Damage Rate Forecasting

Capstone Proposal

Dane Anderson

January 14, 2019

Proposal

Domain Background

Businesses work best when there's a plan in place. Without a plan, it can incur costs that are unnecessary and possibly avoidable. For example, if a company lends out assets to other businesses, these businesses are required to return these assets back to their original owners. When it comes to the biggest pieces of the financial analysis for a company that pools their assets, the rate at which their assets come back broken, the cost to make new ones, and the rate in which they are being sent and returned are of the utmost importance. Once the finance team determines this information, they can begin their own proposals to plan the allocation of money to these tasks for the year. In the supply chain industry, these assets don't always make their way back to their original owners in the best conditions. Sometimes they are broken and need repairing. If you are part of a business where all they do is pool assets across the supply chain around the world, a major part of your cost analysis is how much your assets get damaged and need to be repaired.

Problem Statement

(All Numbers mentioned in this section are fictional and are provided purely for demonstration purposes, they hold no value and should only be regarded as hypothetical data for the purposes of explaining the importance)

Mathematically, the damage rate equation is very simple. It is as follows: assets damaged / all assets returned. The full equation used to plan expected costs regarding repairs per asset returned is: (damaged/damaged+working)*repaircost. This simple number tells everyone that for every 100 assets that are returned, X amount are needing to be repaired. There is also a consistent cost to repair these assets, for the sake of ambiguity and for this capstone project, I will put that consistent price to be $4. In turn, if the damage rate for the entire business for this week is .4, then you multiply .4 by 4, which ends up being $1.60. We now know that if we allocate $1.60 towards every expected asset that is returned to us for the month, we should have a strong idea how much money we should set aside for repairs. These repair costs include materials, labor, and shipping. All three are fairly easily obtained if there is a proper amount of time coordinated in advance. Otherwise, this process can be very costly, mostly because labor last minute is a premium, shipping large amounts of assets last minute is

a premium, and sometimes materials are in deficit which, due to supply and demand elasticity, price goes up. A cost to repair an asset goes from $4 if planned properly and goes up to $8 if not. If a network has 30 million assets returning in a given month and a damage rate of .4 is planned, but a damage rate of .6 (12 million) actually happens, the company now has to spend .2 (6 million) of the damage rate at $8 per asset. $48M is paid out with a .4 rate and $48M is paid from the .2 rate due to the premium. The final 1/3rd of the repairs cost the same as the first 2/3rd. Having this kind of spread is akin to flipping a coin every month based on the previous months. If we have an accurate forecast, this can save a considerable amount of money for the organization that would allow them to place it in other places like employee bonuses.

Datasets & Inputs

Disclaimer: The dataset is scrubbed from proprietary real world data from my company. I have been given consent to replicate the data and remove any identifiable information by leadership.

- Damage Rate Actuals
    - Having the historical data for the dataset you are forecasting is the most important dataset because the model uses this to find patterns etc. to generate the forecast
    - The structure is as follows:
        - Calendar Day
            - First day of the business week
        - Bucket
            - Grouping of the types of customers returning the asset
        - Damage Rate
            - In decimal form the number of assets return damaged out of every 100.
- Damage Rate Actuals Network
    - Same as damage rate actuals but it's aggregated from the top to show the rate for the entire business.
- Inflow Actuals
    - Inflows are the weekly numbers that represent the number of assets returned to the business
    - This is a secondary metric used to add complexity to the model and a secondary dataset for developing a more robust model
    - The structure is as follows:
        - Calendar Day
            - Sunday each week
        - Bucket
            - Grouping of the types of customers returning the asset

        - Inflows
            - The number of assets returning to the business
- Inflow Forecast
    - Without the forecast of the second dataset used in the model, it couldn't make the forecast with inflow data as a secondary dataset.
    - As an added challenge to the project, the inflow forecast is monthly instead of weekly like all the other datasets
    - The Structure is as follows:
        - FY
            - The year
        - FM
            - The month
        - Bucket
            - Grouping of the types of customers returning the asset
        - Fcst
            - The number value of expected returns to the business

Solution Statement

In order to get an accurate plan for damage rate, you must have a forecast. There are many ways to get there though. You could always do an ARIMA forecast model but those are univariate. Multivariate forecast models are proven to have consistent positive results, especially when using a machine learning approach. The best regression technique so far is the XGBoost model. Extreme Gradient Boosting Regression is good because it is not sensitive to noise or outliers and can pick up patterns and make a great output with a small amount of data engineering or hyperparameter tuning.

Benchmark Model

Since this is a real-world business issue, there isn't going to be a model that I will compare success to. Instead, I will compare accuracy and success of the model by the historical data using the train-test-split function of scikit learn. This can properly evaluate how well the model is training and testing. Further, I will use an older extract of the data where the forecast can compare to what actually happened in reality at the end, simply for demonstration purposes for the project.

Evaluation Metrics

I will use error bias and accuracy as an evaluation metric. This can tell us how far off from reality the forecast was. This should be a satisfactory measurement to determine model success. In the business, if the model reaches a 95% accuracy, then it is considered satisfactory, but of course, it's always suggested to strive to perfection. Another possible way to calculate an evaluation metric is how far the output deviates from the mean per time period. If a non-

programmer, non-statistician attempted to make a forecast, they would just create some sort of rolling 4 week mean and apply it to each further week. We will compare these.

Project Design

When it comes to determining what strategies to employ, the solution is rather apparent for those who have used regression algorithms before. For other projects done in my field of work, there have been many various strategies attempted to create a forecast. Some of those attempts include using MatLab, a Recurrent Neural Network, or even some standard regression models like Croston-Fourier, AVS-Graves, or even a Moving Average. MatLab is too expensive to use in an enterprise solution to just pick up without corporate approval, recurrent neural networks are too sensitive to seasonality, outliers and require an enhanced form of stationarity which in turn removes a lot of the patterns from the datasets, thus causing the output of an RNN to be flat and only deviate from this mean by decimals. Standard regression models are good for quick and small tasks that don't require a lot of variation between each time period and sometimes can skew in a way that is not wanted. After many months of testing different models and solutions, the best option ended up being DMLC's XGBoost regression package. It's easy to use out of the box, we can use grid search to easily obtain optimal hyperparameters and create quick regression forecasts for nearly anything.

The flow for the script is as follows: