An Introductory Approach to Time-Series Data Preparation and Analysis

Edward Baumann[1], Hayley Buba[2], Taylor Cox[3], and Charles Hsu[4]

[1,2,3,4] *Trident Systems Inc., Fairfax, VA, 22030, USA*
edward.baumann@tridsys.com
hayley.buba@tridsys.com
taylor.cox@tridsys.com
hsu@tridsys.com

ABSTRACT

Machine learning (ML)/Artificial Intelligence (AI) has widespread applications and has revolutionized many industries due to advanced and matured sensor technologies, as well as large-scale data collection efforts. One of the key tasks for effective ML/AI operations is the extraction and identification of useful and usable data to identify complex interrelationships and solve problems efficiently. The usefulness of the data is the value and meaning of the data within the desired model, while the usability of the data refers to the ease of use of data in a model. Complex supervised and unsupervised ML models, which used to be the domain of cutting-edge scientists and academics, can now be invoked as basic function calls in public domain packages within Python, R, MATLAB, and other languages. While these functions require effective data preprocessing to overcome the unpredicted impacts of data quality in the real world (e.g. missing data, environmental noise, synchronizing at different sampling rates, etc.), their ease of use means they are often called with little to no understanding of the underlying math or ways to efficiently work through the data set. The approachability provided by the packages enables users to dive into complex problem sets with little advance preparation. However, in doing so there is a lack of understanding which will inevitably cause problems, skew results, or force the user to take a less efficient path to get to a similar answer. Each package provides relatively simple examples that deal with specific public data sets, yet not many provide the background knowledge and comprehensive methods required for building the inputs for extensive and effective time-series data modeling. Typically, the complex nature of time-series sensor data requires an in-depth understanding of signals analysis and domain subject expertise to use in ML/AI predictive models. This paper will provide the reader an overview of the problems associated with time-series sensor data modelling, propose a common set of preprocessing steps to follow, demonstrate a taxonomy classification for time series data, provide introductory reasoning regarding the underlying process, and discuss the models that would benefit from such a methodology. This is done here with the goal of equipping non-knowledge-domain experts with updated and approachable techniques to find which features to focus on while preprocessing for their time-series data preparation efforts.

Keywords: Machine learning (ML)/Artificial Intelligence (AI), supervised and unsupervised ML, data preprocessing, time-series data, knowledge domain, probability distribution, feature extraction and selection, data preparation.

1. INTRODUCTION

Time series sensor data is a sequence of historical measurements of an observable variable at prescribed time intervals. There are many interests of studies that use time series for predictions, but the scope of this paper will concentrate on the data required for prognostic health management (PHM) of combustion engine vehicles such as speeds, pressures, temperatures, and the like.

A common viewpoint is to see time series data as just another data point to help with model predictions, however adding this variable into a problem makes it more complex. A few example concerns that come with handling time series data are unordered timestamps, timestamp format changes, timestamps collected incorrectly/have unexpected delay, missing values or timestamps, sudden changes in data types, out of range values, rounded values, or aggregated data points. With all the problems that could come up with time series data preparation, the later sections of the paper will show how to

