Design and Implementation of a Model Selection Pipeline for Prognostics and Health Management in the Operational Environment

Peter Bishay[1], Sarah Lukens[2], Damon Rousis[3], Nathan Danneman[4],

[1,2,3,4] *LMI, Tysons, VA, 22102, USA*
peter.bishay@lmi.org
sarah.lukens@lmi.org
damon.rousis@lmi.org
nathan.danneman@lmi.org

ABSTRACT

Model selection is a crucial aspect of Prognostics and Health Management (PHM). However, many PHM models are developed for specific data sets and lack flexibility to adapt to different data sets with varying data quality considerations. To address this gap, we propose a generalizable model selection pipeline for PHM. Our approach involves creating a pipeline for testing models that users can tune in various ways. We designed a sequential pipeline of steps for model selection with a focus on implementation considerations which include recommendations for handling environmental variables, capabilities for remote and local work environments, and storage considerations of the serialized pipeline. Performance metrics are designed to consider data quality characteristics such as ambiguous labeling. We illustrate the generalizability of our approach through a case study of our model selection pipeline applied to a field data set with ambiguous labels. Our design accommodates data characteristics commonly found in field data, such as ambiguous labels and data wrangling. Our contribution fills a gap in real-world implementations of PHM by offering technology considerations and recommendations for effective deployment.

1. INTRODUCTION

The components of implementing a PHM system include data collection of the sensor variables as well as other contextual data sources, development of PHM models, validation, and designing policy around outputs of those models. In theory, a PHM program should result from a formalized process based on risk-assessments that consider the business case, the stakeholders and the physical system. Thanks to recent advancements in technology such as faster data processing and tools for advanced analytics, there are now additional opportunities for purposing existing field data which may not have been collected for a PHM program in mind for a PHM program. This data may include sensor data collected during operations or data from preexisting condition-based programs. Use of such data may lead to challenges related to the structure or quality of the data which may impact the suitability for PHM tasks.

From a modeling perspective, a PHM system will incorporate one or more of three main PHM tasks, which are fault detection, fault diagnosis and prognostics. Fault detection is the detection of abnormal system behavior. Fault diagnosis is the separation of the various failure modes of the system and their classification into known classes. Prognostics (degradation) are predictions about the evolution of system health to return equipment lifetime predictions such as Remaining Useful Life (RUL).

The different data characteristics, physical system characteristics and business needs points to many decision variables in attempting to develop a PHM model on field data. As a result, many PHM models are developed for specific data sets based on its specific characteristics. In this work, we describe a flexible and generalizable pipeline for testing and selecting models for PHM. Our contribution is an end-to-end, data specific engineering process for ingesting unseen sensor and maintenance data (of any quality). The end result is a flexible, modular framework with components that consider data quality analysis, suitability metrics and model experimentation, developed in a way that the final model is deployable for use.

The framework is tailored to handle the immense volume and complexity of sensor data, data quality challenges, and non-

integrated contextual data sources needed for labeling using the following properties:

- Data-driven and generalizable enabling rapid scaling to multiple systems
- Re-usable, repeatable, and extensible code libraries developed in both native Python and Apache Spark when sensor data volume requires scalable computation.
- Incorporates a full data quality framework which identifies, measures, and applies data conditioning where appropriate.
- Integrated and standardized data model which includes contextual data from various sources and enables extensibility and repeatability.

The rest of the paper is organized as follows. Section 2 provides background information on model development and deployment frameworks, presenting contextual definitions and a review of existing literature. Section 3 outlines the methodology used for the model testing evaluation process. In Section 4, an illustrative case study is presented to exemplify the process. Sections 5 and 6 delve into the discussion and conclusion, respectively, summarizing the findings and offering insights for future work.

2. BACKGROUND

The framework can be thought of as modular with a high level structure each split into sub-modules containing different operations specific for sensor based algorithms (SBA). We call The high-level foundational structure a Prognostics and Health Management Framework (PHMF), which is composed of data operations, model operations, and operationalization (Figure 1).

For model development on a new data set, the **data operations** phase contains tools for formal and repeatable data wrangling as well as bringing scientific rigor to the exploratory data analysis of sensor data. The primary purpose of this process is to draw conclusions regarding the suitability of data for modeling and to identify the requirements for model pre-processing.

The **model operations** phase covers the training, validating, storing and analysis of models. The data feasibility assessment results help specify the data conditioning and modeling task as well as the design for the modeling experiment. Model performance metrics and validation tools are also included as well as fine tuning and analysis of modeling experiments. The outputs of the model operations stage are model inputs, outputs, and storage requirements.

Operationalization focuses on transitioning from model training to prediction while ensuring scalability and robustness of the deployed model. The rest of this section summarizes related work, organized using these three major phases.

2.1. Design for PHM and PHM Frameworks

There are many complimentary PHM design and implementation frameworks. From the PHM lifecycle perspective, one perspective is DE^3, which is Design, Development and Decision (Hu, Miao, Si, Pan, & Zio, 2022). The design portion of the lifecycle includes requirement analysis, framework design, and verification and validation. Guidelines for requirements specifications for a prognostics initiative which integrates safety, reliability, cost and real-time viability are found in (Saxena et al., 2010; Goebel et al., 2017; Walker & Kapadia, 2009). In terms of frameworks which integrate model development and operations, (Guo, Bao, Wu, Jin, & Lee, 2019) summarize the application of DevOps from software development as a framework for managing the lifecycle of prognostic models, using goal of reducing the time it takes to bring industrial AI applications to market while effectively addressing real world uncertainties. A framework for PHM which addressed model design, development and implementation focusing specifically on considering relationship between data availability and PHM tasks is presented applied to hydrogen storage systems (Corrêa-Jullian, Camila and Groth, Katrina M, 2022). The scope of this paper is around the data engineering and model development component of the full system, with focus on use cases interested in leveraging existing data.

2.2. Data Operations

We further divide Data Operations into four major steps. The first step is **transmittal and storage**, which evaluates data characteristics such as scope, data sources, access, mode of transmittal (batch or streaming), size, storage medium. Completion checks of this stage test for sufficient depth and breadth of the data such as specific checklist items for specific data sources (Lukens, Rousis, Baer, Lujan, & Smith, 2022).

