Anomaly Detection in Time Series: Predictive Machine Failures

Luke Muther and Professor Jon Chun

Abstract

The goal of this project is to take a variety of machine data from a Schwann food delivery company factory and create a classifier model that will predict if a given machine will fail in the next 12 hours. This involved cleaning and augmenting the dataset, training the model, and fine-tuning hyperparameters. Through this I was able to create a model that was able to successfully predict 98% of the machine failures as well as only had 6 falsely predicted failures for every 100 successfully predicted failures.

After fine-tuning the model, I created a simulation with the same hyperparameters I found to be the best to see how long, or how much training data, it would take for the model to be successful.

Methodology

For this project, I decided to use a combination of four CSV files from the factory dataset that had 100 machines. Included was telemetry data, which was the voltage, pressure, rotation, and vibration of the machine every hour of every day for a year. Next was data on when and what kind of errors occurred. Then, data on when maintenance occurred and on what component of the machine. Finally, a data file on when machines failed and what component failed.

I used Pandas, a python data analysis library, within Google Colaboratory notebooks to combine all 4 data sets into one DataFrame. From this data frame I could find all the failures and then go back and take statistics of the previous data. With these statistics along with randomly sampled normal machine operational statistics, I was able to create a labelled dataset to feed into my classifier model.

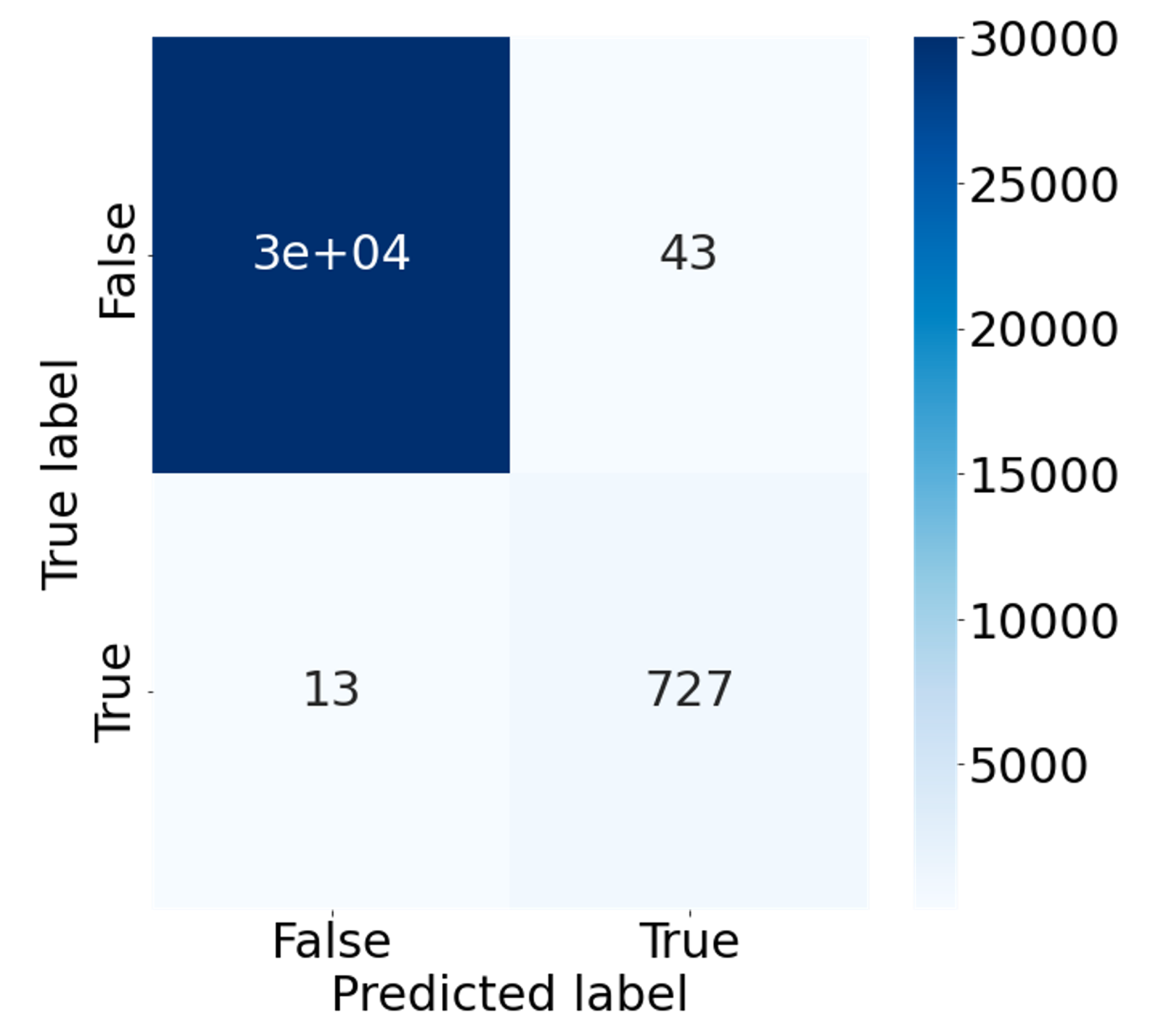
pandas dmlc XGBoost + scikit learn

With this labelled dataset, I was ready to train a model. I chose to use a classifier model from the XGBoost library, which provides a Scikit-learn compatible gradient boosting framework. This uses decision trees and adds new trees to correct for the mistakes of the previous one. I chose to fine-tune three hyperparameters. They are the depth of the trees, the number of trees, and the learning rate of the model. To do this fine-tuning, I used SK-learn's grid search class.

