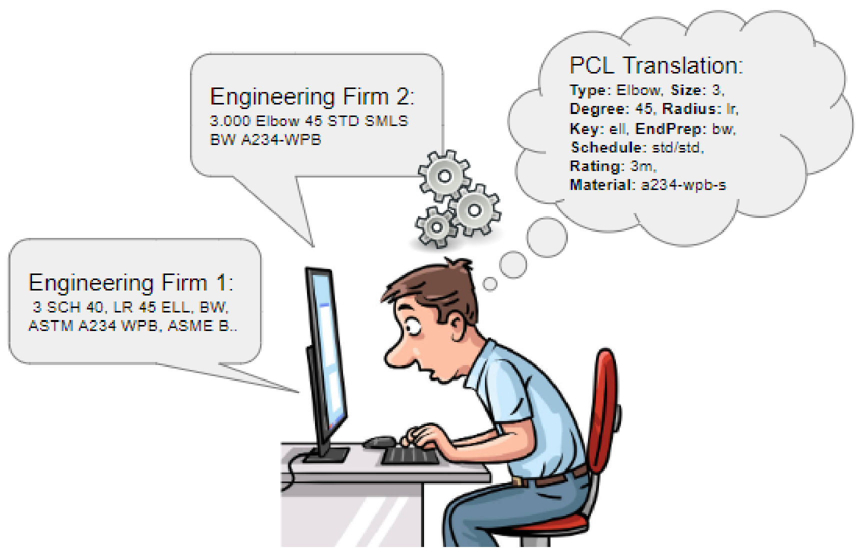
Addition of XGBoost to Translation Pipeline Overview

By: Maxym Wojnowskyj

This task was worked on with the Data Science team at the Fabrication facility in Nisku AB. Through my request and extra effort I was able to schedule meetings with my Team lead and the Data science team lead to take on this project for the last month of my work term. My motivation for taking on this extra work was to learn a bit about PCL's data science operations at the facility and get some private industry machine learning experience. This is a high level summary overview of the work I completed.

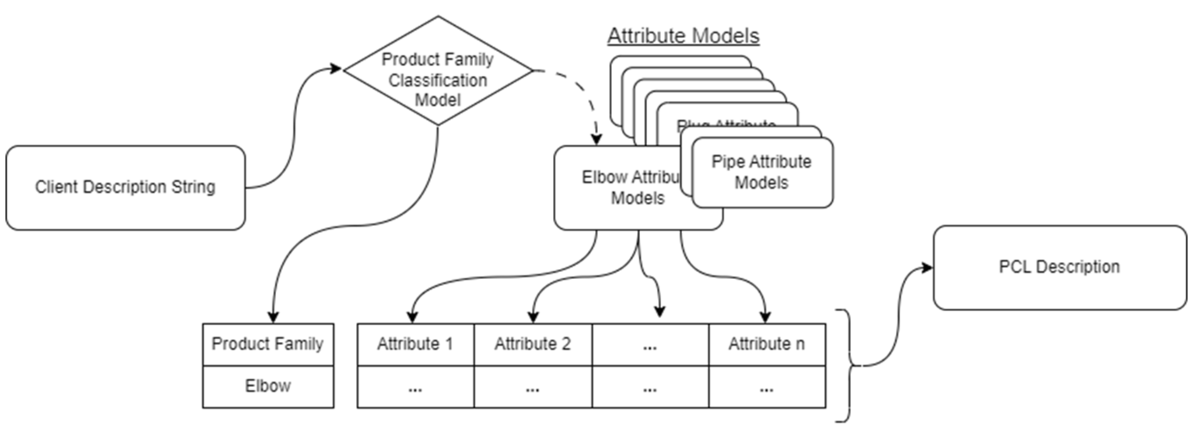
Project and Problem Description:

PCL's fabrication facility gets orders for components from all around the world from different engineering firms. Issue is that there is no standard naming method for components:



Data science team has a reported human Error rate on these translations of 2%. An error in translation has a large magnitude of lost time and extra work. One wrong translation from a PCL Engineer can result in many extra weeks of overtime for hundreds-thousands of workers on site.

To combat this issue, the PCL data science team has developed a Prediction model containing a pipeline of ML models. The prediction model predicts each attribute (ex: Size, Degree) of a component independently. The model trains on each attribute and stores the best performing model from the pipeline on said attribute. In production an inputted engineering component has its attributes predicted by its stored best ML prediction model.



My task was to add XGBoost to this pipeline of ML models.

Personal Task Description:

Before addition to the pipeline on the prediction models github. Testing had to be done in a Jupyter Notebook. The preprocessing code from the github had to be added into the notebook for testing with the proper cleaned data. There were several issues that I found when attempting to use the data for training an xgboost model. The most significant was a class imbalance issue.

I implemented 3 different solutions to handle the class imbalance issue.
- Adding Oversampling to the data preprocessing functions
- Adding Undersampling to the data preprocessing functions
- Adding in a custom built cross validation function from scratch

These 3 solutions were added in as a user enableable feature to test with and select which solution provides the best result with the current training and testing data.

Along with an XGBoost predictor, I added in Bayesian optimization for hyperparameter search using [hyperopt](). Due to the high amount of hyperparameters, grid search nor random search were viable methods to find the optimal set of hyperparameter values within a reasonable amount of time. Therefore I chose to use bayesian optimization to smartly find the best set within a reasonable training time.