Exploration is primarily concerned with surveying the different data sources to assess the data structure and evaluating different fields data completeness, for readability, identifying join keys (if they exist) and validating the join keys between different data sources. Variable selection, through both data analysis metrics and through surveying subject matter expertise may occur here, which can reduce variables from hundreds to a more manageable number (Griffiths, Corrêa, Hodkiewicz, & Polpo, 2022; Cofre-Martel, Lopez Droguett, & Modarres, 2021). For time series data, this phase also includes determining sampling frequency and how to summarize information if the raw sampling frequency is more frequent (descriptive statistic selection) or less frequent (interpolation/inference) (Cofre-Martel et al., 2021). One component of this phase is identifying and implementing any historical data quality improvement approach such as imputation for time series data sources or Technical Language Process-

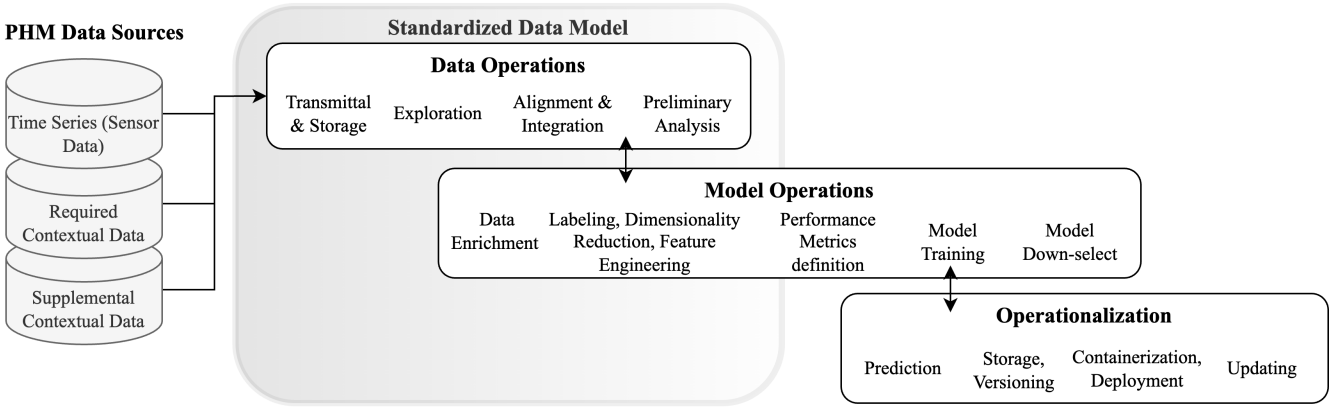


Figure 1. PHMF Operations Framework

ing (TLP) (Brundage, Sexton, Hodkiewicz, Dima, & Lukens, 2021) for transactional data sources.

The third phase is **alignment and integration**. Alignment refers to aligning different sensor variables at the same time stamps. Griffiths, Corrêa, Hodkiewicz, and Polpo (2022) recommend approaches for aligning and merging different data sources. These models include matching the frequencies of time-readings, arranging the sensor data in wide form, aligning with contextual information, and preparing for fault detection labeling. The fourth phase is **preliminary analysis**, which is concerned with preparation of the data for modeling, which includes feature preparation and data reduction techniques. Feature preparation includes generating features, such as preparing lagged data. Feature selection includes data reduction techniques (Griffiths et al., 2022; Cofre-Martel et al., 2021). Feature extraction techniques are covered in many places as a key area for data pre-processing in a PHM model development pipeline (Atamuradov, Medjaher, Dersin, Lamoureux, & Zerhouni, 2017; Elattar, Elminir, & Riad, 2016), and have different considerations when deep learning models are introduced (Fink et al., 2020).

The ultimate design of the model for a PHM system depends on many factors ranging from data characteristics, physical properties of the system from which the data was collected to organizational objectives. For any given PHM task, the data must meet certain required properties for an effective model. A **data suitability assessment** refers to assessing the suitability of a prepared data set for a PHM task. In the literature, Chen, Zhu and Lee (2012) developed a methodology for evaluating the quality of training data for PHM model development, which measures suitability first for fault detection, then diagnostics then remaining useful life (RUL) prediction (Chen, 2012; Chen, Zhu, & Lee, 2013). Metrics for detectability (suitability for fault detection) and diagnosability were proposed using Maximum Mean Discrepancy, a kernel-based approach for measuring the distance between multivariate distributions (Jia, Zhao, Di, Yang, & Lee, 2017). Coble and Hines recommend suitability metrics or prognostics models based on three key qualities for prognostics parameters: monotonicity, prognosability and trendability (Coble & Hines, 2009). Calculation of these metrics helps not only identify the suitability of data, but also helps in fitting a prognostic parameters through methods such as regression.

A summary of different assessment measuring approaches for the different PHM tasks are summarized in Table 1. The data must not only exhibit required properties for an intended task, but meet additional requirements on data collection and data labeling processes. In order to train a fault detection or anomaly detection model, the data must be labeled as healthy or unhealthy. A diagnostic model requires labels based on different possible failure modes or patterns. A prognostics model requires some measure of remaining useful life. The results of this assessment are the inputs to the model operations phase for determining a model.

2.3. Model Operations

Solutions to PHM problems typically fall into two categories: physics-based and data-driven. Our approach is data-driven modeling, which can be generalized to other systems via transfer learning and do not require expert knowledge of the physical processes or underlying mechanisms governing their operations. Different modeling approaches for different PHM tasks are covered in PHM textbooks such as (Vachtsevanos, Lewis, Roemer, Hess, & Wu, 2006; Goebel et al., 2017). Recent review articles PHM modeling methodologies include reviews of the full model space (Jimenez, Schwartz, Vingerhoeds, Grabot, & Salaün, 2020) and reviews specific to deep learning approaches which has shown great performance and flexibility in the past decade (Rezaeianjouybari & Shang, 2020; Y. Wang, Zhao, & Addepalli, 2020; Zhao et al., 2019).

Model operations pipelines for the development and deployment of data-driven models, which specifically consider data quality challenges observed in field data have recently been

Table 1. Summary of metrics for assessing a PHM dataset for suitability for a given PHM task. The data must not only exhibit required properties for an intended task, but meet additional requirements on data collection and data labeling processes.