Results

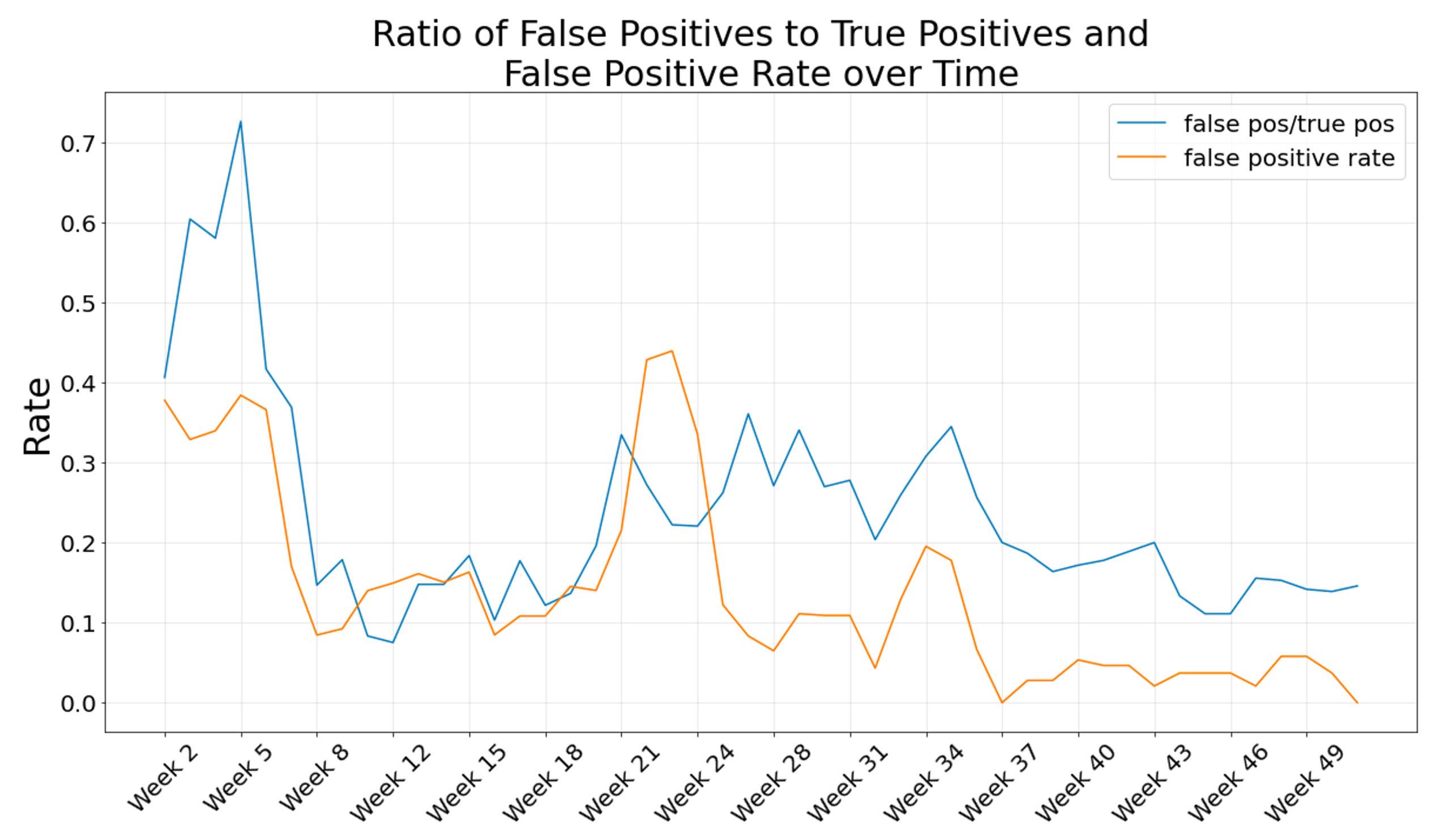
Through creating the model, I found that the data I had were good predictors of whether a machine would fail or not.

Using grid search I found that the optimal depth of the trees was four, the number of trees was 300, and the learning rate was 0.08.



Above shows a confusion matrix with the results of the final mode. As you can see, normal functioning machinery is the norm, this is where the True Label is False. This is what makes it anomaly detection. However, looking at the other pieces of the matrix, that false negatives are rare and false positives are about four times as common, but still unlikely.

The simulation showed that the model is able to learn quite successfully on low amounts of data. It only took eight weeks for the rate of false positives and the ratio of false positives to true positives to steady out.



The graph above shows both aforementioned metrics as they change over throughout the simulation. There is a slight downward trend over the year, but I think it is more notable that there is a steep downward trend in the beginning for both metrics. At week 8 the model was trained on 1,755 data points. Seeing as the original fine-tuned model was trained on more than 20,000 data points, it is quite impressive the model can be quite successful with so little training.

Conclusion

Creating and optimizing a gradient boosting classifier model showed that the data available was successful in terms of being able to predict machine failures twelve hours in the future. A real factory would be able to be able to implement this method and save costs to minimize surprise failures and unnecessary maintenance on its machines.

In addition, the simulation showed how fast this classifier model could be trained to be useful for a factory owner. Not only this, but also it would continue to get better as more and more data is captured, and the model is retrained.

Acknowledgements

Andrei Karpau, *Failure Prediction Training*, GitHub, 2020. https://github.com/andreikarpau/DataScienceRepo/blob/master/PredictiveMaintenance/FailurePredictionTraining.ipynb

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785

Deepti Chevvuri, *Predictive Maintenance Modelling Datasets*, GitHub, 2018. https://github.com/DeeptiChevvuri/Predictive-Maintenance-Modelling-Datasets

Huge thank you to Professor Jon Chun and Professor Tim Sullivan for guiding me through this project!