TabTransformer for Datacenter Equipment Failure Monitoring using SMOTE and Drift Detection

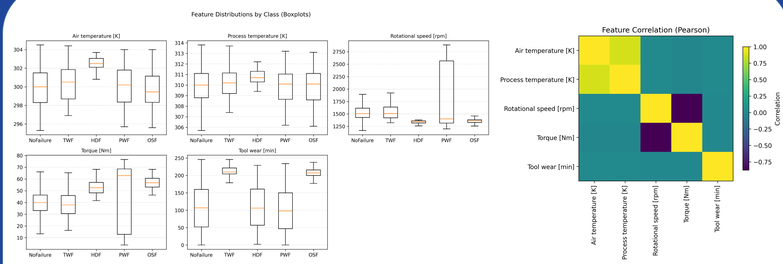
Brandon McCoy - Saint Louis University

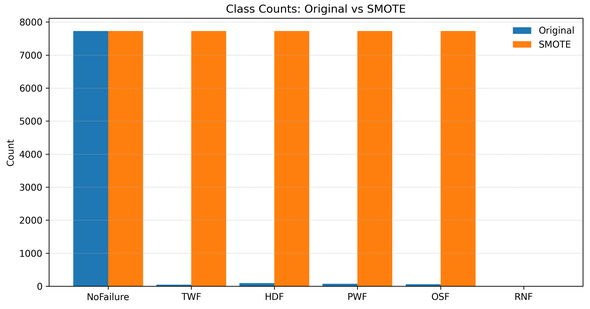
Abstract

Unplanned datacenter downtime can cause significant financial losses, operational disruption, and reputational damage. This project proposes a proactive machine failure prediction pipeline using tabular diagnostic data from industrial equipment. A TabTransformer architecture was implemented to capture feature dependencies through multi-head self-attention, combined with SMOTE to address severe class imbalance and TorchDrift for distribution shift monitoring. Models were evaluated on the AI4I predictive maintenance dataset with binary and multi-class failure targets. Results demonstrate strong binary failure detection accuracy and substantial improvements in minority class recall after SMOTE augmentation. Confusion matrix analysis highlights remaining challenges in distinguishing specific failure types, indicating a need for expanded datasets and continued learning pipelines. The method shows potential for enabling lightweight deployment for real-time monitoring and early intervention in large-scale datacenter environments.

Data and Class Imbalance Issue

For this project, I utilized the AI4I 2020 Predictive Maintenance dataset used for predicting datacenter equipment failures, available on the UCI Machine Learning Repository. But this dataset comes with a significant class imbalance issue, which lead to use of the Synthetic Minority Oversampling Technique (SMOTE).





Note:
RNF represents random failures, with no significant indications for failure type. Only 17 out of 10,000 observations. And therefore were removed from the dataset in preprocessing and the machine learning pipeline.

Methodology

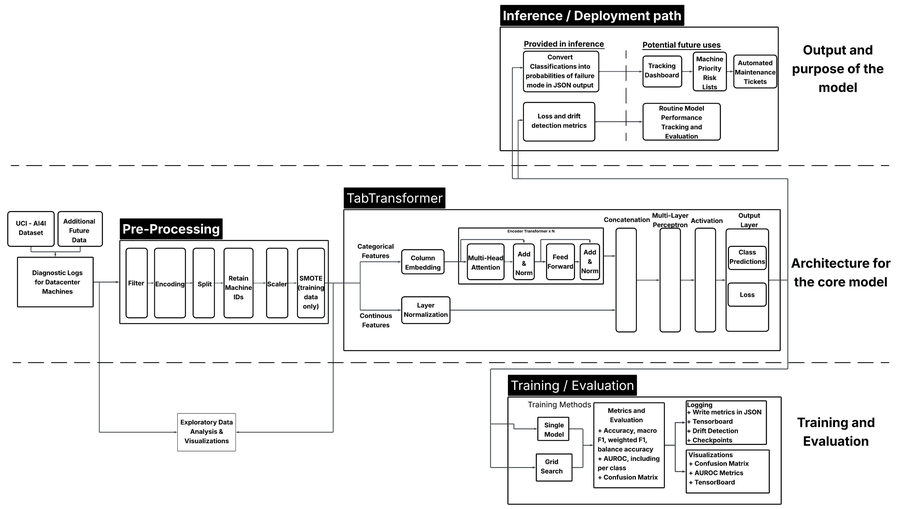
The machine-learning pipeline using tabular diagnostic data from datacenter equipment begins with data preprocessing, standardizing numerical features, and encoding categorical machine types. Due to failure events being rare relative to normal operating observations, the dataset exhibited a substantial class imbalance. SMOTE (Synthetic Minority Oversampling Technique) algorithm was applied to generate realistic synthetic examples of under-represented failure categories. SMOTE improves the density of minority classifications and supports the learning.

The model utilizes a TabTransformer architecture, which leverages multi-head self-attention to learn contextual relationships between tabular features Because machine behavior and sensor distributions can evolve over time, a drift detection module was incorporated to monitor statistical distribution shifts and performance decay during inference. Model performance was evaluated using macro-F1 score, recall, precision, accuracy, and AUC-ROC, with additional focus on confusion matrix analysis to verify improvements in failure type classification sensitivity. Metrics were benchmarked both before and after SMOTE augmentation to quantify its impact.