PHM Task	Suitability Criteria	Labeling Requirements	Data Requirements
Fault Detection	Detectability: equipment abnormality can be detected as an outlier or as a different distribution from normal	Binary labels such as faulty or healthy state	Anomalies must be detectable from normal conditions. Must have sufficient training data to cover all possible states.
Diagnostics	Diagnosability: a clear decision boundary can be obtained to classify multiple failure modes	Multiple labels based on different failure modes	Data must be separable within the different desired prediction classes.
Prognostics	Trendability: related to machine degradation, which is typically slow and monotonic.	Equipment or component lifetimes	Methods for assessing RUL are typically based on assumption of monotonic fault progression. Proposed metrics for determining the suitability of a "trajectory", and data must contain some indicator of lifetime.

developed and are of practical importance. Common data quality challenges with sensor data include data of insufficient volume to fully capture system complexity and class imbalance, both which lead to the challenge of generalizing poorly. Modeling approaches for fault diagnosis when the data is small and imbalanced are reviewed in (Zhang et al., 2022).

Another challenge in field data is around ambiguous labeling. One specific pipeline was developed for fault classification of sand rakes system in mineral processing, where each stage of the model experiment pipeline was well-defined with a set of experiments across different labeling strategies (Corrêa et al., 2022). Another framework addressing the ambiguous labeling in field data collecting in a mining processing line used a rule-based approach for labeling health states based on when the system was stopped and the failure mode at the time of stoppage (Cofre-Martel, Sergio and Corrêa-Jullian, Camila and López Droguett, E and Groth, Katrina M and Modarres, MM, 2021). A framework was developed and implemented where model experiments could be run and compared for different hyperparameters on the stoppage windows on the labels. Experiments with model performance with the removal of data outliers in field data for wind turbine SCADA data in (Marti-Puig, Blanco-M, Cárdenas, Cusidó, & Solé-Casals, 2018).

2.4. Operationalization of PHM models.

Operationalization of models transitions the model from training to prediction while ensuring scalability and robustness, and can follow machine learning operations process flow. Considerations for operationalization of PHM models include mechanisms in place for monitoring model drift, continuous model updating, mechanisms and interface for decision support(Guo et al., 2019). Additional considerations around model storage, containerization and incorporation to front-end interfaces are here as well.

3. METHODOLOGY

In this section, we outline the key steps and implementation details involved in model operations, which encompass testing and development for the purpose of model operationalization. Through pipeline development and serialization, this framework allows for more structured model operations.

3.1. Requirements definition

We first define the following requirements for the development of a toolkit for PHM model development following the framework.

1. Single tool (application) which can perform all or most functions related to performing exploratory analysis, model development and model deployment.
2. Handle field data with varying quality and ambiguous labeling.
3. Allow data (in any stage of processing) and saved models to be accessible to multiple individuals for collaboration.
4. Tune multiple types of decisions related to data preprocessing and model training.
5. Load and overwrite (processed) data remotely.
6. Configure variables to prepare data for staging.
7. Serialization of pipelines for repeated experimentation and models for implementation.

The framework is designed and implemented such that as new data and tasks are encountered, modules can be added at various points. This paper specifically focuses on the stages of the Model Operations phase. In any modeling pipeline, there are several, often intersecting, choices to be made on how data

is included, split, normalized, modeled, and labeled. Overall, the design space – that is, the total count of the number of possible variations – is massive, number, in conservative cases, in the billions. SBA Tools was developed in order to be flexible in pipeline design.

3.2. SBA Tools Framework

As each module of our framework for handling sensor (and other relevant contextual) data for PHM applications handles sensor-based algorithms (SBA), we call it "SBA Tools" denoting the collection of different capabilities. The SBA Tools Framework spans exploratory analysis, model development, and model deployment.

The framework effectively handles various qualities of field data and addresses challenges in labeling clarity. Additionally, it promotes cross-collaboration by integrating with a shared remote environment and supports the serialization of pipelines and models. This serialization facilitates practical implementation and repetitive experimentation. Each stage in the pipeline has various inputs and decisions that can be customized according to specific requirements. Options for stages specific to Model Operations include the following:

Labeling: This stage defines the rules for labeling training data. Depending on the task, this may include labeling healthy and unhealthy states (fault detection task) or remaining useful life (prognostics task) on historical training data instances. A specific column and function are utilized to create these labeling rules.

Interval Filter: By "Interval", we refer to a collection of data observations segmented by a time window, such as a collection of time series of readings of one component or system from healthy to failed. The interval filter stage removes categories in the data (such as time intervals) if they do not meet a certain requirement such as a specified minimum number of observations required in an interval.

Test-Train-Validation: In this stage, the data is divided into test, train, and validation sets. Users have the flexibility to specify the split based on an attribute, ensuring that attributes of the same type are grouped together.

Modeling: Modeling is the model training and selection stage. Users can choose from a range of available models. Hyperparameters can also be specified in this stage to fine-tune the models. In addition, a subset of the sensor variables can be selected to train and predict on for feature selection testing.

Performance: This stage enables users to assess the performance of the models. Performance metrics can be averaged at different levels, such as across time intervals or split groups. Users can select the desired performance metrics they wish to analyze and retrieve.

3.3. SBA Tools Implementation

We developed a toolkit instance in Python where objects are created for different types of components within the overarching pipeline containing methods that allow users to extract information, load information, and perform operations. While the toolkit itself is proprietary, in this section we outline the general design and architecture so the approach may be reproducible.

The toolkit is implemented as a library of tools which simplifies the end-to-end stages of the PHM data-driven modeling process, leaving a data scientist to spend the most capacity on model development and abstracting away many engineering and Continuous Integration and Continuous Delivery (CI/CD) tasks. The tool is implemented with a simple Application Programming Interfaces (API's) with which to save models so that they may be later loaded by the build machine and incorporated into prediction pipelines.

This implementation approach includes a thin wrapper around the most common machine learning (and anomaly detection) algorithms, developed by third parties and open-sourced, with-out having to learn the specifics of each library. Additionally, the team can work collaboratively by saving not just model building scripts that normally version controlled, but data, models, and pipelines to centralized locations. Any updates to data conditioning, transforming, or stages of a pipeline that required high computation cost, could be performed once then saved and re-used in later stages model building.

A configuration file utilizes a key-value format to specify data fields as time columns, interval identifiers, inputs, and target variables. The framework components are structured as classes and subclasses, namely object-related and process-related classes. Object-related classes are serializable and encompass storage capabilities, while process-related classes accept data set instances as input and enact specific processes.

