Condition-Based Maintenance and Remaining Useful Life Analysis

Data 606: Capstone in Data Science
Dr. Ozgur Ozturk

Carol Kingori, Jacob Shaw, Neftalem Negussie

Introduction

As automation increases in the world of manufacturing, sensors provide the operational condition of the machine in an instant such that the working condition of the machine can be assessed online. The data produced from sensors assess machine performance at a given time and can be used as an indicator of machine failures. Failure analysis is especially crucial since it becomes more expensive to fix a machine as humans remove themselves from the factory floor as much as possible.

This study utilizes runtime sensor information for an ion etching machine to predict occurrence of various failures. Taking into consideration historical data of failures, it will predict when the next failure is likely to happen and provide early warning to operators so that preventive maintenance is done on the machine, thereby reducing the downtime. The early warning is then used to predict how long until the machine will fail.

Condition-Based Maintenance and Remaining Useful Life

As the cost to repair goes up, a maintenance strategy called Condition-Based Maintenance (CBM) has become increasingly popular (Zou, 2019). Compared to the traditional style of Time-Based Maintenance (TBM) which focuses on maintaining on a set schedule, CBM utilizes the aforementioned sensor data to analyze the status of machines to only repair when necessary. The utilization of CBM has proven to be more effective than TBM, especially as the cost of repairing a failure becomes more expensive given the complexity of current machines and less people being present at a given time to repair a machine. However, CBM requires finely tuned models since predicting a failure too early leads to unnecessary costs fixing something that isn't broken, but waiting too late and the machine requires major repairs. Some studies refer to conditional maintenance as predictive maintenance.

On top of forecasting that a failure may happen in the near future, sensor data can also be used to predict how long until the machine breaks down, also called the

Remaining Useful Life (RUL). RUL is the remaining time that equipment or machines have to operate optimally before their performance starts to decline (Huang et al., 2018). Our specific case of RUL is described as the Run-to-Failure Data approach where data points are analyzed as they come off the machine and compared against previous runtimes to see if the new points match data consistent with degradation of the machine (Baru, 2018).

Other Works

Studies have been done using the PHM 2018 Data Challenge dataset (Botatakis, 2018) to estimate the remaining useful life of equipment. Unlike other studies, this study uses different scenarios with various cutoffs for failure classification using different models. Similar to other studies, this study also does regression to estimate the remaining useful life of the machine.

Kang et al.(2021) estimate the remaining useful life of manufacturing equipment in a NASA turbo engine dataset. Their overall goal is to optimize predictive maintenance with the aim of significantly reducing maintenance cost by doing maintenance of equipment as needed. Since the dataset is highly correlated, they use PCA to select features. A linear regression is used to interpolate the data to estimate the health index. To determine the remaining useful life (RUL) a polynomial is fitted to the health index data and the failure point is determined by intercepting the fitted line with the cycle axis. Support vector regressor (SVR), random forest and multi-layer perceptron (MLP) are used to predict the generated labels. To optimize the parameters grid search cross-validation is applied. MLP has the best performance in predicting the RUL.

Wu et al.(2021) approached the PHM 2018 dataset with a two step approach. First, they used a Random Forest (RF) to determine the probability of failure of a machine for a certain failure type given the data, called the Degradation Warning (DW). Defining a cutoff point with probability is necessary since it is unreasonable to predict the RUL when a machine is operating normally. Once the RF model is triggered by returning a probability of failure over 0.5, the RUL analysis is triggered. Long Short-Term Memory (LSTM) neural networks, Gated Recurrent Unit (GRU) neural networks, and additional Fully-Connected layers (FCs) were used to predict the RUL. The GRU with FCs that ran once the DW was triggered yielded the best performance.

TV et al. (2018) used the PHM 2018 dataset to estimate the remaining useful life of the ion etching machine using a deep Recurrent Neural Network (RNN) model to learn and develop the metric for RUL for the various failures (3) on line. Since the time series data they had from the 20 machines provided in the dataset was large, they had down sampled the initial series in two steps first with a factor of 2 and then using the

Timeseries To Vector method (T2V) to down sample it further by a factor of 10 in effect making the series down sampled by a factor of 20 (2*10). The LSTM Multivariate Regression model with two layers was then trained on the down-sampled data with MSE as a loss function to compute the RUL. From the two modeling approaches they experimented with (LSTM-MR and LSTM-MR ensemble), they found out that performance of the model improved with the ensemble approach.

Understanding of the Dataset

The dataset is from the PHM 2018 Data Challenge. The dataset consists of files from 20 different tools, however, in this study we analyze only one tool, 01_M02.The dataset for tool 01_M02 consists of 3 files; Train, Time to Fault (TTF), and Fault. The size of the dataset is as follows:

Table 1: Dataset Description

Dataset	**Size**	**Dimensions**
01 M02 Train	1.79 GB	5110542 x 24
01 M02 Train - TTF	169.76 MB	5110542 x 4
01 M02 Train - Faults	6 KB	109 x 4

The Train dataset contains the sensor data from the Ion Etching Machine with the following features:

Table 2: Features in the Dataset

ID#	Parameter Name	Type	Description
S1	time	Numeric	time
S2	Tool	Categorical	tool id
S3	stage	Categorical	processing stage of wafer
S4	Lot	Categorical	wafer id
S5	runnum	Numeric	number of times tool has been run
S6	recipe	Categorical	describes tool settings used to process wafer
S7	recipe_step	Categorical	process step of a recipe
S8	IONGAUGEPRESSURE	Numeric (Sensor)	pressure reading for the main process chamber when under vacuum
S9	ETCHBEAMVOLTAGE	Numeric	voltage potential applied to the beam plate of the grid assembly
S10	ETCHBEAMCURRENT	Numeric	ion current impacting the beam grid determining the amount of ions accelerated through the grid assembly to the wafer
S11	ETCHSUPPRESSORVOLTAGE	Numeric	voltage potential applied to the suppressor plate of the grid assembly
S12	ETCHSUPPRESSORCURRENT	Numeric (Sensor)	ion current impacting the suppressor grid plate
S13	FLOWCOOLFLOWRATE	Numeric	rate of flow of helium through the flowcool circuit, controlled by mass flow controller
S14	FLOWCOOLPRESSURE	Numeric (Sensor)	resulting helium pressure in the flowcool circuit
S15	ETCHGASCHANNEL1READBACK	Numeric	rate of flow of argon into the source assembly in the vacuum chamber
S16	ETCHPBNGASREADBACK	Numeric	rate of flow of argon into the PBN assembly in the chamber
S17	FIXTURETILTANGLE	Numeric	wafer tilt angle setting
S18	ROTATIONSPEED	Numeric	wafer rotation speed setting
S19	ACTUALROTATIONANGLE	Numeric (Sensor)	measure wafer rotation angle
S20	FIXTURESHUTTERPOSITION	Numeric	open / close shutter setting for wafer shielding
S21	ETCHSOURCEUSAGE	Numeric	counter of use for the grid assembly consumable
S22	ETCHAUXSOURCETIMER	Numeric	counter of the use for the chamber shields consumable
S23	ETCHAUX2SOURCETIMER	Numeric	counter of the use for the chamber shields consumable
S24	ACTUALSTEPDURATION	Numeric (Sensor)	measured time duration for a particular step

The TTF dataset contains the time until the next fault for each of the 3 fault types present. The dataset consists of 3 faults: Flowcool pressure drops below limit, Flowcool pressure too high, and Flowcool leak:

time	TTF_FlowCool Pressure Dropped Below Limit	TTF_Flowcool Pressure Too High Check Flowcool Pump	TTF_Flowcool leak
3907040	28	8523264	9308
3907044	24	8523260	9304
3907048	20	8523256	9300
3907052	16	8523252	9296
3907056	12	8523248	9292
3907060	8	8523244	9288
3907064	4	8523240	9284
3907068	0	8523236	9280
3907072	7998	8523232	9276
3907076	7994	8523228	9272
3907080	7990	8523224	9268

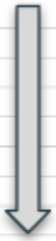

Figure 1: Snapshot of how TTF Dataset is organized

There is missing data throughout the dataset in large portions, sometimes hundreds of thousands of seconds of data would be missing in a row, meaning the time

column would skip those points entirely. There were even faults logged in the fault file that had timestamps listed for a fault, but neither of the other datasets had data surrounding that timestamp. This made the disparate faults already more sparse in the dataset. There are 109 faults, not all of which are usable due to missing data, and there are over 5 million data points in the train and TTF file. The fault dataset is therefore not used, in addition, it achieves the same results as finding where TTF is 0, so it was not used.

Preprocessing and EDA

To start our analysis, we first need to perform feature engineering to keep only the important rows and columns. After dropping the nulls and duplicate columns, the categorical columns were dropped to lower the complexity of the models. Second, the columns that were constants were dropped as they had no effect on the models. A correlation matrix was then used to view the highly correlated columns and saw the following results:

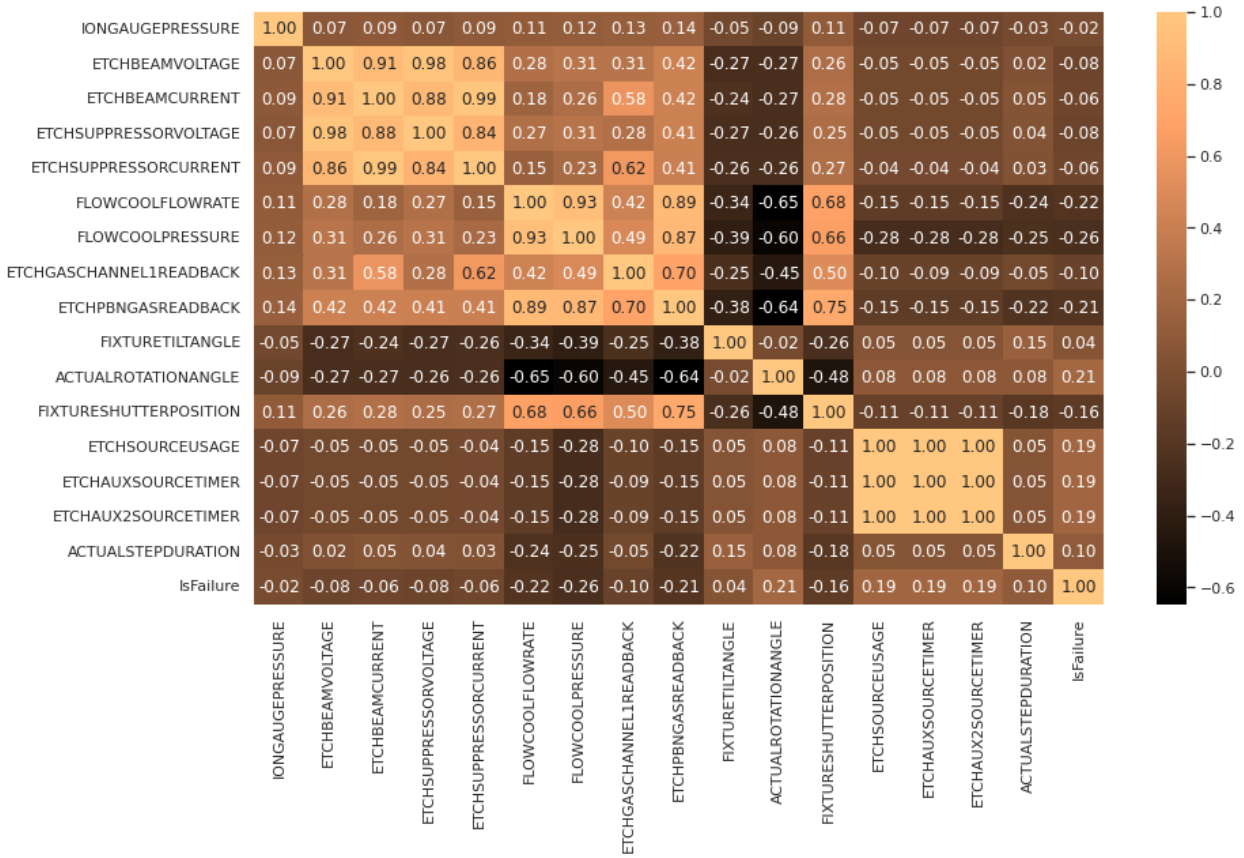

Figure 2: Data Correlation Matrix

We then dropped the columns that had a high correlation. The columns that were left were plotted on line graphs, with red vertical lines plotted at the points of failure for

each fault. For an initial understanding of the data surrounding the faults, a subset of these are shown below:

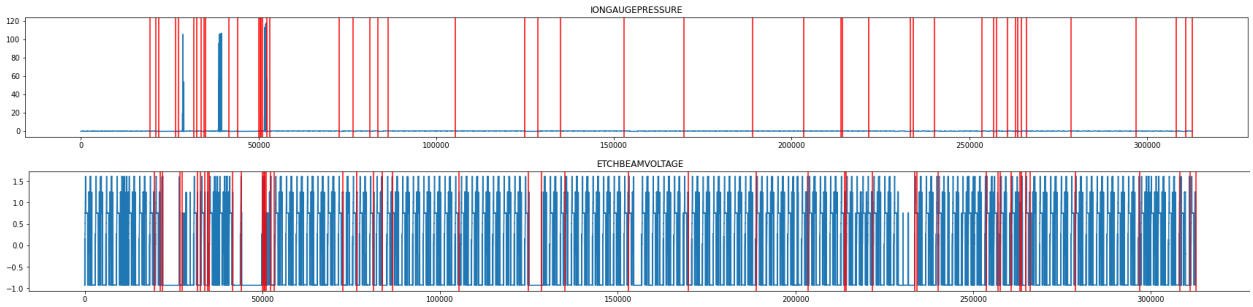

Figure 3: FlowCool Pressure Dropped Below Limit (Fault 1), IONGAUGEPRESSURE and ETCHBEAMVOLTAGE

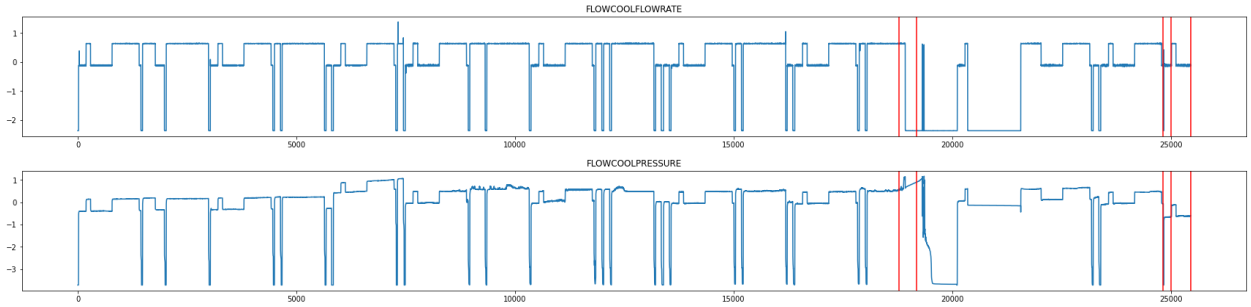

Figure 4: Flowcool Pressure Too High Check Flowcool Pump (Fault 2), FLOWCOOLFLOWRATE and FLOWCOOLPRESSURE

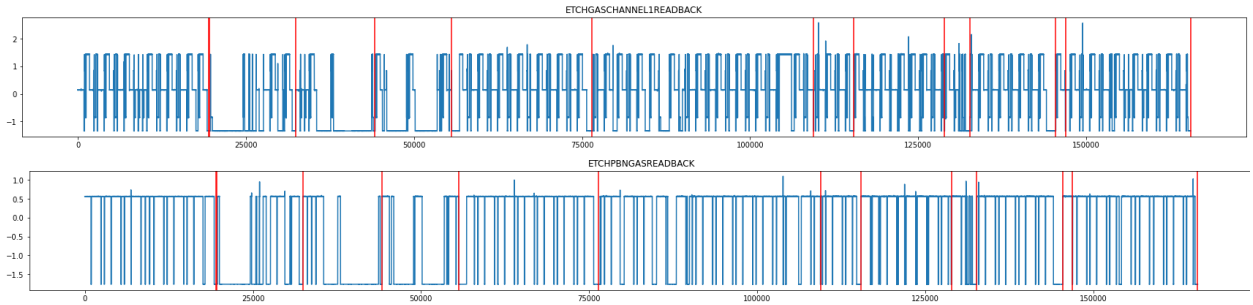

Figure 5: Flowcool Leak(Fault 3), ETCHGASCHANNEL1READBACK and ETCHPBNGASREADBACK

We further looked at how our data is distributed using density plots. The density plots were generated for both normal runs (0) and failure cases (1) to capture the significance of each feature in our dataset to determine if the machine is failing or not. For most features, the dataset was multi-modal. A sample density plot for TTF_FlowCool Pressure Dropped Below Limit is shown below:

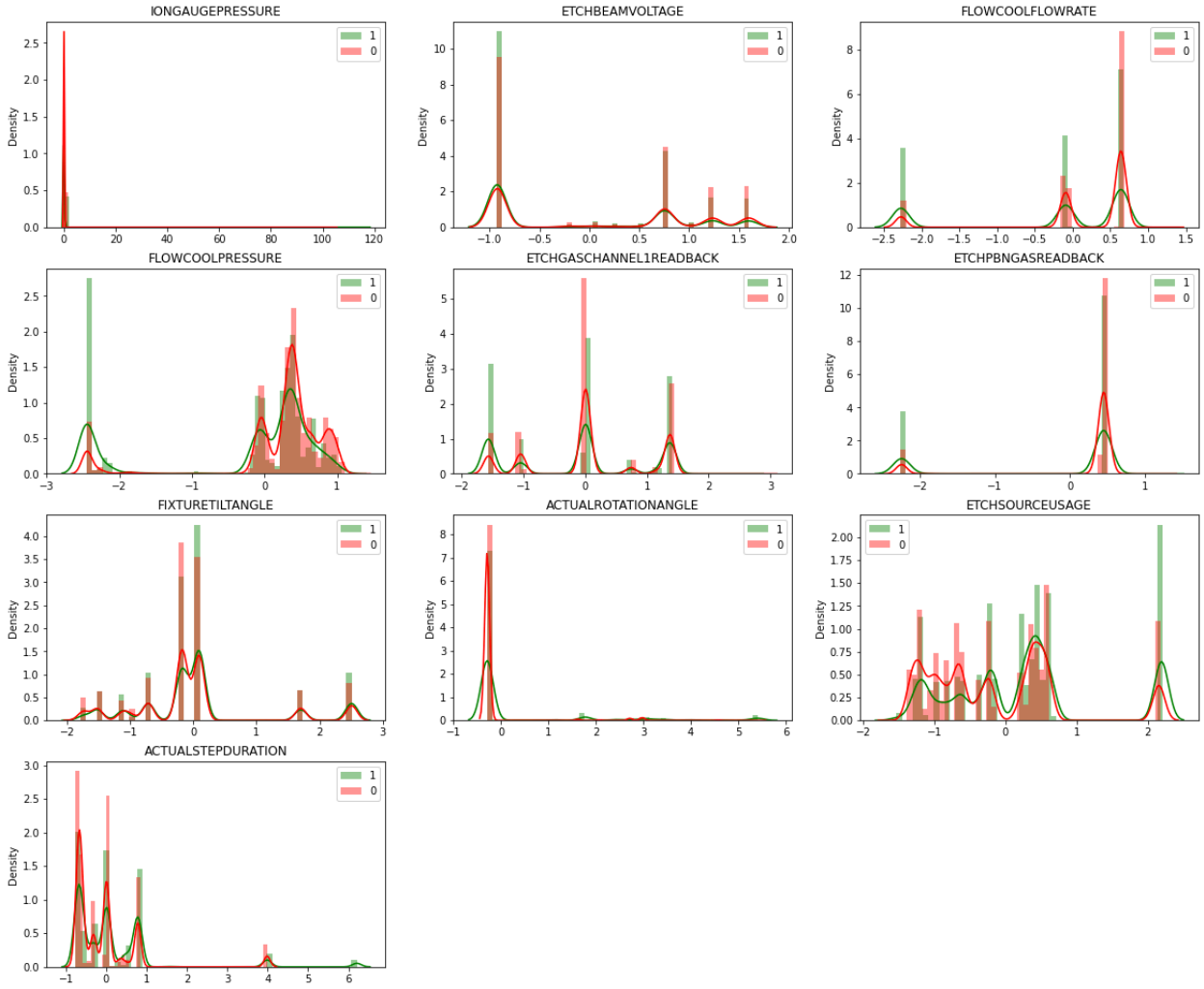

Figure 6: Density Plots

Due to the time series nature of the dataset, the train test split was done manually by finding the indices of the TTF plot where TTF was equal to 0 for each fault type, and picking a slice of one occurrence of the data going from a TTF that reached the provided cutoff to 0. This slice is the test set, while the rest of the data with that slice of data removed was used for the train set.

Defining our Cutoffs

Now that we have the important columns, we need to ensure we are only looking at the important rows. Since the TTF was extremely high for certain faults, we had no need to use data that was over a certain length from a fault. We named this the cutoff point, and had it set to 1 day (86400 seconds) worth of data. A 1 day cutoff was picked because there exists many faults in the dataset that happen quickly (less than 1 day),

and the machines with data spanning longer than a day didn't show significant data until less than 1 day out.

We then had another cutoff point that determined how far from a fault we are classifying as a fault point. We did this since in the description of the dataset, it says "The actual start of the failure may occur much earlier than the provided failure time—this time is not provided" (PHM, 2018). To find our cutoff points, the Birch clustering model took an unsupervised approach to find fault data for ANY fault within the dataset. and found a maximum TTF of 20968.0 where it found significant data. Henceforth, we used a fault cutoff point of 20000 for the 86400 set of data.

Other cutoff points we decided to explore were 3 hours, 5.5 hours, and 8 hours (10800 seconds, 19800 seconds, and 28800 seconds respectively). For the 3 hour example, we decided to use a fault cutoff point extremely close to the fault (10 minutes) to see how the data would fit data when the cutoff is directly adjacent to a TTF of 0. For the 8 hour example, we wanted to test data a bit further away from the fault in relation to the data included in the initial cutoff, so a halfway cutoff of the 25% used in the 1 day example and the 5% used in the 3 hour example was used, meaning we used a failure cutoff of 12.5% (20000/86400 ~ 25%, 600/10800 = 5%, 3600-28800 = 12.5%).