Problem Definition

Given the diagnostic data in a tabular format for devices in a datacenter, the objective is to create a transformer that utilizes attention to accurately track for the risk of devices failing. While also addressing issues of class imbalance due to low classification counts in comparison to the overall data. The classification should come in two forms, firstly a binary 'True' or 'False' for risk of machine failure, and secondly a multi-class estimate for the type of failure. This includes tool wear failure (TWF), heat dissipation failure (HDF), power failure (PWF), and overstrain failure (OSF). Additionally, there needs to be an output able to specify the machine ID that is being flagged with probability predicted of failure. Ideally the prediction model should be able to run on a lightweight device.

Architecture



Things of note and Possibilities for Future Developments

First and foremost, this project has been a great way to learn about how to deal with datasets that use continuous data that has a large amount of observations for 'normal' states (NoFailure for AI4I data) and a small amount of targeted events, such as machine failures for a datacenter with a large number of machines. The binary classifier of 'Failure' vs 'NoFailure' is reasonably achievable as it uses the sum of all the sub-categories of failures. But when getting granular with the failure types, using traditional class balancing technique such as weighting the class split for the train/text/validation datasets will not suffice.

This is where the Synthetic Minority Oversampling Technique (SMOTE) came in very helpful. As it does not simply create duplicate records for the underrepresented classes, but instead uses a nearest neighbor method of generating synthetic observations for the underrepresented multi-class values. This is then extrapolated out until the underrepresented failure types have equal representation between 'NoFailure' and each type of failure.

As for the model selection, while reading the literature I found most examples of working with this dataset utilize decision trees, SVM, XGBoost, and ensemble methods. With decision trees being the most effective, however often having an overfitting problem. And after reading 'Attentional is all you need' I wanted to see how a transformer utilizing attention faired in comparison. That is where I found the TabTransformer and used it as my baseline architecture. My model didn't perform as well as those models claimed, but I also didn't have extra datasets for training and didn't to explore continued learning and utilization of the drift detection for hyper tuning. Which are possible developments for the future and may be able to close the accuracy gap with less risk of overfitting.

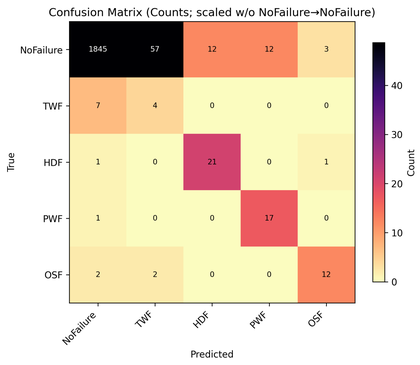
One of the concerns about using the SMOTE method is that real world changes in equipment or run-times may result in changing conditions in which the prediction effectiveness degrades, or original training data doesn't fully represent real scenarios. That is why drift detection was additionally important in exploring. For this iteration it allows to diagnosis for model performance, but in future developments of continued learning it could be utilize to tune the model.

Results

To the right we can see the confusion matrix of the checkpoint with highest macro-AUC score. I'm finding a high accuracy score in the binary classification of Machine Failure being true or false. With the recall score being lower than liked due to false positives.

However, this is a vast improvement from the pre-SMOTE model runs in which the model was over predicting false positives by the hundreds. With false positives being primarily an issue with Tool Wear Failures, I believe there is a need for more data to further train the model and to improve the recall score, but these failure flags can still be used to track potential failures before they happen.

As can be see in the classification report, the precision and recall is worse with those that had fewer support observations in the validation dataset. This is a place of opportunity for future developments to use more data and a continued learning integration to improve from this baseline model provided here.



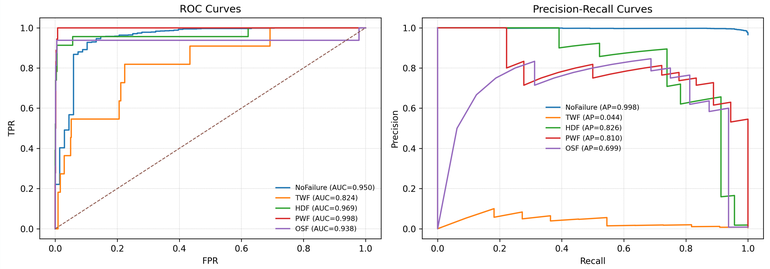

```
Test accuracy: 0.9509
Macro F1: 0.6613 | Weighted F1: 0.9635
Balanced accuracy: 0.7855

Classification report:
              precision    recall  f1-score   support

   NoFailure       0.99      0.96      0.97      1929
         TWF       0.06      0.36      0.11        11
         HDF       0.64      0.91      0.75        23
         PWF       0.59      0.94      0.72        18
         OSF       0.75      0.75      0.75        16

    accuracy                           0.95      1997
   macro avg       0.61      0.79      0.66      1997
weighted avg       0.98      0.95      0.96      1997
```

ROC & Precision-Recall (One-vs-Rest)



Conclusion

When working on a binary 'Failure' vs 'NoFailure' using the attention-based tab transformer it was able to get high accuracy scores with relatively low adjustments and hyperparameter tuning. But it became more difficult when trying to address recall accuracy on the multi-classifier for type of failure. This has to do greatly with the class imbalance issue I ran into. The SMOTE method was great for getting a reasonable accuracy score for a baseline method. And I was able to include TorchDrift to assist with further model evaluation and tracking. But I believe having more data and moving the model to integrate a continued learning enforcement will help further increase recall scores and accurate tracking types of failures and when machines start driving into possible error states.

References

Project GitHub: https://github.com/McCoyBrandon/AI_Capstone

X. Huang, A. Khetan, M. Cvitkovic and Z. S. Karnin, "TabTransformer: Tabular Data Modeling Using Contextual Embeddings," *arXiv preprint* arXiv:2012.06678, Dec. 2020.