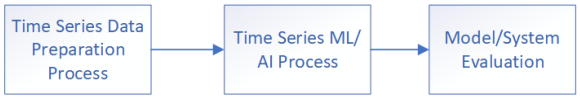

Figure 1. High Level Time Series Model

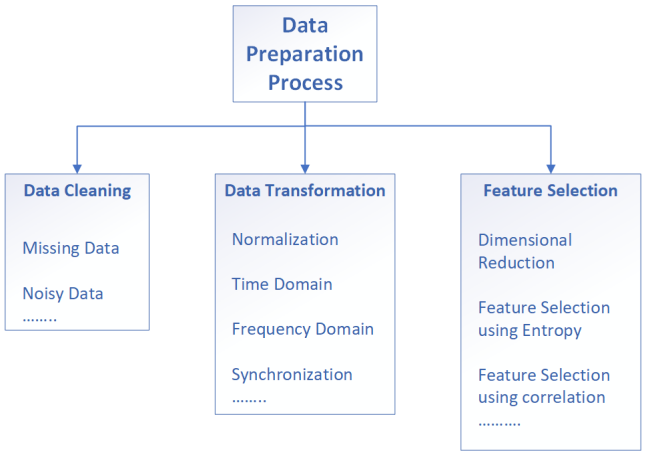

Figure 2. Time Series Data Preparation Components

deal with them to make a dataset useful for ML/AI models to process.

ML/AI Time Series modeling is an important, but challenging, topic of research and development (R&D), which has attracted the attention of research communities within numerous practical fields such as business, defense, economics, finance, science and engineering, and more over last few decades. The main objective of ML/AI time-series modeling is to effectively ingest sensor data, extract features, arithmetically analyze the past time series observations/features and develop an appropriate ML/AI time-series model to describes the inherent structure of the series. This model is then applied to generate future values for the series and make forecasts. With the increase of time series data availability, more ML/AI time series algorithms have been proposed and developed. A high level ML/AI time series model is depicted in Figure 1.

The data preparation process in Figure 1 has a significant impact on the performance of the ML/AI models. From a technical perspective, the main objective of the data preparation process is to ensure the data quality for the ML/AI (Han, 2011) interpreted by its usability and usefulness. The usability of the data is commonly determined by several factors including *accuracy*, *completeness*, *consistency*, *timeliness*, *reliability*, and *interpretability* (Teng, 1999), while the usefulness of the data is ensured after three operations: data cleaning, data transformation, and feature selection as depicted in Figure 2.

During the last two decades, time series data classification and prediction has been considered as one of the most challenging problems in data mining (Esling & Agon, 2012). One of the most popular and traditional time series data approaches is the use of a nearest neighbor (NN) classifier coupled with a distance function to support classification between known classes or distance away from known classes. It was also shown that collecting the individual NN classifiers (with different distance measures) outperforms the ensemble's individual components (Lines & Bagnall, 2015). In each of these approaches, selecting the correct raw and engineered features is critical to a successful and efficient model output. Within time series data, the raw and engineered data can be classified based on the data contents. Signals can be considered constant, binary, low-state, mid-state, high-state, diagnostic, or utility. Each category of signal can indicate the appropriateness for different time series models.

This paper will first introduce the issues/problems faced with time series data, preprocessing, and review time series ML/AI model types. Section 2 outlines the problem regarding time series data models more specifically. Section 3 identifies steps and methods of time series data preprocessing including signal taxonomy, feature selection, data imputation, cleaning, synchronization, determination of data usefulness and usability to optimize the ML/AI process. Particular models are introduced in Section 4 to provide background on time series model capabilities. Next, model evaluation with the preprocess of time series data using different evaluation metrics is examined to understand the model performance of ML/AI as well as the strengths and weaknesses in Section 5. A ML/AI model utilizing some of the preprocessing is demonstrated and implemented with Python using real world data from a Honda CR-V in Section 6. Section 7 outlines the conclusion and future work.

2. PROBLEM FORMULATION

A time series is a sequential set of data points, typically measured over known time steps. It is mathematically defined as a set of vectors $x(t), t = 0, 1, 2, \ldots, n$ where t represents the time elapsed, and the variable $x(t)$ contains the measurements taken during an event in a time series in a set chronological order. For simplicity, the time steps are considered to be linear in our discussion, as small fluctuations in the time delta are normal for engine data points, but are typically not harmful to the time series modeling.