Classification

Using the cutoffs, a classification feature is created to designate a fault. Being able to detect a fault early is a critical component of predictive maintenance, however according to Huang et. al (2018), this leads to imbalanced datasets and most classifiers tend to predict the majority class and have poor performance on predicting the minority class. In this case, the minority class is the probability of a fault happening. In addition, failures are rare events causing highly imbalanced datasets. Since there are no preset cutoffs set for estimating the RUL four different scenarios are created to predict fault times**.** SMOTEENN, a sampling method, is used which combines oversampling and under-sampling approaches to deal with imbalanced data. It is used in classifiers that do not perform well in estimating failure or the minority class.

In Scenario 1, the cutoff is at 10800 seconds to fault and a TTF of 600 seconds is considered as failure points. In Scenario 2, 19800 seconds to failure is considered the boundary point and 2400 seconds as failure data points. Scenario 3, any data 28800 > RUL is considered normal and data 3600 seconds are considered close to fault. Scenario 4, is any data below 86400 seconds and failure is defined at 20000 seconds. Cross-validation was used to check the performance of the models and different hyperparameters applied to improve the performance of the models. To avoid

overfitting, a section of the dataset was considered constituting 1.5 million rows for the first fault.

Table 3: Flowcool Pressure Dropped Below Limit (Fault 1)

Models	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Random Forest Classifier	0.858592	0.961331	0.993075	0.989732
Gradient Boosting Classifier	0.865138	0.960759	0.989673	0.987736
Logistic Regression	0.032259	0.345086	0.583617	0.579865

Table 4: Flowcool Pressure Too High (Fault 2)

Model 1	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Random Forest Classifier	0.992476	0.998324	0.999333	0.999554
Gradient Boosting Classifier	0.993367	0.997488	0.998888	0.999554
Logistic Regression	0.561766	0.794380	0.721356	0.658448

Table 5: Flowcool Leak (Fault 3)

Model 1	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Random Forest Classifier	0.940691	0.964918	0.984791	0.981814
Gradient Boosting Classifier	0.903621	0.937468	0.946776	0.888114
Logistic Regression	0.000000	0.425200	0.509740	0.003080

Feature permutation importance was run for different scenarios, however, different feature importance depended on the different scenarios, so no conclusive findings on most important feature.

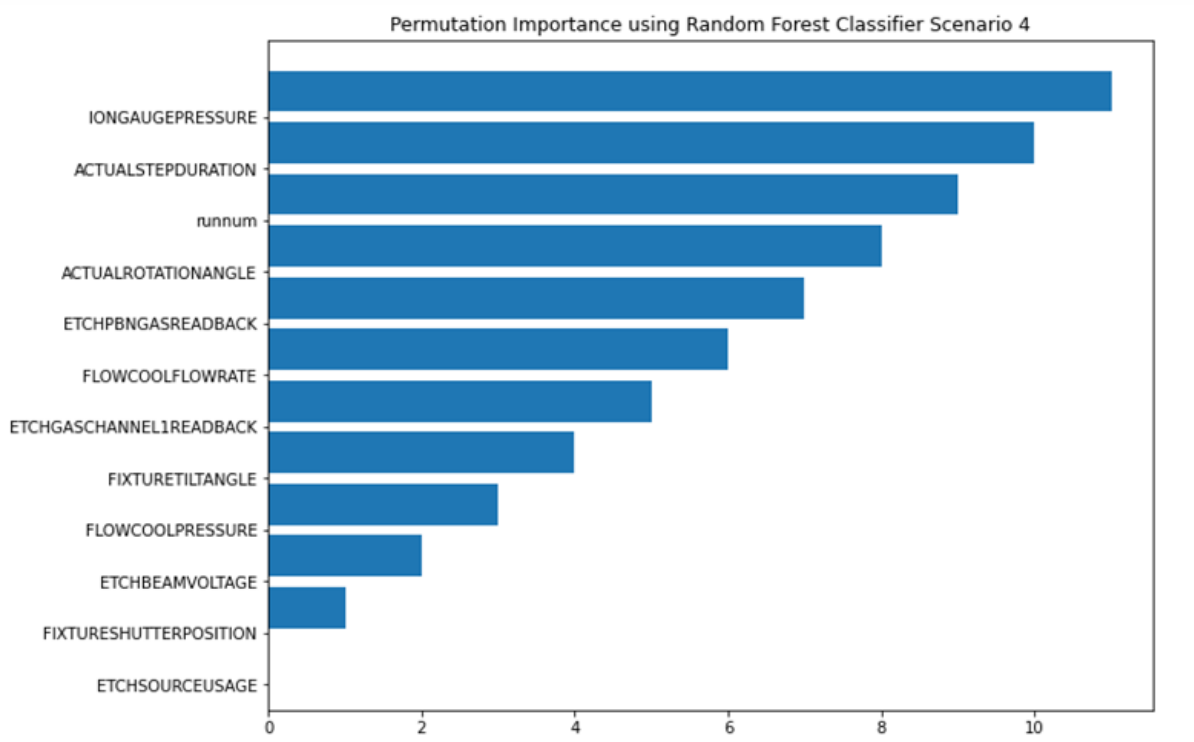

Figure 7: Sample Feature Importance

Regression

Now that we know we can find data that warns us of a fault, we want to be able to know how long we have until the machine actually fails. For regression analysis, we used a variety of models, including: Long Short-Term Memory (LSTM), Random Forest Regression (RFR), Lasso, and Ridge. For each model, we take the R-Squared (R2), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). The results are as shown below:

Table 6: TTF_FlowCool Pressure Dropped Below Limit (Fault 1)

	LSTM			RFR		
Cutoff	R2	RMSE	MAE	R2	RMSE	MAE
86400	-0.32	26623	22280	-0.33	31171	29540
25000	-0.19	7846	6688	-0.19	12042	11830
10000	-0.32	3317	2822	0.25	6221	6119

	Lasso			Ridge		
Cutoff	R2	RMSE	MAE	R2	RMSE	MAE
86400	-0.12	32484	32398	-0.12	32483	32397
25000	-0.21	9884	9824	-0.21	9884	9823
10000	-0.35	4399	4313	-0.35	4401	4315

Table 7: TTF_Flowcool Pressure Too High Check Flowcool Pump (Fault 2)

	LSTM			RFR		
Cutoff	R2	RMSE	MAE	R2	RMSE	MAE
86400	-2.64	61849	59523	-2.07	28369	22432
25000	-2.88	7040	6551	-2.91	7867	7039
10000	-1.82	5135	4385	-1.41	6095	5883

	Lasso			Ridge		
Cutoff	R2	RMSE	MAE	R2	RMSE	MAE
86400	-	-	-	-	-	-
25000	-	-	-	-	-	-
10000	-0.85	5061	4792	-0.85	5061	4793

Table 8: TTF_Flowcool Leak (Fault 3)

	LSTM			RFR		
Cutoff	R2	RMSE	MAE	R2	RMSE	MAE
86400	0.23	21529	20076	-0.01	39788	38161
25000	0.20	5891	4971	-0.03	9006	8596
10000	-0.32	2158	1750	-0.23	3843	3190

	Lasso			Ridge		
Cutoff	R2	RMSE	MAE	R2	RMSE	MAE
86400	0.02	41390	41198	0.02	41388	41196
25000	0.14	10233	10040	0.14	10233	10040
10000	0.20	3489	3243	0.20	3485	3238

Lasso and Ridge were omitted for Fault 2 due to a sharp jump in the regressor's predictions, throwing the results off to an unusable number.

Results

In the classification, the results of Random Forest Classifier, Gradient Boosting Classifier, and Logistic Regression were compared. In scenario 1, the data was imbalanced, however, the Random Forest Classifier and Gradient Boosting Classifiers were able to predict the minority class with relatively good performance of 86 and 83 percent respectively. The Logistic Regression was able to predict the majority class but performed very poorly at predicting the minority class at 3 percent. In the second, third and fourth scenario, the data was not imbalanced and therefore the Random Forest and Gradient Boosting models had very good performance in classifying the minority groups. The Logistic Regression performed at 32, 58 and 58 percent respectively for the flowcool pressure dropped below limit. The flowcool pressure too high and the flowcool leak faults showed similar model performance.

After fine tuning the hyperparameters the overall performance of the Random Forest and Gradient Boosting Classifier were slightly improved but their performance in predicting the minority class slightly decreased. Fine tuning the hyperparameters however improved the performance of the Logistic Regression in scenario 1 from 3

percent to 18 percent for the flowcool pressure dropped below limit. After applying SMOTEENN the overall performance of the Logistic Regression model reduced, but the performance to predict minority class improved from 3 percent to 85 percent.

The results of the regressor are poor with high MAE and RMSE, and an R2 close to 0 or negative. Fault 3 had the best performance for predicting the RUL, but even still had results that were less than desirable. For fault 3, when more data was present, the LSTM was producing the best performance, while Lasso was performing better for the cases with fewer data. Fault 2 had very few failure points due to the missing data, thus making the analyses surrounding this fault perform poorly. For fault 1, the LSTM performed the best when looking at the RMSE and MAE, but did poorly for R2.

Production Environment Mockup

To see the use of these models and analyses in a real life environment, an app was built in Streamlit to see what monitoring a machine's faults would be like online. First, open this Streamlit Page and upload the csv named 690_Fault_1 into the Upload File section. From there, the features will be plotted one data point at a time with the words "Machine running steady" printed on the top right. This means the data being plotted is data that is being classified as standard runtime data. Under the feature's name is the level of the feature's importance, with 1 being the highest and 10 being the lowest.

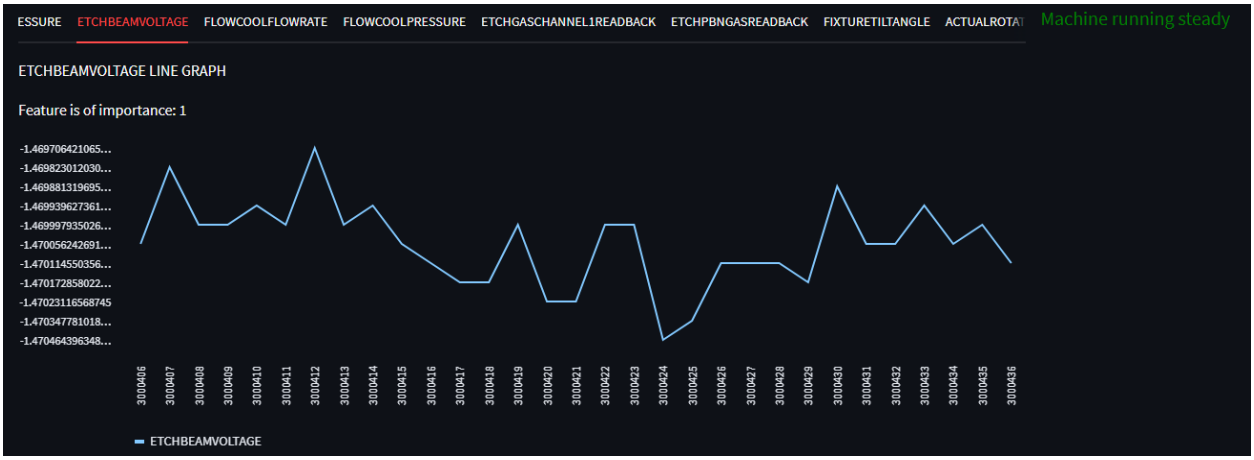

Figure 8: Data being streamed

After the classifier sees a data point that looks like a failure, the text at the top right will turn yellow and print "Warning" and the regressor's prediction of how long until the machine fails. The regressor in this case is an overfit regressor to properly show the functionality of the tool.

