Central European University: MSc Business Analytics
Capstone Public Project Summary
Nadine Isabel Levin
IBM: Prediction Binary Classifier Model for UNSPSC Codes

OVERVIEW

For my Capstone project, I worked with IBM on their procurement analytics team, PAaaS (Procurement Analytics as a Service) to create binary classifier models in Python within Jupyter notebooks. The goal of my project is to create a model using a binary target variable if the raw procurement data provided by a new onboarding client will be able to be mapped to the existing PAaaS spend enrichment process. This process takes raw procurement data from a third-party client and categorizes the data rows using different algorithms to determine if the data can be mapped. PAaaS leverages corpuses for transactional, vendor, and master columns to easily map procurement data to universal classification using UNSPSC, United Nations Standard Products and Services Code, which is a global classification system of products and services. I created eight different binary classifying models and determined XGBoost to be the most accurate model at predicting what overall percentage of the raw data from a new client can be classified to the current mapping model.

PROJECT SUMMARY

To begin the data engineering of my project, I first started with a flat file as shared by a new onboarding client to the PAaaS product. This client did not have any of their procurement data columns added to the three corpuses, so it was an ideal dataset to test out a prediction model for what percentage of their data could actually be classified. The customer data was later mapped to the spend enrichment process, and I was able to use this separate dataset to create the target variable if the individual row was mapped. I first gathered my variables through the current manual classification process, and then joined my dataset to this separate data file which determined if the particular classification was mapped to the IBM corpuses. This created my binary target flag variable with a 0 for unmapped and 1 for mapped. I gathered 12 variables throughout the data engineering process to create my final data table fitted for modeling.

Then I created a series of models ranging in complexity to predict if the data row could be mapped to the PAaaS classification process. Each model had the target variable of a mapped client classification to the UNSPSC codes, and used inputs from the data engineering extractions. I broke the data into two different sets, test (20%) and training (80%), to train the model on a set of data while later testing the results on a smaller subsection of the data. This ensured that the model performed well against the real data rather than simply the training set. I discovered my target variable was severely imbalanced (93% at matched compared to 7% unmatched), and I needed to balance my training set so the model would

be trained on an equal dataset. I decided to balance the model using under-sampling, and reduced the number of total rows, but still validated using an imbalanced test set.

I created eight different models first testing a simple ordinary least squares (OLS) regression, then a logistic regression model, a horse race with more complex models, and finally creating a XGBoost model to investigate performance with a higher achieving machine learning algorithm. Within the horse race, I compared the logit model with a support vector machine, decision tree, Random Forest, Naive Bayes, and K-Nearest Neighbor all through the sklearn package. The best performing model was the Random Forest with a 76.7% accuracy, 79% precision, and 98.9% recall. Finally, the XGBoost model outperformed all other binary classifier models with an accuracy of 85.7%; almost 6% higher accuracy than the Random Forest model. I fitted this model with hyperparameters to further improve accuracy by .5%.

However, XGBoost is a fairly intensive model in terms of runtime with even the unfitted model requiring an estimated 70x longer runtime than Random Forest. It is also much more complex model to interpret due to the consecutive tree building and pruning. A gradient boosting algorithm like XGBoost grows trees sequentially using information from the previous tree to grow a better tree next time. XGBoost takes the residuals from the previous prediction and fits a model on the residuals instead of the original target variable which continuously improves the tree. Overall, I recommend that IBM leverage the XGBoost model for total accuracy (85.7%), recall (83%), and precision (94%).

Benefits of Project

There was a business need to better understand what percentage of a new client's data could be mapped to the existing infrastructure without manually running the data through the entire data engineering process. This project could help PAaaS provide a better onboarding experience by sharing the estimated percentage of how the new client's data could actually be mapped within the procurement product. During the data cleaning and normalization process, sometimes IBM found that a new client's data was so unique compared to the current classification process that the amount of a client's data which could actually be used in the procurement analytics was fairly low without manual additions to the corpuses. By better understanding this limitation upfront, IBM could help set customer expectations of how long the integration would take and how useful the immediate data would be for them.

Key Outcomes

In conclusion, I developed eight models in total which are able to predict with up to 85.7% accuracy whether a procurement analytics data row from a new onboarding client to the PAaaS program is able to be successfully mapped in the spend enrichment process. I gathered the input variables for my models

through selecting key variables during the PAaaS ETL (extract transform load) process in their existing spend enrichment process. I broke out the data into a train and test set with equally weighted target variables while investigating multi-collinearity and principal component analysis. I built models ranging in complexity from a simple ordinary least squares model to a Random Forest model before finding Random Forest as my best performer in terms of accuracy, precision, and recall. From there, I fitted my model to try to improve the performance, but saw the best performance was from including the custom variables for the test client. I then created a complex XGBoost model with custom parameters but with significant increase in runtime. I thoroughly compared the Random Forest and XGBoost models to fine tune both and ensure stringent evaluation. In the end, I recommend IBM use the XGBoost model for prediction with the full list of variables per client for the highest accuracy, recall, and precision.

KEY LEARNINGS

My learnings and takeaways from this project are below:

1. **Deeper Understanding of Python:** As I have only recently learned Python, this project gave me a deep understanding of how to use Python for data engineering, cleaning, and model creation through important packages like pandas and sklearn.
2. **Procurement Analytics:** Previously I was not familiar with the industry of procurement analytics so this project gave me valuable insights into this data heavy world. I became accustomed to analyzing industry trends like classification depth and data intake methods.
3. **Real ETL Pipeline:** This was my first beginning-to-end data science project working with real life ETL processes. The script I used to extract my independent variables was roughly 30,000 lines long with a 12,000-line helper function. I became familiar with the complexities of a full data pipeline used in production.
4. **Learning Internal IBM Tools:** I used the virtual cloud environment IBM Watson to develop my models and IBM Cognos to help visualize the procurement data. I had not used these tools previously and became familiar with their capabilities.