In time series ML/AI modeling, past observations are collected, analyzed, and used to develop a suitable mathematical model. The future events are then predicted or classified using the model. This approach is useful when there is insufficient knowledge about the statistical pattern followed by the successive observations or when there is a lack of a satisfactory explanatory model. Time series forecasting has important applications in various fields. Often valuable strategic decisions and precautionary measures are made due to a good

Table 1. A taxonomy of time series data

Signal Type	Information Value	Example Data
Constant	Contains static vehicle information and signals that never change state.	*Vehicle Identification Number*
Binary	Commonly represents warning lights and other on/off systems that vary.	*Switches, Indicator Lights*
Low-State	Low number of possible values for this signal (generally single digit).	*Vehicle Gear*
Mid-State	Contains Low-State signals that exhibit variance into the double digits.	*Percentages, Pressures*
High-State	Complex signals that represent high-resolution or variance sensor data.	*Engine Temperature, RPM*
Diagnostic	Typically string values that provide information on the system failures.	*Failure Mode Indicator (FMI)*
Utility	Values for overall vehicle usage or as tools to support calculations.	*Time, Trip Distance, Counters*

forecast, i.e. fitting an adequate model to a time series is very important.

However, there are certain fundamental problems with ML/AI applications to time series data. While a relatively recent explosion in standardized packages has made complex models available in a single line of code, little information is typically provided beyond a basic example using synthetic data (many times built on a sine function). The user is left a lot of trial and error or complex digressions into ML theory to try to identify the best method to attack the problem. Time series data further complicates this situations with the fact that the data is typically cyclical, but in often erratic ways. Patterns may be observed in one data set, but not again until a few more data collection efforts have been undertaken. This issue and problem make the ML/AI training inefficient, even untrainable, if the data pattern cannot be found or recovered. Overfitting happens when a model learns the detail and noise in the training data to the quantity that it negatively impacts the performance of the model on generalized new data. This occurs when the noise or random fluctuations in the training data are measured and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models to generalize. The Principle of Parsimony (Ariew, 1976) states that a good time series ML/AI model goal is to achieve a desired level of data fitting using as few explanatory features as possible.

3. STEPS FOR PREPROCESSING TIME-SERIES DATA

When handling time-series data common problems to arise may include non-synchronized sample rates, the significant presence of noise in a target signal, and outlier detection. This leads to the implementation of core signal processing techniques into the preprocessing workflow to generate data usable for the purposes of implementation with ML and AI processes. This section explores these problems and endeavors to provide sample solutions by which this necessary preprocessing may be conducted. Furthermore, the background and basis of these techniques is briefly discussed for the benefit of added context.

3.1. Usefulness and Usability of Data

Before beginning with the preprocessing of data for ML and AI, it is prudent to first evaluate the dataset being worked with to form a determination of its usefulness and usability. In this context the data's usefulness refers to the value provided by the data to the end model's expected output. A data point that does not contribute to the classification of a system has low usefulness. The data's usability refers to the ease with which the data may be handled to arrive at the desired end result. Hence, data the requires cumbersome amounts of preprocessing and large amounts of training time to produce a result of comparably low merit demonstrates a poor relationship between the data's usefulness and usability when compared to input data that can be more easily processed and generates more valuable results. These factors are useful in guiding efforts to develop successful ML and AI algorithms and encourage identifying opportunities presented by datasets that are not only valuable, but attainable as well.

3.2. Time Series Taxonomy

The direct applicability of a signal to a desired model output (that is, it's usefulness and usability) may not be immediately known. A method to quantify the usefulness of signals to models is presented in the Taxonomy in Table 1.

The process begins by first evaluating the extent of variation within the input signal. A signal with a single possible value leads to an input being evaluated as a *constant* signal (e.g. the Vehicle Identification Number or a "power on" signal). These signals represent useful metadata to track, but are typically not otherwise helpful.

A signal with two possible values is evaluated as a *binary* signal (e.g. signal lamp indicators or door status indicator). These signals are typically useful for platform status information or configuration.

Signals with a limited number of Y-values (typically less than 10) would lead to an input's evaluation being that of a *low-state* signal (e.g. the transmission gear selected). These signals are beneficial to more complex models to provide input on the operational state of the platform in question.