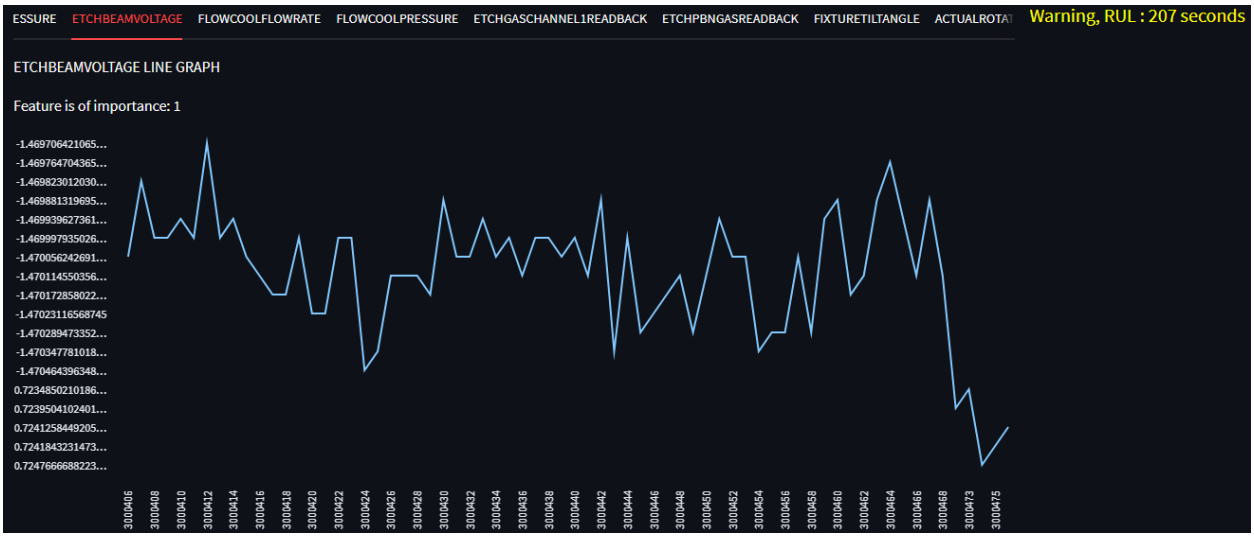

Figure 9: Regressor triggered warning

After the regressor hits 0 seconds, the text turns red and says “MACHINE HAS FAILED”, signifying the machine is currently failing, and must be stopped immediately.

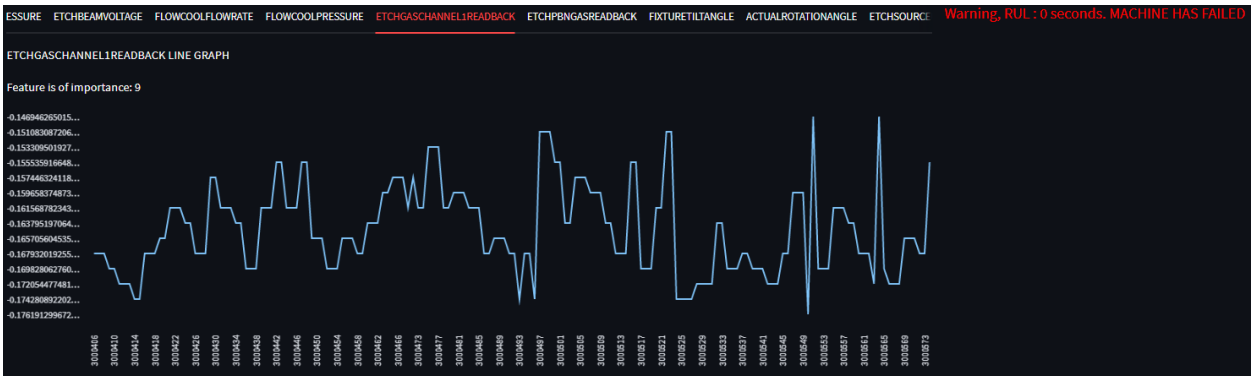

Figure 10: Predicted RUL

Now, using the sidebar on the right, the Historical tab will show all of the data, previous and the data that was just inserted with the Live tab. The historical tab can be filtered by time and pass/fail data. Then, select a feature to explore and see a line plot, histogram, and pie chart for the data within that column.

