The Impact of Sensor Faults on Condition Monitoring of a Hydraulic Actuator

Stephen Adams[1], Dan DeCollo[2], Floyd Steele[3], Nate Brown[4], Sherwood Polter[5], and Peter A. Beling[6]

[1, 2, 6] *Virginia Tech National Security Institute, Arlington, VA, USA*

[3, 4] *Luna Labs, Charlottesville, VA, USA*

[5] *Naval Surface Warfare Center Philadelphia Division, Philadelphia, PA, USA*

ABSTRACT

While supervised machine learning is prevalent in prognostics and health management applications, the success of these models is dependent upon being trained on accurate data. Training data collected with faulty sensors can degrade the performance of these models when deployed in an operational environment. This study investigates the impact of faulty data and the robustness of feature extraction methods and tree-based classifiers. This study also provides an open-source software package for injecting faults into time series data. The numerical experiments are performed on an open-source hydraulic actuator data set and demonstrate that certain features are robust to certain types of faults and that more complex models, such as ensemble techniques, are more robust to sensor faults than simple models. This work suggests that more complex models and larger (and possibly redundant) feature sets may be preferred in situations where sensor faults are likely. Furthermore, certain feature extraction techniques may be selected if certain faults are more likely than others.

1. INTRODUCTION

Machine learning (ML) has had a significant impact on the field of prognostics and health management (PHM) (Rezaeianjouybari & Shang, 2020; Fink et al., 2020; Zhang et al., 2019). When deploying an ML-based PHM system, the data pipeline for many applications is to collect data using sensors from a system, extract features from the sensor signals, and then use the ML model for estimation of the health state or remaining useful life. For successful deployment of the model, it must be trained on data that is representative of the deployed environment. There are numerous sources of error that could cause the deployed environment to differ from the training environment. Furthermore, when designing an ML-based PHM system and the data pipeline, it is critical to understand how the system will respond and adapt to the deployed environment. The data pipeline may be able to be designed to be robust to particular disturbances that are likely to occur in the deployed environment.

Sensors play a critical role in the data pipeline, but like any other system, they can degrade over time or be faulty from the start. Corrupted sensor data can degrade the performance of the ML model by creating differences in the marginal distributions of the training and deployed data. In the ML community, this is often referred to as domain shift and can be addressed using transfer learning techniques (Pan & Yang, 2009). However, transfer learning requires data collection and retraining in the deployed environment. ML models have also been developed to detect sensor faults but these studies do not address mediation of the fault (Argawal, Kalel, Harshit, Domnic, & Singh, 2021; Martakis et al., 2021; Mohapatra, Subudhi, & Daniel, 2020; de Silva et al., 2020; Adams, Beling, Greenspan, Velez-Rojas, & Mankovski, 2018; Liu, Adams, & Beling, 2020). A more straightforward approach is to train models that are robust to differences in the training and deployed environment, essentially zero-shot transfer learning.

Adversarial ML techniques assume that an adversary is intentionally trying to degrade the ML pipeline with the objective of changing the prediction of the model (Wang, Li, Kuang, Tan, & Li, 2019), but most of the work in this area focuses on computer vision (Machado, Silva, & Goldschmidt, 2021). Adversarial training adds corrupted examples to the training set with the objective of making the learned model robust to similar examples during deployment (Wiyatno, Xu, Dia, & De Berker, 2019). Adversarial training could be used to strengthen PHM models against sensor faults, however the techniques used to generate adversarial examples often focus on selecting examples that will most affect the model, which

aligns with the adversary's intent. Sensor faults are naturally occurring and adversarial selection techniques may select examples that are not possible on a real system. Other machine learning methods account for noisy data during the training process but do not explicitly model sensor faults (Gupta & Gupta, 2019; Alzraiee & Niswonger, 2024; Poulinakis, Drikakis, Kokkinakis, & Spottswood, 2023), i.e. specific sensor fault types are not considered. However, it is possible that these methods could be adapted to explicitly account for sensor faults in future studies.

The presented study focuses on sensor faults during the model training process and seeks to understand the impact of common sensor faults on the model's performance in the deployed environment. Essentially, this study investigates the case where sensors on the training testbed are faulty, and the training data has been collected with faults. The first contribution of this work is a study of the relationship between common fault types and feature extraction techniques. Two common faults, a noise fault and an offset fault, are injected into the sensor data before feature extraction, and two statistical features are extracted from the data: the mean and the variance. This study also evaluates the performance of two classification models on sensor faults: classification trees and random forests. There are numerous other feature extraction techniques, such as frequency domain features and higher order moments, and classification models, such as neural networks, but this initial study focuses on a simple case. Other feature extraction methods and models should be incorporated into future work. The second contribution of this work is the release of a software package for injecting faults into data[1]. The software package contains common sensor faults that arise in PHM settings. A similar study has investigated the impact of sensor noise for structural health monitoring (Ibrahim, Eltawil, Na, & El-Tawil, 2019).

Hydraulic actuators are utilized in numerous applications and have been used for many PHM studies (Adams et al., 2016, 2017; Meekins et al., 2018; Cody et al., 2019; Adams et al., 2019; Adams, Cody, Beling, Polter, & Farinholt, 2020; Meekins, Adams, Farinholt, Polter, & Beling, 2020). Much of this prior work focuses on reducing the power consumption of the model by selecting simple models (Adams et al., 2016) and implementing feature selection (Adams et al., 2017; Meekins et al., 2018, 2020). One of the results of the presented work is that more complex models, such as the random forest - an ensemble of classification trees, and diverse sensor and feature sets create models that are robust to sensor faults. The numerical experiments in the presented study use an open-source hydraulic actuator data set (Helwig, Pignanelli, & Schtze, 2015; Helwig, Pignanelli, & Schütze, 2015a; Schneider, Helwig, & Schütze, 2017). A similar study has investigated sensor faults with this data set (Helwig, Pignanelli, & Schütze, 2015b). The presented study builds upon the previous study by focusing on designing a robust PHM system through feature extraction and classifier selection, instead of identifying faulty sensors and eliminating them from the classifier's input. Furthermore, the presented study uses tree-based methods that outperform the linear discriminate analysis in the previous study.

This study is organized as follows. Section 2 outlines the capabilities of the fault injector software package. Section 3 describes the statistical methods used for analyzing the features, and Section 4 presents the numerical experiments. Section 5 outlines the conclusions of this study and areas for future work.

2. FAULT INJECTOR

The Fault Injector software package allows users to augment data in a way that represents a signal fault. The Fault Injector has six different faults:

1. Drift - a linear line is added to the original signal data resulting in the signal drifting further from the original over time,
2. Offset - a constant value is added to the original signal,
3. No output (NaN) - all original values replaced with NaN representing no output from the sensor,
4. Stuck value - all the faulty values will be equal to the same value,
5. Gaussian noise - Gaussian random noise is added to the original signal, and
6. Uniform noise - a uniform random variable $[0, 1]$ is added to the original signal.

All of the fault options augment the data in different ways, while still resembling actual faults that can occur with sensors. Additionally, they are customizable to inject different fault magnitudes and varying lengths of data.

The standard ML pipeline for PHM applications is to collect sensor signals, extract features from the signal, feed the extracted data to the ML model, and produce a prediction. The faulty pipeline investigated in this study corrupts the sensor signal before feature extraction. Figure 1 displays the standard ML pipeline (top) and the faulty ML pipeline (bottom).

3. STATISTICAL METHODS

All sensors have some amount of measurement error, which can be broken down into two types of error: variance and bias. Let τ_0 represent the standard deviation of the measurement error, and let b_0 represent the bias of the measurement error. Therefore, the collected signal s can be modeled as

$$s = s_0 + b_0 + \epsilon_0, \tag{1}$$

[1] $https://github.com/vtnsi/fault\_injector$

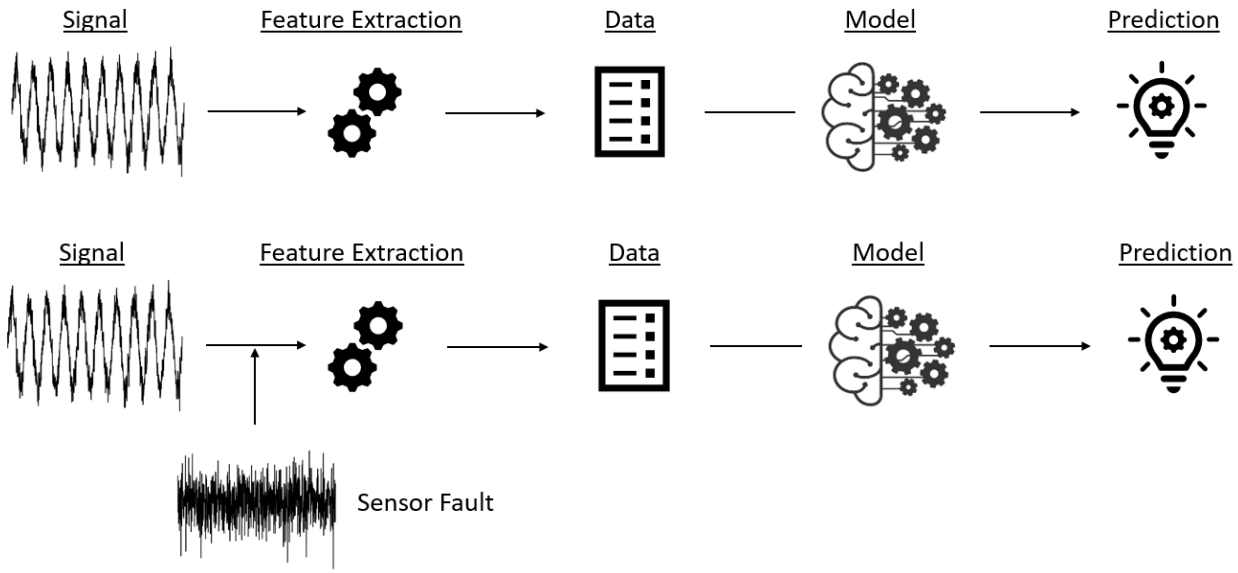

Figure 1. The top figure displays standard ML pipeline for PHM applications. The bottom figure displays the pipeline with a noisy sensor fault.

where s_0 is the true signal, and $\epsilon_0 \sim \mathcal{N}(0, \tau_0^2)$.

There are many types of sensor faults. This study implements two types of faults: a noise fault and an offset fault. A noise fault adds additional random noise to the collected signal, and the offset fault adds additional bias to the collected signal. These types of faults are present when the collected signal contains error that is greater than the measurement error. Let s' represent a sensor signal with a sensor fault

$$s' = s + b + \epsilon, \tag{2}$$

where b is the bias from the offset fault, and $\epsilon \sim \mathcal{N}(0, \tau^2)$ is the random noise from the noise fault.

Assume that there are I sensors, and J features to extract from each sensor. Let $x_{ij} = h_j(s_i)$ represent the feature extraction function for feature $j = 1...J$ that converts the sensor signal for sensor $i = 1...I$ to x_{ij}. The individual features can be accumulated into a single array $\mathbf{x} = [x_{11}, ..., x_{IJ}]$. Let $y = f(\mathbf{x})$ be a classifier that maps the feature array to classes $y = \{1, ..., C\}$. The entire machine learning pipeline can be modeled as $y = f(h(\mathbf{s}))$, where $\mathbf{s} = [s_1, ..., s_I]$.

There are numerous feature extraction functions that could be used for $h(s)$. In this study, the mean and the variance of the sensor signals are used as features. Let μ and σ^2 represent the mean and variance of s collected for T time steps. The calculation of the mean and variance is

$$\mu = \frac{\sum_{t=1}^{T} s_t}{T}, \tag{3}$$

and

$$\sigma^2 = \frac{\sum_{t=1}^{T} (s_t - \mu)^2}{T}. \tag{4}$$

When a noise fault is present, $b = 0$ and $\tau > 0$, so $s' = s + \epsilon$. After feature extraction, the mean and variance of the faulty sensor signal are $\mu' = \mu$ and $\sigma^{2'} = \sigma^2 + \tau^2$. When an offset fault is present, $b > 0$ and $\tau = 0$. After feature extraction, $\mu' = \mu + b$ and $\sigma^{2'} = \sigma^2$. In summary, a noise fault should not affect the mean features, and an offset fault should not affect the variance.

These results can be proven by treating the sensor signal as a random variable S with expectation $\mathbb{E}[S]$ and variance $\text{Var}[S]$. The noise fault is the addition of two independent random variables, so

$$\begin{aligned} \mathbb{E}[S + \epsilon] &= \mathbb{E}[S] + \mathbb{E}[\epsilon] \\ &= \mathbb{E}[S], \end{aligned} \tag{5}$$

and

$$\begin{aligned} \text{Var}[S + \epsilon] &= \text{Var}[S] + \text{Var}[\epsilon] + 2\text{Cov}[S, \epsilon] \\ &= \text{Var}[S] + \text{Var}[\epsilon], \end{aligned} \tag{6}$$

where $\text{Cov}[S, \epsilon]$ is the covariance. The offset fault is the addition of a constant to a random variable, so

Table 1. Summary of Sensors. This table contains the sensor type, the number of sensors of each type in the data set, the units for each sensor, and the sampling rate.

Sensor Type	# Sensors	Units	Hz
Pressure	6	bar	100
Temperature	4	degrees C	1
Flow	2	l/min	10
Motor Power	1	W	100
Vibration	1	mm/s	1
Cooling Efficiency	1	%	1
Cooling Power	1	kW	1
Efficiency Factor	1	%	1

$$\begin{aligned}\mathbb{E}[S+b] &= \mathbb{E}[S] + \mathbb{E}[b] \\ &= \mathbb{E}[S] + b,\end{aligned} \tag{7}$$

and

$$\begin{aligned}\mathrm{Var}[S+b] &= \mathbb{E}[(S+b) - \mathbb{E}[S+b]]^2 \\ &= \mathbb{E}[S - \mathbb{E}[S]]^2 \\ &= \mathrm{Var}[S].\end{aligned} \tag{8}$$

4. NUMERICAL EXPERIMENTS

This section outlines the data set used in the numerical experiments, the benchmark experiments, and the single and multi-sensor fault analysis.

4.1. Data Set

The hydraulic actuator data set used in the numerical experiments is publicly available on the UCI Machine Learning Repository (Helwig, Pignanelli, & Schtze, 2015). The data set contains 2205 observations. Seventeen sensors are used to collect data from the hydraulic actuator at various sampling rates. More information on each sensor type is displayed in Table 1.

The data set contains four target conditions.

1. Cooler condition (3 states): 100% - full efficiency, 20% - reduced efficiency, and 3% - near failure.
2. Valve condition (4 states): 100% - optimal switching, 90% - short lag, 80% - severe lag, and 73% near failure.
3. Internal pump leak (3 states): 0 - no leak, 1 - small leak, and 2 - severe leak.
4. Hydraulic accumulator (4 states): 130 bar - ideal pressure, 115 bar - slightly reduced pressure, 100 bar - severely reduce pressure, and 90 bar - near failure.

Each value for the target variable state is treated as a discrete class. The distribution of the target variables is displayed in Figure 2.

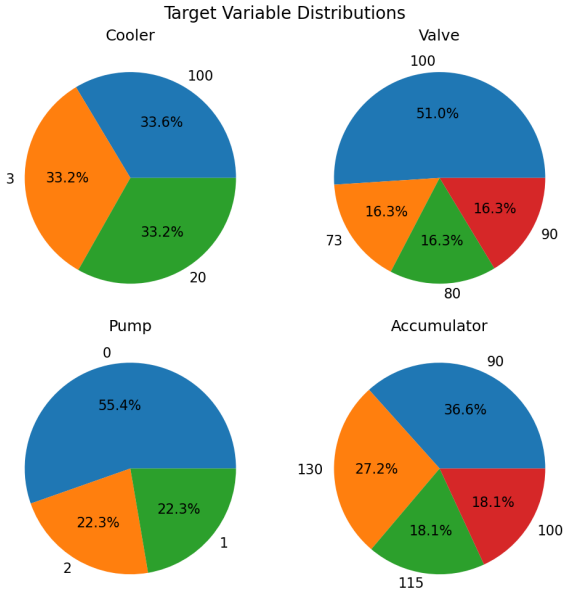

Figure 2. Distribution of target variables.

4.2. Benchmark Experiments

This section outlines the benchmark experiments that were conducted to establish the accuracy of the models without sensor faults in the training data. Two models are evaluated: classification trees (Rokach & Maimon, 2010) and random forests (Breiman, 2001). For this set of experiments and all following experiments, default parameter settings from the scikit learn Python package (Pedregosa et al., 2011) are used for the model, e.g. 100 estimators for the random forest. The data set is split into training and testing sets with one third of the data withheld for testing. Two features are extracted from the sensor data: the mean and the variance. Before the features are extracted, the data from the sensors are normalized between 0 and 1. Models are trained on each feature set and a combination of the feature sets, i.e. a data set with the mean and variance features. The accuracy of each model is calculated on the test set. Each condition in the data set, e.g. cooler condition, is treated as a unique target variable and numerical experiment. The training process is displayed as a block diagram in Figure 3. The dashed line around the Fault Injection block indicates that this step is not performed during the benchmark experiments.

Table 2 contains performance results for the benchmark experiments. For the Accumulator target variable, the random forest model trained using both feature sets (mean and variance) achieved an accuracy of 98%. The random forest using just the mean features also had an accuracy of 98%, but the random forest using just the variance features only had an ac-

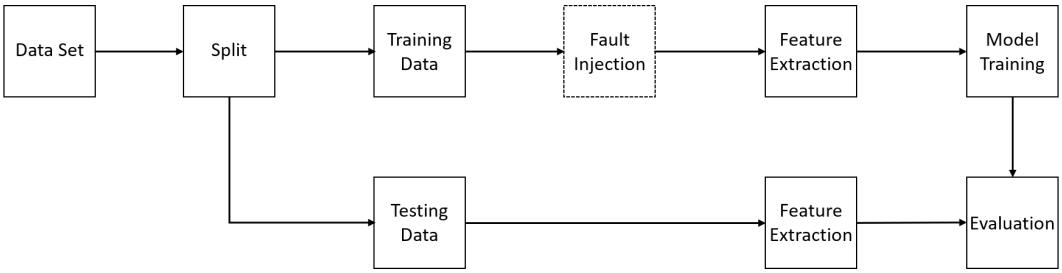

Figure 3. Training process. The dashed lines around the Fault Injection block indicates that this step is not conducted during the benchmark experiments.

Table 2. Benchmark performance measures.

Target	Model	Mean	Variance	Mean+ Variance
Accumulator	Tree	0.930	0.809	0.930
	RF	0.984	0.924	0.982
Cooler	Tree	0.997	0.986	0.997
	RF	0.997	0.996	0.997
Pump	Tree	0.993	0.966	0.984
	RF	0.997	0.982	0.996
Valve	Tree	0.913	0.930	0.941
	RF	0.972	0.985	0.984

curacy of 92%. The classification tree was less accurate with an accuracy of 93% when using the combined feature set and the mean features and an accuracy of only 81% when using the variance features. For the Cooler target variable, both models and all feature sets had a high accuracy above 98%. The models for the Pump target variables also achieved high accuracy scores above 96%. For the Valve target variable, the random forest had an accuracy above 97% for all feature sets. The classification tree was less accurate but still maintained accuracy above 91%.

4.3. Single Sensor Faults

This section outlines the numerical experiments where a fault was added to a single sensor in the training set before feature extraction. The test data is not corrupted with a fault. The fault was added to the entire length of the observation for that sensor replicating the situation where the sensor is consistently generating the fault. Every sensor was used as the fault sensor. The standard deviation for the noise fault was $\sigma = [0.25, 0.5, 0.75, 1.0, 1.5, 2.0]$. The value of the offset used the same values as the standard deviation for the noise fault. Similar to the previous section, the mean and variance are used as features along with a combined feature data set.

Figures 4 and 5 present the results for the noise fault and the offset fault respectively. The mean over the different sensors for each fault parameter value is displayed. Generally, a fault in a single sensor does not affect the accuracy of the model on the test set. This can be observed by comparing the results in Figures 4 and 5 to the benchmark results in Table 2. Similarly, increasing the magnitude of the fault does not decrease performance. Therefore, we can conclude that for the actuator data set corrupting a single sensor will not impact the performance of a model.

4.4. Multi-Sensor Faults

This section sequentially corrupts sensors in the training set. The order of the sensors is determined by the feature importance calculated from a model trained on an uncorrupted training set. The importance of the sensor is estimated using the mean of the features from that sensor. Let η_i^{μ} represent the estimated feature importance for the mean of the i^{th} sensor, and let $\eta_i^{\sigma^2}$ represent the estimated feature importance for the variance of the i^{th} sensor. Then the importance of the i^{th} feature is

$$\eta_i = \frac{\eta_i^{\mu} + \eta_i^{\sigma^2}}{2}. \tag{9}$$

For thoroughness, the same parameters as the previous experiments are used for the magnitude of the fault, however the magnitude has little impact on the performance. The presented results reflect the mean of the accuracy over the different magnitudes of the fault.

Figures 6 and 7 display the results for the noise and offset faults respectively where the order is least important sensor to most important sensor. When adding noise to the mean feature subset, the performance across all labels does not start to drop until more than 10 sensors are corrupted for both models. While theoretically adding noise should not affect the mean feature subset, the mean of the finite sample from ϵ is non-zero. When adding noise to the variance feature subset, the effect can be seen almost immediately, and the classification tree degrades faster than the random forest. The results for the combined feature set show a decline but more features need to be corrupted for a significant effect, and the classification tree degrades faster than the random forest model. When adding an offset to the mean feature subset, performance begins to degrade quickly. However, adding an offset to the variance feature subset has no effect. As before, the

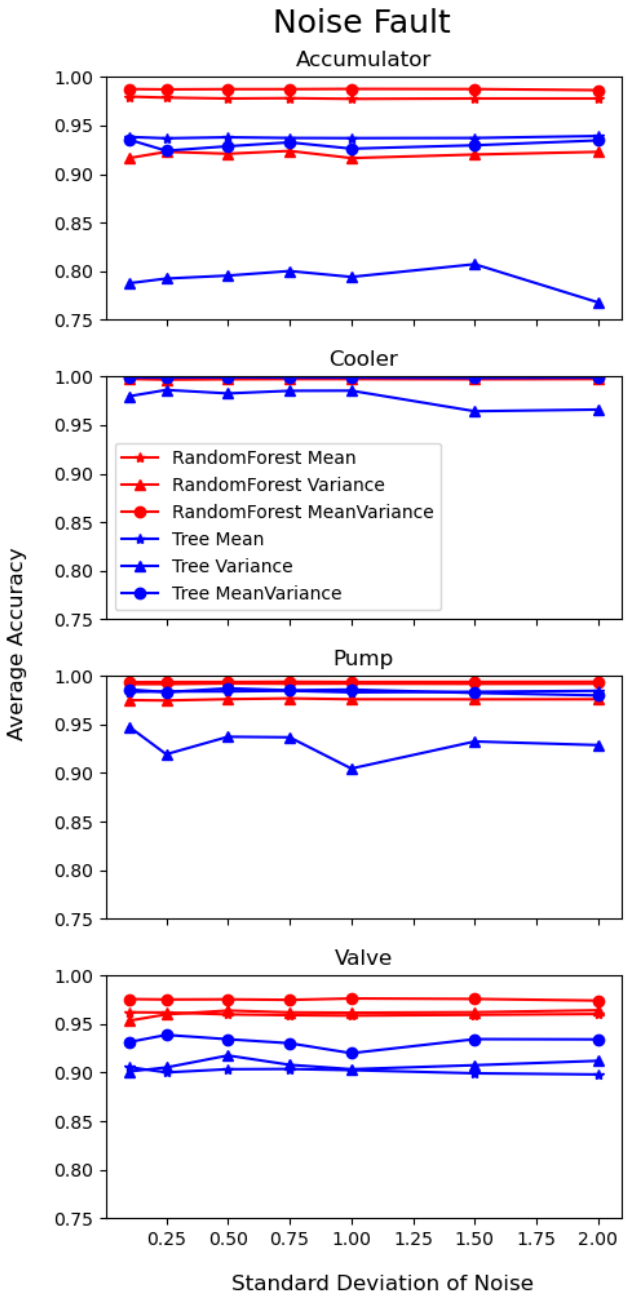

Figure 4. Average accuracy when a noise fault is injected into a single sensor. The average accuracy is over the sensors for each value of the standard deviation of the noise.

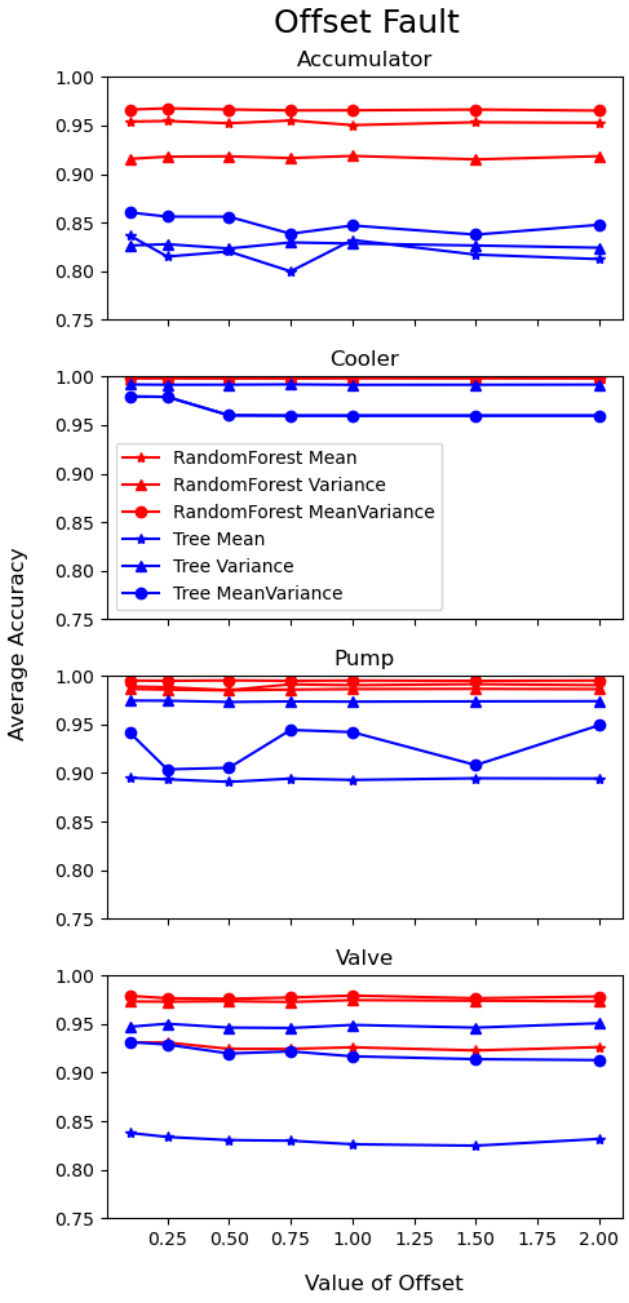

Figure 5. Average accuracy when an offset fault is injected into a single sensor. The average accuracy is over the sensors for each value of the offset.

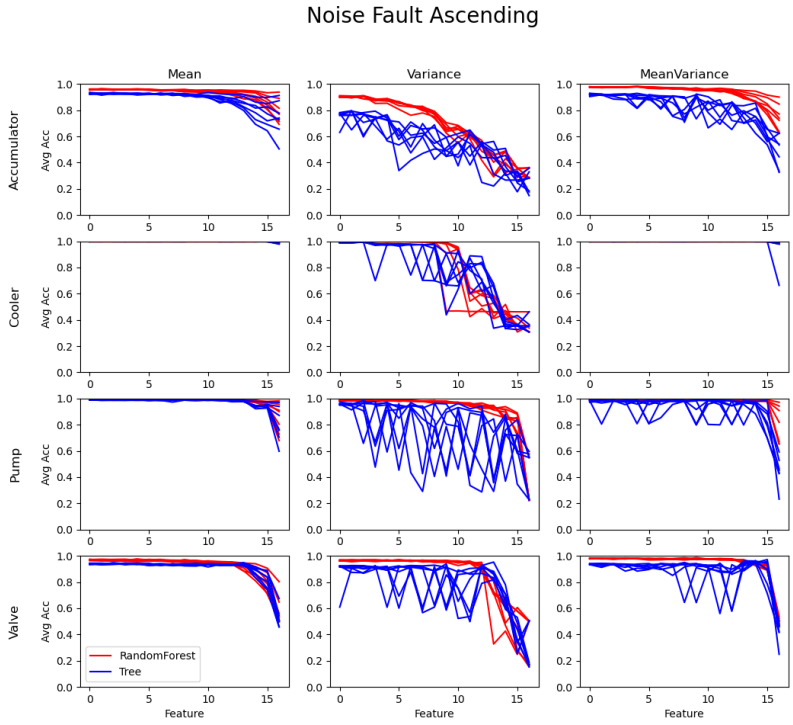

Figure 6. Accuracy as noise is added to each feature sequentially starting with the least important feature.

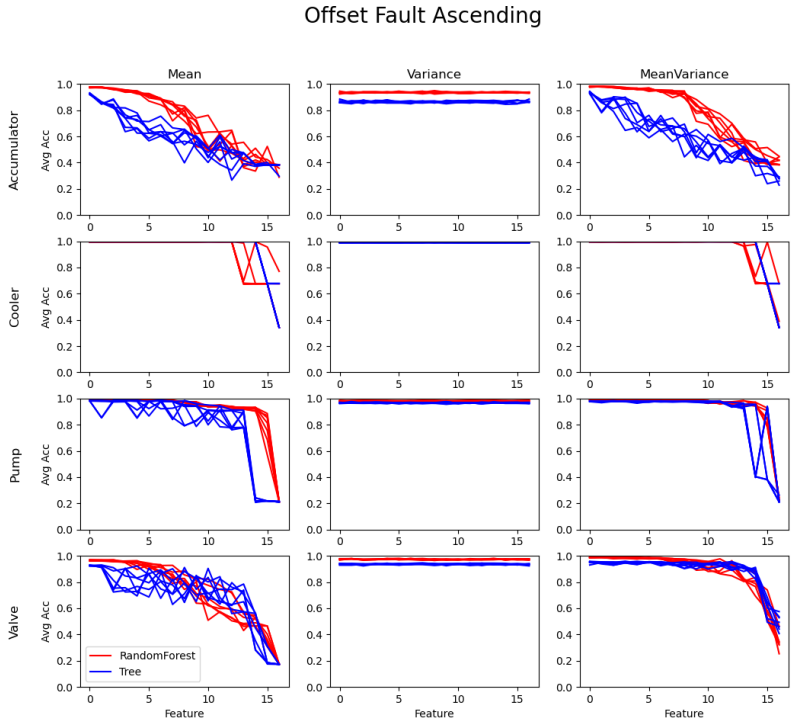

Figure 7. Accuracy as offset is added to each feature sequentially starting with the least important feature.