Signals exhibiting variation beyond the low-state limit are treated as variable signals as either *mid-state* or *high-state*,

where the distinction is based on the number of bits used for the data. How the data is interpreted is based on the data definition for the signals, which should clearly articulate what the data ranges are, removing issues with misclassification. These data points provide the richest data set for complex machine learning, although they can be prone to noise or other artifacts in the data.

Two other categories exist in the data. *Diagnostic* signal data types are those where the "signal" that is presented is really an indicator. An example of this would be a Controller Area Network (CAN) bus Failure Mode Indicator (FMI), where different numeric values represent distinct failure modes and there is no math or conversions that can change their meaning and have it still be intelligible. The final category are *utility* signals. These are not useful for machine learning directly, rather they provide context, indeces, and support for calculations (e.g. time, hours, distance).

To build an effective time series model, mid and high-state signals will be considered for the main data inputs, using constant signals for metadata tagging and low-state signals to define the operational profile of the platform and provide a means for clustering the results.

3.3. Noise Removal and Outlier Handling

After selecting the appropriate signals, the next step of preprocessing required for the preparation of the data is noise removal and outlier removal. Most commonly, noise removal is taken to refer to the process by which undesirable background-level characteristics of a given signal are smoothed out or otherwise replaced to diminish their overall impact on the represented signal as a whole. In the context of time series data, noise could additionally be introduced from extra sources, such as a data input remaining active in a time period it is not expected to be active, or from a temporary time period in which a given input demonstrates erratic or abnormal behavior of its own.

In these cases of noise processing, it is important that the person processing the data takes the appropriate time to assess the data being considered as noise in order to verify that it is appropriate to treat it as such. If a time period of abnormal sensor readings is indeed indicative of an external failure, removing the evidence of that process's presence or diminishing it via the removal of the "noise" could prove significantly damaging to final results.

One common method implemented for noise removal is to apply a low pass filter to the data. Many possible window functions exist to accommodate the variety of signal properties and behaviors that a given problem set may need to address, but a common choice is the Gaussian filter, which is known for its ability to smooth the data to which it is applied. Through this process some noise is removed, but it is also critical to understand the additional side effects of any window function that is selected for noise removal. For instance, while the Gaussian window smooths the target signal it may also spread out isolated peaks in the data to inhabit an artificially widened span of time. Behaviors like this are what render it critical to understand the behavior of any window that is applied to data for final processing.

With noise removed from the data, outlier detection and correction follows. Various metrics exist for determining what values to classify as outliers, such as the 1.5 Inter-Quartile Range (IQR), 3σ, machine learning models such as Autoregressive Integrated Moving Average (ARIMA), Holt-Winters, Dynamic state-space models, Principal Component Analysis (PCA) analysis, Long-short term memory (LSTMs) and Recurrent Nueral Networks (RNNs). Each has their own strengths and weaknesses, however, the 3σ rule and IQR can be considered reliable means of outlier detection for common use cases.

There are three categories of outliers that are commonly seen in data sets (Jones, 2019). The first are *global outliers* or *point anomalies*, which are data points that are far outside the operational range of the data set. *Conditional outliers* are if the value significantly deviates from the rest of the data points in the same context. So, in time series data, in the context of time passage, there would be some data that would be considered an outlier in the context of the time series. This type of outlier is common in time series data. The last type are *collective outliers*, which is a collection of data points that deviates significantly from the entire dataset. The data points themselves may look fine and would not be classified as outliers, but when they are investigated as a group, their behavior is anomalous. An example in time series data would be the normal peaks and valleys of the data set occurring outside of the time frame.