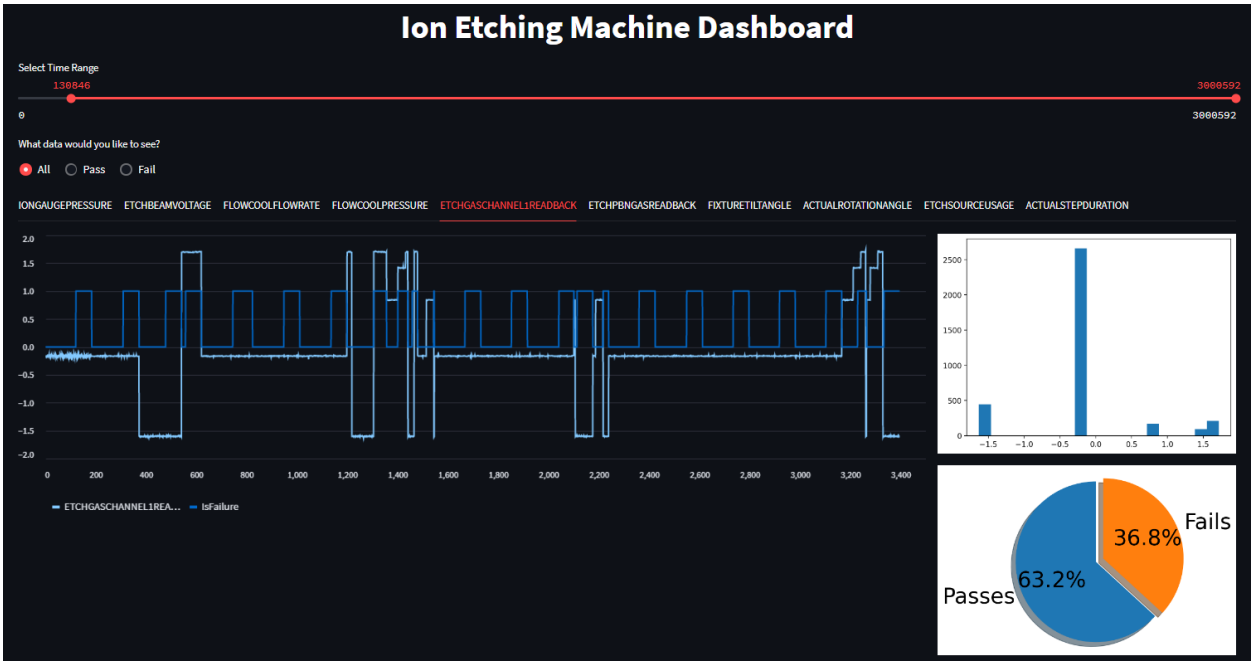

Figure 11: Machine dashboard

Finally, the Data Exploration tab is used to view the data behind the plots. The data points can be clicked (shift and control clicks as well) to save a chunk of data to a CSV to your personal machine. This way the data that seems to be causing issues can be easily found and further analyzed within other programs. The Production Environment takes the approach that was outlined by Wu et al. where a classifier triggers a regressor to predict the RUL (Wu, 2021).

Conclusion

For the classifier, the Random Forest and Gradient Boosting classifier performed best in classifying minority classes, especially for imbalanced data. SMOTEENN is especially useful in improving the prediction of minority classes in models with poor performance.

For regression, we recommend using LSTM due to its overall performance being better when more data is present. However, the results of our models are low enough that we don't recommend our specific implementation of the LSTM and our cutoffs used to feed the model.

Future Improvements

Time series data from sensors is subjected to random variations. As part of the future improvement, we may benefit from some type of smoothing technique to reduce

the random fluctuations that we observed in our dataset. As stated earlier, the dataset is riddled with missing data points, which is especially important when the missing data is the data describing a fault. A reason we didn't use imputation for the missing data, is because there would be no way to properly simulate the data surrounding the missing faults, and would most likely harm the overall analysis. It would be to our benefit to ensure that the dataset used in our next analysis does not contain so much important missing data.

The other future improvements with the current data would be to improve the performance of the regressors. While the LSTM was tuned, the other regressors were using mostly their default settings, and thus may have underperformed. However, seeing the performance of the LSTM, there is a good chance this wouldn't be the case. Another improvement using the provided data would be to provide more of the data to the regressors to train on. A cutoff of 1 day may have limited the available data too much such that the regressor had trouble singling out the fault related data.

Github

Github Link https://github.com/JacobShaw98/PHM_RUL

References:

Baru, A. Three Ways to Estimate Remaining Useful Life for Predictive Maintenance. (2018). *Mathworks*. https://www.mathworks.com/company/newsletters/articles/three-ways-to-estimate-remaining-useful-life-for-predictive-maintenance.html

Botatakis, J., Chokor, A., Propes, N. PHM DATA CHALLENGE 2018. (2018). *Annual Conference of the Prognostics and Health Management Society*. https://phmsociety.org/wp-content/uploads/2018/05/PHM-Data-Challenge-2018-vFinal-v2_0.pdf

Huang, W., Khorasgani, H., Gupta, C., Farahat, A. and Zhang S. (2018). Remaining Useful Life for Abrupt Failures. Proceedings of the Annual Conference of the PHM Society 2018. 10(1).

Kang, Z., Catal, C. and Tekinerdogan, B. Remaining Useful Life (RUL) Prediction of Equipment in Production Lines Using Artificial Neural Networks. (2021). *Sensors 2021, 21, 932.* https://doi.org/10.3390/s2103093

TV, V., Gupta, P., Malhotra, P., Vig, L., & Shroff, G. Recurrent neural networks for online remaining useful life estimation in Ion Mill Etching System. (2018). *Annual Conference of the PHM Society*, *10*(1). https://doi.org/10.36001/phmconf.2018.v10i1.589

Wu, S., Jiang, Y., Luo, H., & Yin, S. Remaining useful life prediction for ion etching machine cooling system using deep recurrent neural network-based approaches. (2021). *Control Engineering Practice*, *109*, 104748. https://doi.org/10.1016/j.conengprac.2021.104748

Zou, G., Banisoleiman, K., Gonzalez, A., Faber, M. Probabilistic investigations into the value of information: A comparison of condition-based and time-based maintenance strategies. (Sept 15, 2019). *Ocean Engineering Volume 188*. https://doi.org/10.1016/j.oceaneng.2019.106181