I also created a 15 slide presentation where I described the model, my task, the problems I ran into, how I solved the issues, my results, and recommendations for future work to be done with the model and my task specifically. There were many optimizations for training time that I did not have time to implement like: creating custom distributions for each of the hyperparameters for bayesian search (uniform distributions were used) and setting the maximum amount of interactions for bayesian search based on the dataset size and complexity.

These are some of the interim results I received from testing my different XGBoost implementations:

Results:

Default results with no solutions enabled for the class imbalance issue (only 24 are able to run):

Feature	XGBoost Test Accuracy	Best Test Accuracy	Difference in Predictions
Elbow Size	99.00%	97.74%	5
Elbow Degree	100.00%	100.00%	0
Elbow Radius	100.00%	99.47%	2
Elbow End Prep	99.55%	100.00%	-2
Elbow Rating	99.00%	97.00%	2
Pipe Construction	99.30%	99.79%	-7
Pipe End Prep	100.00%	100.00%	0
Tee Rating	100.00%	98.96%	1
Coupling Key	100.00%	97.70%	2
Coupling Rating	100.00%	100.00%	0
Coupling End Prep	97.73%	100.00%	-2
Olet Key	98.95%	98.33%	3
Olet Rating	98.87%	100.00%	-3
Olet End Prep	94.12%	100.00%	-4
Nipple End Prep	97.69%	98.08%	-1
Nipple Schedule	99.23%	98.85%	1
Cap End Prep	98.04%	100.00%	-2
Plug Size	100.00%	77.27%	5
Plug Key	100.00%	100.00%	0
Plug Head	92.00%	96.00%	-1
Plug End Prep	100.00%	100.00%	0
Swage End Prep	96.64%	97.90%	-3
Union Size	100.00%	58.33%	5
All Type	99.91%	99.94%	-2

24/71 models successfully ran. 9/24 saw improvement. 14/24 saw improvement or equal performance.

With Oversampling enabled:

Feature	XGBoost Test Accuracy	Best Test Accuracy	Difference in Prediction
Elbow Size	99.00%	97.74%	5
Elbow Degree	100.00%	100.00%	0
Elbow Radius	99.73%	99.47%	1
Elbow Key	100.00%	100.00%	0
Elbow End Prep	100.00%	100.00%	0
Elbow Rating	99.00%	97.00%	2
Pipe Key	100.00%	99.93%	1
Flange Size	97.50%	96.33%	7
Flange Key	100.00%	99.84%	1
Reducer Sub Type	100.00%	100.00%	0
Reducer Key	100.00%	100.00%	0
Reducer End Prep	100.00%	100.00%	0
Tee Key	100.00%	99.74%	1
Tee Rating	100.00%	98.96%	1
Coupling Key	100.00%	97.70%	2
Coupling Rating	100.00%	100.00%	0
Olet Key	99.58%	98.33%	6
Nipple Key	100.00%	100.00%	0
Nipple Construction	100.00%	100.00%	0
Cap Size	96.10%	93.51%	2
Cap Key	100.00%	100.00%	0
Cap End Prep	100.00%	100.00%	0
Plug Size	100.00%	77.27%	5
Plug Head	100.00%	96.00%	1
Plug End Prep	100.00%	100.00%	0
Swage Key	100.00%	100.00%	0
Swage End Prep	97.90%	97.90%	0
Union Key	100.00%	100.00%	0
Union Rating	100.00%	100.00%	0

65/71 models successfully ran. 13/65 saw improvement. 16/65 equal performance. 29/65 saw better or equal performance.

With Undersampling enabled:

Feature	XGBoost Test Accuracy	Best Test Accuracy	Difference in Predictions
Elbow Size	97.99%	97.74%	1
Elbow Key	100.00%	100.00%	0
Elbow Rating	97.00%	97.00%	0
Pipe Key	100.00%	99.93%	1
Flange Size	97.33%	96.33%	6
Flange Rating	99.53%	99.53%	0
Flange Key	99.84%	99.84%	0
Reducer Sub Type	100.00%	100.00%	0
Reducer Key	100.00%	100.00%	0
Reducer End Prep	100.00%	100.00%	0
Tee Key	99.74%	99.74%	0
Tee Rating	100.00%	98.96%	1
Coupling Key	100.00%	97.70%	2
Coupling Rating	100.00%	100.00%	0
Nipple Key	100.00%	100.00%	0
Nipple Construction	100.00%	100.00%	0
Cap Key	100.00%	100.00%	0
Plug Head	100.00%	96.00%	1
Union Key	100.00%	100.00%	0
Union Rating	100.00%	100.00%	0

43/71 models successfully ran. 6/43 saw improvement. 14/43 equal performance. 20/43 saw better or equal performance.

I am very happy and impressed with the results, given that I was able to receive an equal to best or improved accuracy score with XGBoost in about 40% of the features that needed a prediction. Since the results met and exceeded the Data science team's minimum threshold for improvement, I also worked with the team to implement the XGBoost code into their pipeline along with the bayesian optimization functions. The bayesian optimization functions were also built in a way so that the data science team could add bayesian search for the other models in their pipeline to try and further improve their results.