The primary class, termed `SBAToolsObject`, serves as the core and facilitates serialization and storage of essential class objects: `Pipeline` and `DataSet`. These objects are stored locally based on the home directory and remotely through SSH, using environment variables `SBA_TOOLS_USER` and `SBA_TOOLS_HOST`.

The data set class can be initialized or loaded from local/remote sources and subsequently used as input for the pipeline. It inherits methods from SBAToolsObject and introduces functions for sampling and data retrieval. Underlying the data set object is a pandas dataframe.

The Pipeline object, which can be initialized or loaded from a local/remote location, enables the insertion of stage classes. These stages execute sequentially, encompassing data preprocessing, labeling, data division for training/testing, applica-

tion of a modeling layer, and the computation of performance metrics. Serializing this objects allows it to be treated as a full queue of processes which can be easily rerun on other data sets.

The final primary class, Stage, acts as a parent for all stages. It furnishes methods to configure stage-specific seeds and an execution method to enact processes.

All remaining class objects inherit from the Stage class, facilitating their integration into a pipeline. This arrangement allows for consecutive application of multiple processes, serving data preprocessing and model development purposes. Parameter specifics differ for each stage:

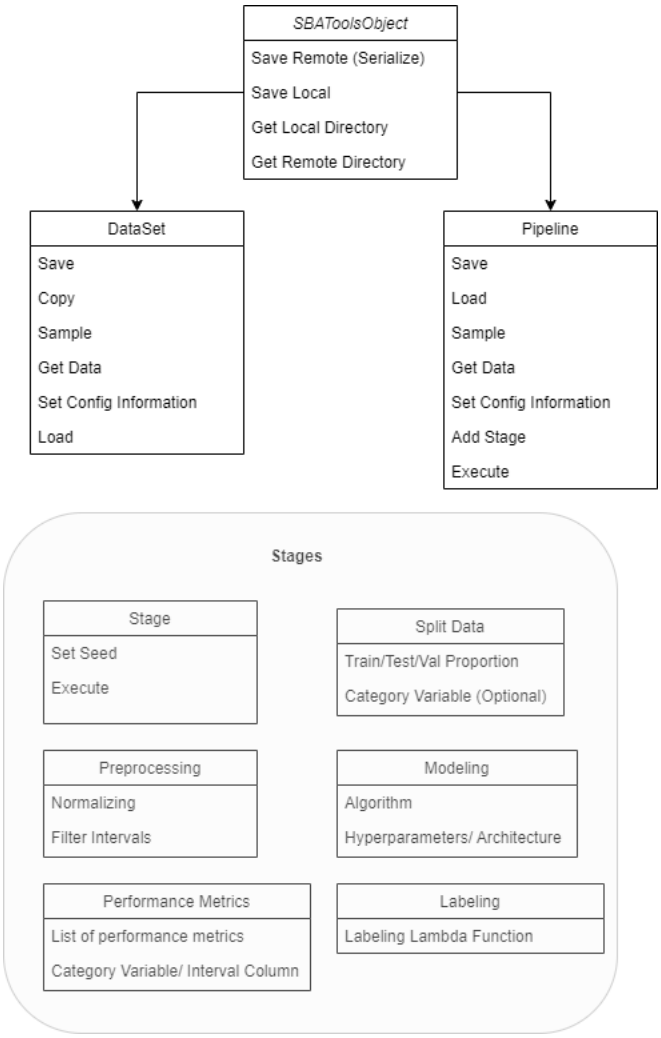
The modeling layer accepts algorithm input from a predefined model list, including scikit-learn classifiers (e.g., logistic, random forest, SVC), as well as alternatives like AAKR and Autoencoders utilizing keras. Hyperparameters for the algorithms are additional inputs.

Other stages include:

- Labeling stage, which incorporates a labeling rule as input.
- Normalizing stage, requiring a normalization method and category column for operation.
- Splitting stage, dividing data into train, test, and validation sets, with seed values and proportions as parameters.
- Performance stage, tailored to interval or aggregate levels, accepting a list of metrics as input.

Figure 2 provides additional details on the class objects, subclasses, and their methods:

Figure 2. Class Object Organization and Methods. The relationship of class object between each other and their methods are shown here.



4. CASE STUDY

Development of the model building functionality of the SBA Tools framework was motivated by a project which entailed identification and development of a model using sensor data for a specific asset which could be used for PHM applications. The sensor data was field data which was already being collected for process control purposes. Identification of specifically what PHM tasks and which data would be useful was part of the project requirements.

Data. The data set contains time-series measurements collected on a system for power generation as well as relevant maintenance history logs. The data set contains several subsystems of a larger asset with different time series collections each corresponding to a unique maintenance event. The sensor data was collected before and after each maintenance event. The volume of data available for each event varies depending on the system's data availability during the fault period, but typically spans 60 days prior to and 60 days following the repair. We call each collection of readings associated with a single event an *interval*. The data set includes categorical variables such as equipment identifier, reading timestamp and corresponding sensor readings for the component. The final data set contained around 125 intervals spanning about 100,000 total sensor observations.

Due to the proprietary nature of the data, while we report actual observed data characteristics and data quality challenge, an open source data set is used to report numerical results of the pipeline. The model testing pipeline was reproduced using C-MAPSS, NASA's Aircraft Engine Simulator data, which provides data on the degradation of engine quality across multiple engines (Saxena, Goebel, Simon, & Eklund, 2008). The data set consists of 21 sensor values, an indicator for engine/time intervals, and time information. For simplicity, the most basic C-MAPSS data subset (FD001) is used which has 100 units, one fault mode and one operating condition. Use of an open source data set also promotes reproducibility of our results.

4.1. Data Operations

The Data Operations phase of the framework includes a data feasibility assessment stage which determines modeling decisions to be made. For the Case Study, the Data suitability metrics indicated that fault detection was the most promising PHM task on this data, as there were no observable degradation signatures but some indicator of detectability between the data before and after a significant event. Consequently, the task was defined to distinguish between healthy and unhealthy component states by leveraging supervised training techniques to identify patterns in sensor data.