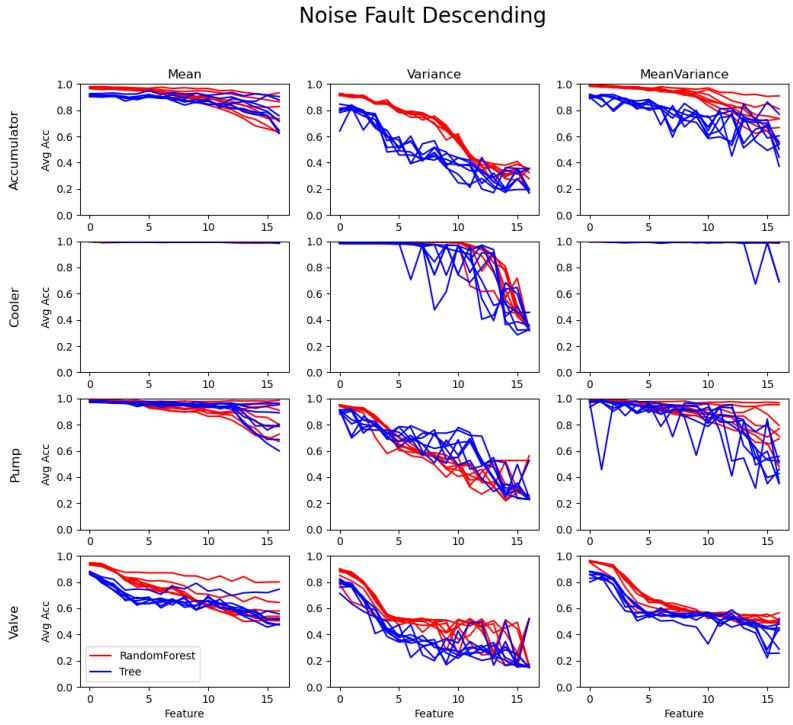

Figure 8. Accuracy as noise is added to each feature sequentially starting with the most important feature.

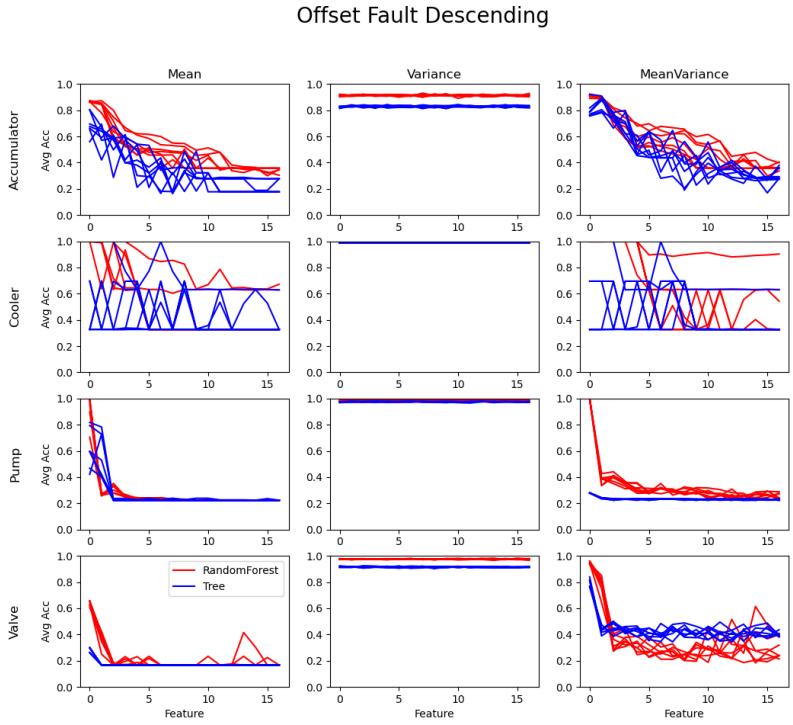

Figure 9. Accuracy as offset is added to each feature sequentially starting with the most important feature.

combined feature set shows a degradation of performance but more sensors need to be corrupted for a significant impact.

Figure 8 and 9 display the results for the noise and offset faults respectively where the order is most important to least important. As would be expected, the noise fault has little impact on the mean feature subset until a larger number of sensors are corrupted but an immediate impact on the performance of the variance feature subset. The combined feature set degrades at a slower rate than the variance feature subset. Similarly, the offset fault quickly degrades the mean feature subset, has little impact on the variance feature subset, and slowly degrades the performance on the combined feature set.

In summary, the impact of the fault is dependent on the type of fault and the extracted features. The mean feature appears to be robust to the noise fault, and the variance feature appears to be robust to the offset fault. If the type of fault that may occur is unknown, a combined feature set may balance the tradeoffs between the individual features. Furthermore, the random forest appears to be more robust to sensor faults than the classification tree. This observation aligns with theory as ensemble techniques should be more robust to corrupted data.

5. CONCLUSION

This study demonstrates, through a statistical analysis, that certain types of feature selection methods should eliminate the impact of certain types of sensor faults. Specifically, noise faults should not impact models that use mean of sensor signals as the features, and offset faults should not impact models that use the variance of the sensor signals as features. Furthermore, this study has demonstrated that more-complex ensemble techniques, such as the random forest, are more robust to sensor faults than simple models, such as a classification tree. This work suggests that larger feature sets and more complex models may be preferred when sensor faults are a concern. This could be contrary to work that focuses on reducing feature sets and simplifying models to save cost and deploy at the edge. Future work should focus on specific ML methods that account for sensor faults during training, such as the probabilistic random forest (Reis, Baron, & Shahaf, 2018). Furthermore, future work could expand the types of faults and feature selection methods that are studied.

ACKNOWLEDGMENT

This material is based upon work supported by the Naval Sea Systems Command under Contract No. N68335-22-C-0213. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Naval Sea Systems Command.

REFERENCES

Adams, S., Beling, P. A., Farinholt, K., Brown, N., Polter, S., & Dong, Q. (2016). Condition based monitoring for a hydraulic actuator. In *Annual Conference of the PHM Society* (Vol. 8).

Adams, S., Beling, P. A., Greenspan, S., Velez-Rojas, M., & Mankovski, S. (2018). Model-based trust assessment for internet of things networks. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (pp. 1838–1843).

Adams, S., Cody, T., Beling, P. A., Polter, S., & Farinholt, K. (2020). Hierarchical classification for unknown faults. In *2020 IEEE International Conference on Prognostics and Health Management (ICPHM)* (pp. 1–8).

