of choosing the correct cost code for an cost is to implement a predictive model that will use the data that is present on a purchase order (The vendor name, product description, cost, etc.) to predict what the associated cost code should be for each item on a purchase order.

Using this prediction, I could present the end-user with a top 5 suggestions for the cost code that a purchases order item should belong to, therefore, reducing the amount of they need to spend looking for the correct code and increasing the accuracy of their cost coding.

Datasets and Inputs

The dataset (https://github.com/Daniel-M-Kelly/Udacity-MLND-Project/blob/master/raw_data/PO_Dataset.csv) that I will use for this project is a CSV list of purchase order items that I have exported from the SQL server database of my company's ERP system and obtained permission to use. This file contains over 39,000 examples of purchase order items, their accompanying information and their corresponding cost codes. These purchase order items have been previously entered into the company's ERP system and had their cost codes selected manually over the course of over 5 years and several construction projects. This labelled data will be split up and used as the training, validation, and test sets for the model.

As previously mentioned the raw export contains over 39,000 records, however through some data exploration, documented in this workbook (https://github.com/Daniel-M-Kelly/Udacity-MLND-Project/blob/master/PO_Dataset_Exploration.ipynb) , I found that there were some records that would not be helpful for achieving an accurate model. For example, there were some items with negative cost amounts which would correspond with credits that we had received from a supplier for items. These records are not relevant to the purchase of new items.

The clean dataset (https://github.com/Daniel-M-Kelly/Udacity-MLND-Project/blob/master/clean_data/Clean_PO_Dataset.csv) that I created as a result of the data exploration and cleaning resulted in a file containing approximately 28,000 records.

Solution Statement

I believe a solution to the problem of having employees spending time manually is to use a predictive model that takes input from the purchase order item such as vendor name, product name, quantity ordered, unit cost, and cost then outputs the top 5 suggested cost codes for the item. This process would happen for each purchase order item entered on a project and the goal would be to have a model that is accurate enough where employees can save time by selecting suggested cost codes rather than searching through an entire list of codes available.

Furthermore, a longer-term goal would be to have this model reach a point where the accuracy of its predictions is high enough where the model can be used to autonomously code costs associated with a project, without input from an employee. I envision this working in steps, where this project with the model making suggestions that the user selects is the first step. The second step would be where a higher accuracy model predicts and selects the most likely cost code and the employee confirms or changes the code as required. The final step would be where the model predicts and codes the cost directly into the budget and the employee would only be required to make a change if they noticed a problem or possibly if there is a low confidence prediction.

Benchmark Model

Currently, the processes for selecting cost codes for a purchase order items is entirely manual. We do not have statistics for how accurate the initial cost coding is, or how often Project Managers change cost codes when they are reviewing them. Additionally, we also have no documented data on relationships between the information on a purchase order and the cost codes. Therefore, without any additional data on what the correlation is between a cost code and the information in the purchase order, I believe that the most relevant benchmark model would be to use a model that always predicts the most commonly used cost code. After doing data exploration and clean up, the most commonly used cost code was used 4,157 times out of a total of 28,446 records so a model that always predicted this cost code would have an accuracy of almost 15%.

Evaluation Metrics

This is a supervised classification problem where we are trying to predict what cost code a purchase order item belongs to. I believe that the best evaluation for this metric is an F1-Score. I based this decision on my findings in the data exploration workbook.

The conclusion of my findings is that the dataset is very unbalanced. The most used cost code appears 4,157 times, whereas the mean usage of a cost code is 97 times with a median of 5.5. This means that there are a small number of codes that are used a large number of times, but most codes are used relatively rarely.

This imbalance makes accuracy poor measure of performance because, as discussed in the benchmark model section, the model can achieve an accuracy of almost %20 by always predicting the most common cost code despite being an obviously poor model. F1-score is a better metric than accuracy.
The formula for f1-score is:

Source: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

(https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9)

By using the f1-score we get a balance between precision and recall that better reflects the performance of the model when compared to accuracy.

Fbeta-score is another metric that could be useful, however, this metric is used to weight either precision or recall higher than the other. This would be used if one metric was more important than the other. Eg. Recall is more important if the cost of a false negative is higher than the cost of a false positive. In the case of this model, since we are suggesting cost codes to an end-user, the cost of a false positive is the same as a false negative.

As a result, I will use F1-score as the main metric to evaluate the model's performance but will also use recall and precision individually for reference.

Project Design

The project workflow will be:

1. **Continued data exploration** I have already done some data exploration while creating this proposal, I still need to look more in-depth into the description text field and perform any cleaning required.

2. **Decide how to handle text and numeric features** I think the most difficult challenge in the project will be how to handle the description field and combine that with the numerical data to make an accurate prediction. Two options are: To Convert the text to vectors using a process like bag-of-words or word embeddings and then include them in the input of a single model. Or creating two different models, one that evaluates the description and one that uses the other fields, then combine their resulting predictions.

3. **Create a baseline model** To start with I will use a relatively simple to understand Random Forest model which are generally known to give good performance for supervised classification problems like this one.

4. **Test additional models** Once I have a baseline model and know how it performs, I can test several other models to see if there is a better fit for the data with higher performance. Models I will try using are Adaboost, a Neural Network, and SVM.

5. **Tune Hyper-Parameters** Lastly, using the best model, I will tune the hyper-parameters of the model to continue to achieve the highest performance that I can get.