There were three data quality challenges identified which influenced the model development experiment, reported in Table 2 along with our identified mitigations. The first major challenge was that the labels on the sensor data intervals were ambiguous (label when the fault occurs). To address this challenge, we added the exploration of various labeling approaches for partially or fully labeling the data and tested their effectiveness to our model testing and selection pipeline. For instance, we experimented with treating all data pre-repair as unhealthy and post-repair as healthy. However, this approach could be problematic, as the component may be in a healthy state at the beginning of the analysis and then deteriorate closer to the repair date. Therefore, we also explored other labeling techniques such as segmenting the data into different time intervals based on features such as the type of fault. For example, we considered labeling a specified number of days before a repair as unhealthy and evaluated this approach for different fault types. We recognize that some faults may be gradual, while others can be more abrupt, which we tested for to find better suited labeling methods for different fault types.

The second challenge was that there was low data volume for fault detection model. Anomaly detection algorithms traditionally require a sufficient amount of observations capturing all possibly "normal" states, which may not completely be present but this data was collected 60 days before and 60 days after a significant event. Two design decisions were made here: first, to use not only anomaly detection algorithms, which are trained on healthy data alone, but to also look at classification models with two states in order to use all of the data.

The third challenge driving our model pipeline was class imbalance between intervals. To account for this, we designed performance metrics and our experiments to both measure the variability from the different class sizes and to look at the intervals individually.

Fault Detection Modeling Formulation. Associative and classification models were both explored for anomaly detection. Both modeling approaches take pre-processed data (normalized, labeled, etc.) apply a trained fault detection model, which returns a real value. This real value is then applied to a decision function, which returns a model prediction of healthy or faulty based on applying some sort of threshold to the fault detection model output.

Associative models, such as autoencoder or auto-associative kernel regression (AAKR), are trained on data under normal conditions and where the model output is the prediction of the input under normal conditions. In this case, the real-valued output of the fault detection model comes from a residual function which is the difference between the actual reading and the predicted reading under healthy conditions.

Classification models make predictions through categorizing input data into predefined classes or labels, here the labels being "normal" or "fault". In this case, the real-valued output of the fault detection model can be viewed as the probability of membership to the normal class.

Measuring Model performance. We use balanced accuracy (BA) as the key performance metric, recommended for addressing class imbalance for fault detection algorithms in (Corrêa et al., 2022). BA is defined as the average of the true positive rate (TPR), which can be thought of as the correct prediction rate on the healthy data, and the true negative rate (TNR), which can be thought of as the correct prediction rate on the unhealthy data ($BA = (TPR + TNR)/2$). BA is a useful indicator that high accuracy is not because the model is predicting everything to be in one class.

As our class imbalance is more across the different intervals (rather than having exceedingly more healthy observations than unhealthy), we compare two forms of BA. To determine the balanced accuracy at the interval level (which will be referred to as "Interval Level"), each time interval is considered a fault interval if it contains at least one unhealthy state. For a time interval to be accurately labeled as a fault interval, the model must correctly predict the presence of an unhealthy state at least once during that interval.

For the averaged balanced accuracy (which will be referred to as "Averaged Intervals"), the balanced accuracy calculations are performed within individual intervals, and then the results are averaged together to provide an overall assessment. The last measure used was the first time of detected fault in each interval. This measure requires no label and is useful for comparing two models in terms of which model can predict earlier and by how much.

Test/Train/Validation split. Test-train-validation sets were selected by splitting different intervals (over selecting random time points) and were split at 70% of intervals for the training set, 20% of intervals for testing with 10% of intervals held out for validation. Seeds were set to keep the splits consistent across runs.

Adjustments to the open source data set. Unlike the pro-

Table 2. Data quality challenges identified from data suitability assessment. Decisions for design on the model selection and testing pipeline were informed by these identified data characteristics.

Data Quality Challenge	Mitigation
Low data volume for anomaly detection	handle the "normal" data in aggregate Include classification models which use both the healthy and the faulty data
Class imbalance among the different intervals	Use and compare performance metrics both across and within the different intervals Perform cross-validation experiments which accounts for different intervals to understand sensitivity of interval size on model response
Ambiguous labeling	Design and include performance metrics which account for ambiguous labels Perform experiments which accounts for different labeling strategies to understand sensitivity of labels on model response

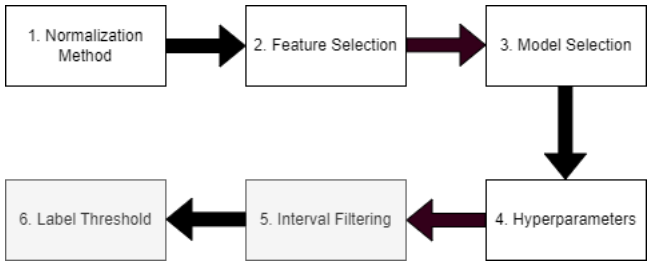
prietary data set, the C-MAPSS data does not have class imbalance issues between the different intervals. However, for a fault detection task the data does not have labels as to when a fault is initiated (the data set is labeled for the remaining useful life). Therefore, the ambiguous labeling data quality challenge is also relevant for applying fault detection algorithm to the C-MAPSS data set. We adjust the initial labeling experiments as follows. For each time interval, the earliest 10% of data is treated as healthy and the latest 10% of data is treated as unhealthy because the engine continuously degrades in quality. This creates an expected distinction between healthy and unhealthy for model exploratory and reporting purposes and later in the pipeline different labeling choices will be explored.

4.2. Experiment design for Model operations

For the case study data, it was determined to use a greedy, step-wise heuristic for a fault detection modeling pipeline which for model selection. Some plausible settings "downstream" in the modeling pipeline are fixed, leveraging the best settings found in "up-stream" portions. With all that fixed, we identify the optimal setting in the portion of the pipeline under inspection akin to "per axis" optimization. While there is no guarantee of optimality coming out of a process implemented in this way, we retrospectively tested a large array of "off-path" pipelines and none compared favorably with the best pipelines found via this method, giving us confidence that we found, at least, a strong local maximum.

The different modeling and pre-processing decisions tested against multiple performance metrics to find optimal solutions for better performing models, allowing for making different decisions in iterations for both performance and addressing data challenges observed. The step-wise, iterative approach was used to address the massive optimization for the case study is shown in Figure 3. These stages are broken into two categories: stages which are common to general model development and validation, and stages which are specific to an observed data set or task.

Figure 3. Step-wise iterative testing approach. Best performing decision continues through the remaining tests. Stages colored gray indicates stages specific tailored to tackling the observed data quality challenges.