Adams, S., Meekins, R., Beling, P. A., Farinholt, K., Brown, N., Polter, S., & Dong, Q. (2017). A comparison of feature selection and feature extraction techniques for condition monitoring of a hydraulic actuator. In *Annual Conference of the PHM Society* (Vol. 9).

Adams, S., Meekins, R., Beling, P. A., Farinholt, K., Brown, N., Polter, S., & Dong, Q. (2019). Hierarchical fault classification for resource constrained systems. *Mechanical Systems and Signal Processing*, *134*, 106266.

Alzraiee, A. H., & Niswonger, R. G. (2024). A probabilistic approach to training machine learning models using noisy data. *Environmental Modelling & Software*, *179*, 106133.

Argawal, R., Kalel, D., Harshit, M., Domnic, A. D., & Singh, R. R. (2021). Sensor fault detection using machine learning technique for automobile drive applications. In *2021 National Power Electronics Conference (NPEC)* (pp. 1–6).

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Cody, T., Adams, S., Beling, P. A., Polter, S., Farinholt, K., Hipwell, N., ... Meekins, R. (2019). Transferring random samples in actuator systems for binary damage detection. In *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)* (pp. 1–7).

de Silva, B. M., Callaham, J., Jonker, J., Goebel, N., Klemisch, J., McDonald, D., ... Aravkin, A. Y. (2020). Physics-informed machine learning for sensor fault detection with flight test data. *arXiv preprint arXiv:2006.13380*.

Fink, O., Wang, Q., Svensen, M., Dersin, P., Lee, W.-J., & Ducoffe, M. (2020). Potential, challenges and future directions for deep learning in prognostics and health management applications. *Engineering Applications of Artificial Intelligence*, *92*, 103678.

Gupta, S., & Gupta, A. (2019). Dealing with noise problem in

machine learning data-sets: A systematic review. *Procedia Computer Science*, *161*, 466–474.

Helwig, N., Pignanelli, E., & Schtze, A. (2015). *Condition monitoring of hydraulic systems.* UCI Machine Learning Repository. (DOI: https://doi.org/10.24432/C5CW21)

Helwig, N., Pignanelli, E., & Schütze, A. (2015a). Condition monitoring of a complex hydraulic system using multivariate statistics. In *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings* (pp. 210–215).

Helwig, N., Pignanelli, E., & Schütze, A. (2015b). Detecting and compensating sensor faults in a hydraulic condition monitoring system. *Proceedings SENSOR 2015*, 641–646.

Ibrahim, A., Eltawil, A., Na, Y., & El-Tawil, S. (2019). A machine learning approach for structural health monitoring using noisy data sets. *IEEE Transactions on Automation Science and Engineering*, *17*(2), 900–908.

Liu, J., Adams, S., & Beling, P. A. (2020). An ensemble trust scoring method for internet of things sensor networks. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)* (pp. 1–6).

Machado, G. R., Silva, E., & Goldschmidt, R. R. (2021). Adversarial machine learning in image classification: A survey toward the defender's perspective. *ACM Computing Surveys (CSUR)*, *55*(1), 1–38.

Martakis, P., Movsessian, A., Reuland, Y., Pai, S. G., Quqa, S., Garcia Cava, D., … Chatzi, E. (2021). A semi-supervised interpretable machine learning framework for sensor fault detection. *Smart Struct. Syst. Int. J*, *29*, 251–266.

Meekins, R., Adams, S., Beling, P. A., Farinholt, K., Hipwell, N., Chaudhry, A., … Dong, Q. (2018). Cost-sensitive classifier selection when there is additional cost information. In *International Workshop on Cost-Sensitive Learning* (pp. 17–30).

Meekins, R., Adams, S., Farinholt, K., Polter, S., & Beling, P. A. (2020). ROC with cost pareto frontier feature selection using search methods. *Data-Enabled Discovery and Applications*, *4*, 1–13.

Mohapatra, D., Subudhi, B., & Daniel, R. (2020). Real-time sensor fault detection in tokamak using different machine learning algorithms. *Fusion Engineering and Design*, *151*, 111401.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Poulinakis, K., Drikakis, D., Kokkinakis, I. W., & Spottswood, S. M. (2023). Machine-learning methods on noisy and sparse data. *Mathematics*, *11*(1), 236.

Reis, I., Baron, D., & Shahaf, S. (2018). Probabilistic random forest: A machine learning algorithm for noisy data sets. *The Astronomical Journal*, *157*(1), 16.

Rezaeianjouybari, B., & Shang, Y. (2020). Deep learning for prognostics and health management: State of the art, challenges, and opportunities. *Measurement*, *163*, 107929.

Rokach, L., & Maimon, O. (2010). Classification trees. In *Data mining and knowledge discovery handbook* (pp. 149–174). Springer.

Schneider, T., Helwig, N., & Schütze, A. (2017). Automatic feature extraction and selection for classification of cyclical time series data. *tm-Technisches Messen*, *84*(3), 198–206.

Wang, X., Li, J., Kuang, X., Tan, Y.-a., & Li, J. (2019). The security of machine learning in an adversarial setting: A survey. *Journal of Parallel and Distributed Computing*, *130*, 12–23.

Wiyatno, R. R., Xu, A., Dia, O., & De Berker, A. (2019). Adversarial examples in modern machine learning: A review. *arXiv preprint arXiv:1911.05268*.

Zhang, L., Lin, J., Liu, B., Zhang, Z., Yan, X., & Wei, M. (2019). A review on deep learning applications in prognostics and health management. *IEEE Access*, *7*, 162415–162438.