1. Normalization: identify normalization scheme for modeling.
2. Feature Selection: identify feature importance of the sensor variables
3. Model Assessment: identify a high-level idea of which families or types of models perform better or worse. The output are candidate modeling approaches for hyperparameter tuning.
4. Hyperparameter Tuning: turning the shortlist of models identified in the previous step.
5. **Specific to the data:** Filtering Time-intervals Experiment: Experiment for gauging the impact of class imbalance between the different intervals on model performance.
6. **Specific to the data:** Labeling Experiment: Experiment for gauging the sensitivity of the model performance on the choice of label.

4.3. General Model Development Stages

Normalization. It is possible to experiment with different normalization approaches as well as different grouping categories for normalization. Here, z-scores were calculated at every data point. Different grouping categories (populations where mean and standard deviations were calculated for normalization) include "All data" which refers to normalization by the mean and standard deviation across all of the data,

Table 3. Normalization Test Results

# Rows Filtered	BA (Interval Level)	BA (Averaged Intervals)	Time After Failure
all data	0.998	0.994	2 days 5 Hours
within time interval	0.998	0.996	2 days 08 Hours

Table 4. Performance for feature selection exercise of 21 sensor variables for the C-MAPSS data.

# Rows Filtered	BA (Interval Level)	BA (Averaged Intervals)	Time After Failure
21	0.996	0.992	2 days 09 Hours
14	0.998	0.996	2 days 06 Hours
12	0.997	0.994	2 days 10 Hours
8	0.998	0.996	2 days 07 Hours
5	0.998	0.998	2 days 07 Hours
3	0.995	0.994	-7 days 04 Hours
1	0.979	0.947	-38 days 10 Hours

"within time interval" which refers to normalization by each interval independently and normalization by asset or system (which may span several intervals). As the C-MAPSS data does not have engine specific identification to calculate, results for the first two normalization approaches are shown in Table 3. For the C-MAPSS test case, both normalization schemes provide very good results, with slightly improved performance for the "within interval" category. The outcome of this pipeline stage is the use of normalization within an individual interval for the duration of the pipeline.

Feature Selection This testing process involved using a random subset of the sensor variables to be used for testing for each iteration. Table 4 shows results for a variable number of randomly selected sensors from the possible 21 sensors available. For the C-MAPSS data, 5-8 sensors appear to be optimal combination, which is consistent with other studies on this data source. For example, seven were identified in (T. Wang, Yu, Siegel, & Lee, 2008) and 11 identified in (Coble & Hines, 2011)).

Model Assessment Model assessment uses a variety of both associative and classification models for anomaly detection. Seven different models were tested which included autoencoder (using The Tensorflow Karas package; (Abadi et al., 2015)), logistic regression, support vector regression, random forest (using the Sklearn package; (Pedregosa et al., 2011)), aakr (using the aakr package; (Myrberg, 2022)), pulearner (using the pulearn package; citation), and aggregation based models which treat multiple sensor readings as single predictions.

The performance results for the model assessment on the C-MAPSS data are shown in Table 5 for model selection. From the C-MAPSS data, all of the models performed extremely well with the exception of the Autoencoder. Due to our systematic, step-wise iterative testing, the Autoencoder did not continue through testing to the hyperparameter step. We believe that the starting architecture for the model is too simple and the model is therefore unable to predict well. Through tuning the model's architecture and hyperparameters, the Autoencoder would have a better performance outcome.

Table 5. Results on Model Assessment for the C-MAPSS data.

# Rows Filtered	BA (Interval Level)	BA (Averaged Intervals)	Time After Failure
Autoencoder	0.432	0.445	-141 days 1 Hour
Logistic	1.00	1.00	2 days 04:33:36
Random Forest	0.999	0.998	2 days 04 Hours
SVR	1.000	1.000	2 days 04 Hours
Multi-Row Predictor	0.996	0.993	2 days 09 Hours
PuLearner	1.00	1.00	2 days 04 Hours

We selected the random forest model with row aggregation because, while it had slightly lower performance than other types of model, the the row aggregation created more flexibility to additional parameters among models which all have similar and high performance. In the context of time series sensor readings, row aggregation proves to be a suitable approach as it allows for decisions based on multiple readings occurring around the same time, rather than relying solely on a single sensor reading to trigger alerts.

Hyperparameter Selection. Hyperparameter testing was performed on the random forest model using grid search to define a grid of parameters that encompassed a wide range of hyperparameter combinations while taking model run time into consideration. In situations where the reasonable parameters resulted in excessive run-times, we used a random search option. The hyperparameters were selected through pairing the random forest model with row aggregation. Specifically, three hyperparameters were varied for the random forest model on the C-MAPSS data set: number of rows aggregated, method of aggregation (mean, median, percentile) and number of decision trees. Under aggregation method, if percentile was the chosen aggregation method, the percentile value was additionally included.

C-MAPSS results for the hyperparameter testing stage are shown in Table 6. The test results shown are selected from the highest performing and lowest performing hyperparameter combinations for comparison. For the C-MAPSS data set, the random forest model already performed very well out of the box and all of the experiments also performed well. The most significant difference in results observed was a slight decrease in balanced accuracy and increase in detection time

for the fifth experiment results (five rows aggregated, median, 25 decision trees).

4.4. Tuning Stages Specific to Observed Data Quality Challenges

Other stages performed and tuned relate more closely to PHM and help address some of the data challenges shown on Table 2. Filtering Time-Interval Experiments addresses issues with data quality related to class imbalance by conditionally removing intervals which don't meet some criteria. In this experimentation, we used overall interval size as the condition.

Ambiguous labeling is also a challenge which benefits from testing different labeling strategies and thresholds related to those strategies. We observed results from labeling data as healthy and unhealthy using different thresholds for splits.

Filtering Time-intervals Experiment The experiments for gauging the impact of class imbalance between the different intervals on model performance was designed through excluding intervals from the test/train/validation set that did not meet a minimum requirement of rows (observations). The experiment varied the minimum row size. This was done to train the data on higher-quality intervals that may have more clearly show differences between healthy and unhealthy. However, there is a trade-off between having more data to train on and having longer, possibly better-quality intervals. Testing involved producing results for intervals with a minimum of 0, 30 and 50 rows.

C-MAPSS results are shown below for time-interval filtering in Table 7. For 0 drops, intervals remained at 100 and dropped to 94 and 17 for the 30 and 50 row filters. Results show a slight increase in balanced accuracy when averaging across intervals and slightly earlier detection time for dropping intervals with less than 30 rows. As we increase to 50 rows required, the amount of data drops significantly and balanced accuracy slightly drop. Detection time has a significant increase of nine days.

Labeling Experiment The experiment for gauging the sensitivity of the model performance on the choice of label was designed through testing different distributions of healthy and unhealthy intervals split from earliest 10% to latest 10% to a 50-50 split. The balanced accuracy values were found to be highly similar across the various labeling methods, both in the original case study and in the C-MAPSS data (shown in Table 8). Balanced accuracy usually dropped with the exception of a 30% split. First detection was also more delayed on average as the split increased. This is likely due to the nature of the degradation of the sensors. Data near the same time period is more likely to be similar, meaning that data at the opposite side of earliest 10% and latest 10% are more likely to be different compared to some of the data closer to the middle in a 50-50 split.

5. DISCUSSION

The utilization of SBA Tools proved advantageous in facilitating a streamlined testing process, allowing for the creation, tracking, and management of various components. Throughout the testing phase, we consistently incorporated additional elements that directly aligned with the case study, resulting in notable enhancements to the testing tool and the development of features specifically tailored for PHM modeling. Our experimentation involved running hundreds of iterations, each exploring different tuning decisions to discern optimal choices that significantly improved performance.

During the experimentation process, certain challenges emerged, particularly in relation to run times. Notably, the selection of one model type over another and the fine-tuning of hyperparameters for specific models (such as increasing the number of decision trees in a random forest or designing a more complex autoencoder architecture) often led to prolonged execution times. In assessing the effectiveness of these choices, we primarily relied on balanced accuracy and distance-based metrics, which gave insight into the average time of first alerts generated relative to the closing period.

6. CONCLUSION AND FUTURE WORK

The SBA Tools framework was developed through PHM applications aimed at the development and deployment of data-driven models using to historical maintenance and sensor data of physical systems. Our implementation approach allows users to upload and load data sets from a remote location or locally. Users can also identify key columns in the data set such as sensor, labeling, and interval columns. Through this tool, a user can select stages that are built into it and push the data set into pre-processing, model training/testing, and performance phases to prepare for model deployment.

The scope of this paper was focused on the design and implementation of the framework, and our case study supported an instance of the framework for illustrative purposes. Future work for the pipeline includes incorporation of additional steps to match similar pipelines in the literature to benchmark model performances. Benchmarking results using similar pipelines requires the addition of additional modules in the pipeline, such as incorporation of a sliding window approach in feature engineering following (Corrêa et al., 2022). More generally, formalization of benchmarking regarding model development pipelines which consider field data characteristics is an emerging and open research topic in the PHM community. Traditional benchmarks on open source data sets typically follow specific guidelines where the data is assumed clean and well structured. For example, benchmarks using CMAPSS data typically follow the performance baselines set by the PHM 2008 data challenge (Ramasso & Saxena, 2014).

Table 6. Hyperparameter Test Results. Explain here that there are 3 hyperparameters and what we're looking at

Case	Rows Aggregated	Aggregation Method	# Decision Trees	BA (Interval Level)	BA (Averaged Intervals)	Time After Failure
Top 1	# Rows Aggregated: 20	median	500	0.9964	0.9956	2 days 09 Hours
Top 2	# Rows Aggregated: 20	mean	100	0.9966	0.9978	2 days 09 Hours
Top 3	# Rows Aggregated: 50	median	500	0.9969	0.9978	2 days 08 Hours
Top 4	# Rows Aggregated: 5	median	100	0.9946	0.9934	2 days 10 Hours
Top 5	# Rows Aggregated: 10	median	25	0.9956	0.9934	2 days 10 Hours
Bottom 3	# Rows Aggregated: 50	percentile-100%	100	0.9968	0.9978	2 days 09 Hours
Bottom 2	# Rows Aggregated: 10	mean	500	0.9959	0.9978	2 days 10 Hours
Bottom 1	# Rows Aggregated: 20	percentile-25%	100	0.9966	0.9978	2 days 09 Hours

Table 7. Remove Time Intervals Based on Minimum Row Requirement

Min Row	# Intervals Included	BA (Interval Level)	BA (Averaged Intervals)	Time After Failure
0	100	0.997	0.990	2 days 08 Hours
30	94	0.997	0.995	2 days 06 Hours
50	17	0.985	0.991	11 days 07 Hours

Table 8. Labeling experiment

Labeling split	BA (Interval Level)	BA (Averaged Intervals)	Time After Failure
10%	0.982	0.989	3 days 07 Hours
20%	0.978	0.982	7 days 01 Hour
30%	0.985	0.990	9 days 10 Hours
40%	0.985	0.981	8 days 10 Hours
50%	0.943	0.945	14 days 12 Hours

ACKNOWLEDGMENT

The authors wish to acknowledge and thank Brian Tonge and Keith Rodgers for their support; Dylan Kaplan, Dominic Thomas and Rion Dooley for their technical support and assistance; and Marshall Smith for his mentorship and vision.

NOMENCLATURE

PHM	Prognostics and Health Management
RUL	Remaining Useful Life
PHMF	Prognostics and Health Management Framework
API	Application Programming Interface
SBA	Sensor-based Algorithms
CI/CD	Continuous Integration and Continuous Delivery
API	Application Programming Interface

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Atamuradov, V., Medjaher, K., Dersin, P., Lamoureux, B., & Zerhouni, N. (2017). Prognostics and health management for maintenance practitioners-review, implementation and tools evaluation. *International Journal of Prognostics and Health Management*, *8*(3), 1–31.

Brundage, M. P., Sexton, T., Hodkiewicz, M., Dima, A., & Lukens, S. (2021). Technical language processing: Unlocking maintenance knowledge. *Manufacturing Letters*, *27*, 42–46.

Chen, Y. (2012). *Data quality assessment methodology for improved prognostics modeling* (Unpublished doctoral dissertation). University of Cincinnati.

Chen, Y., Zhu, F., & Lee, J. (2013). Data quality evaluation and improvement for prognostic modeling using visual assessment based data partitioning method. *Computers in industry*, *64*(3), 214–225.

Coble, J., & Hines, J. W. (2009). Identifying optimal prognostic parameters from data: a genetic algorithms approach. In *Annual Conference of the PHM Society* (Vol. 1).

Coble, J., & Hines, J. W. (2011). Applying the general path model to estimation of remaining useful life. *International Journal of Prognostics and Health Management*, *2*(1), 71–82.

Cofre-Martel, Sergio and Corrêa-Jullian, Camila and López Droguett, E and Groth, Katrina M and Modarres, MM. (2021). Defining degradation states for diagnosis classification models in real systems based on monitoring data. In *Proceedings of the 31st European Safety and Reliability Conference (ESREL 2021), Angers, France* (pp. 19–23).

Cofre-Martel, S., Lopez Droguett, E., & Modarres, M. (2021). Big machinery data preprocessing methodology for data-driven models in prognostics and health management. *Sensors*, *21*(20), 6841.

Corrêa, D., Polpo, A., Small, M., Srikanth, S., Hollins, K., & Hodkiewicz, M. (2022). Data-driven approach for labelling process plant event data. *International Journal of Prognostics and Health Management*, *13*(1).

Corrêa-Jullian, Camila and Groth, Katrina M. (2022). Opportunities and data requirements for data-driven prognos-

tics and health management in liquid hydrogen storage systems. *International Journal of Hydrogen Energy*, *47*(43), 18748–18762.

Elattar, H. M., Elminir, H. K., & Riad, A. (2016). Prognostics: a literature review. *Complex & Intelligent Systems*, *2*(2), 125–154.

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.

Goebel, K., Daigle, M. J., Saxena, A., Roychoudhury, I., Sankararaman, S., & Celaya, J. R. (2017). *Prognostics: The science of making predictions*.

Griffiths, T., Corrêa, D., Hodkiewicz, M., & Polpo, A. (2022). Managing streamed sensor data for mobile equipment prognostics. *Data-Centric Engineering*, *3*.

Guo, Z., Bao, T., Wu, W., Jin, C., & Lee, J. (2019). Iai devops: A systematic framework for prognostic model lifecycle management. In *2019 prognostics and system health management conference (phm-qingdao)* (pp. 1–6).

Hu, Y., Miao, X., Si, Y., Pan, E., & Zio, E. (2022). Prognostics and health management: A review from the perspectives of design, development and decision. *Reliability Engineering & System Safety*, *217*, 108063.

Jia, X., Zhao, M., Di, Y., Yang, Q., & Lee, J. (2017). Assessment of data suitability for machine prognosis using maximum mean discrepancy. *IEEE transactions on industrial electronics*, *65*(7), 5872–5881.

Jimenez, J. J. M., Schwartz, S., Vingerhoeds, R., Grabot, B., & Salaün, M. (2020). Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. *Journal of Manufacturing Systems*, *56*, 539–557.

Lukens, S., Rousis, D., Baer, T., Lujan, M., & Smith, M. (2022). A data quality scorecard for assessing the suitability of asset condition data for prognostics modeling. In *Annual Conference of the PHM Society* (Vol. 14).

Marti-Puig, P., Blanco-M, A., Cárdenas, J. J., Cusidó, J., & Solé-Casals, J. (2018). Effects of the pre-processing algorithms in fault diagnosis of wind turbines. *Environmental Modelling Software*, *110*, 119-128. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1364815217302104` (Special Issue on Environmental Data Science and Decision Support: Applications in Climate Change and the Ecological Footprint) doi: https://doi.org/10.1016/j.envsoft.2018.05.002

Myrberg, J. (2022). *aakr*. Retrieved from `https://pypi.org/project/aakr/`

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.

Ramasso, E., & Saxena, A. (2014). Performance Benchmarking and Analysis of Prognostic Methods for CMAPSS Datasets. *International Journal of Prognostics and Health Management*, *5*(2), 1–15.

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

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management* (pp. 1–9).

Saxena, A., Roychoudhury, I., Celaya, J., Saha, S., Saha, B., & Goebel, K. (2010). Requirements specification for prognostics performance-an overview. *AIAA Infotech@ Aerospace 2010*, 3398.

Vachtsevanos, G., Lewis, F. L., Roemer, M., Hess, A., & Wu, B. (2006). Intelligent fault diagnosis and prognosis for engineering systems. In *1st ed. hoboken.*

Walker, M., & Kapadia, R. (2009). Integrated design of online health and prognostics management. In *Annual Conference of the PHM Society* (Vol. 1).

Wang, T., Yu, J., Siegel, D., & Lee, J. (2008). A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *2008 international conference on prognostics and health management* (pp. 1–6).

Wang, Y., Zhao, Y., & Addepalli, S. (2020). Remaining useful life prediction using deep learning approaches: A review. *Procedia manufacturing*, *49*, 81–88.

Zhang, T., Chen, J., Li, F., Zhang, K., Lv, H., He, S., & Xu, E. (2022). Intelligent fault diagnosis of machines with small & imbalanced data: A state-of-the-art review and possible extensions. *ISA transactions*, *119*, 152–171.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213–237.

BIOGRAPHIES

Peter Bishay is a Data Scientist at LMI with experience in model development focused on resolving complex mapping problems. His recent interests are focused on data and model operation scalability. Peter earned his Master's degree in Economics from George Mason University with a focus on analytics.

Sarah Lukens is a Data Scientist at LMI. Her interests are focused on data-driven modeling for reliability applications by combining modern data science techniques with current industry performance data. This work involves analyzing asset maintenance data and creating statistical models that support asset performance management (APM) work pro-

cesses using components from natural language processing, machine learning, and reliability engineering. Sarah completed her Ph.D. in mathematics in 2010 from Tulane University with focus on scientific computing and numerical analysis. Sarah is a Certified Maintenance and Reliability Professional (CMRP).

Damon Rousis is a Data Scientist at LMI with extensive experience in applying analytical techniques to predictive maintenance problems. In particular, Damon is focused on design and productization of machine learning algorithms using sensor data for real-time decision making. Damon received his Ph.D. in Aerospace Engineering from Georgia Tech in 2011 focusing on complex system design and computer simulation.

Nathan is a Data Scientist at LMI with a broad background in developing and applying machine learning models to thorny data. Nathan's recent focus areas have included niche models for classification tasks where labels are scarce and/or untrustworthy, in the contexts of predictive maintenance and entity resolution. Nathan received his Ph.D. in quantitative political science from Emory University in 2013, where he focused on the quantitative study of international conflict processes.