The 1.5 IQR is a part of the IQR method to detect outliers. For the IQR method, the values needed to calculate the method is the median (or center point) of the data, the first quartile (Q1) which are the values that lie between the minimum and 25% of the data and third quartile (Q3) which are the values that lie between the minimum and 75% of the data. The difference between Q3 and Q1 is called the Inter-Quartile Range (IQR), and the two equations of the upper or lower bounds of the data set so if a data point is less than the lower bound or greater than the upper bound the data point is considered an outlier, so the equation would be:

$$IQR = Q3 - Q1 \tag{1a}$$

$$LowerBound : (Q1 - 1.5 * IQR) \tag{1b}$$

$$UpperBound : (Q3 + 1.5 * IQR) \tag{1c}$$

The 1.5 IQR of the method is the constant multiplied by the IQR such that any data that lies beyond the lower or upper

bound of the mean on either side will be considered an outlier (Jones, 2019).

The 3σ rule is another statistical rule for detecting outliers in data. It states that data populations lie within three standard deviations of the mean. To calculate the 3σ limits, the standard deviation of the data is calculated. The 3 multiplier for this method is a constant that is multiplied by the standard deviation to identify the outliers for the data as the three standard deviations will typically encompass 99.7% of a normally distributed data set (Pukelsheim, 1994).

If the user is unsure, they should physically review the data prior to and after conducting outlier detection and noise removal to ensure the data still exhibits the required features.

3.4. Data Cleaning

Once certain that the data to be used is both useful and usable under a problem's given constraints and the data is time-synced and cleaned, the next step of preprocessing required for the preparation of the data is data preprocessing with data cleaning. Time-series data presents a unique challenge to data cleaning in that the obtained data often may contain results that suffer from asynchronous sampling rates from disparate sensors. This is a critical issue to address early in data preprocessing, as most models require either a common time scale or close time-alignment.

This property is commonly observed in the data by the misalignment of starting times across sensor data, or by varying amounts of samples collected by each sensor as a result of differing sampling rates across the system. The solution to this problem is to process the data such that the same amount of data points are associated with all of the involved sensors, and that the sensors all agree on a common start time. Typically, this is completed by resampling the signal to achieve the sample size desired. In the case of down-sampling a signal, there is an element of information loss that occurs as the quantity of samples is lowered, while the process of up-sampling results in many missing values in the data. In order to avoid the information loss associated with signal down-sampling additional processing must be paired alongside up-sampling to handle the inclusion of null values. Down sampling is typically the target operation as most models are generated off of slower than full speed data.

This processing takes the form of imputation, more specifically interpolation. Other methods of imputation are widespread for data cleaning purposes but are ill-suited to the time-consecutive nature of time-series data. The use of interpolation techniques relies on applying a regression using existing data as endpoints in order to determine values for the null points evenly spaced between them that were introduced by the prior up-sampling. Linear regressions are commonly introduced for this purpose, but a strong understanding of the behavior or a target dataset could as well lead to alternate functions that serve as the basis for the regression such as polynomial interpolation. The primary drivers of the correct type of interpolation are how much data needs to be interpolated and the shape of the data around the missing point.

It is even possible to interpolate the initial data points that may not exist as some sensors were still powering up. By determining the slope of the first values that exist, interpolation techniques can preserve this slope in the beginning data points. For decidedly nonlinear data, alternate best-fit functions may alternatively be used in the determination of suitable data for the replacement of null values. Similar techniques are also applicable to up-sampled data that results in nulls after the final data point in the time series.

3.5. Feature Selection

A feature is the term used for the measurements/values that exist in a dataset, whether they are raw data or engineered features, that will provide the input to the ML model. Thus, feature selection is the most critical part of preprocessing as selecting the wrong features can render an otherwise useful model incapable. The primary reason for downselecting the features that are provided to the model is to achieve the target value or output of the model as efficiently as possible. The first way to deal with feature selection is through the taxonomy provided earlier. After that, the decision point hinges on whether the data is labeled or not. In the scope of PHM, labeling determines whether the data represents a system that is healthy or not at a specified point in time, the remaining time until a failure or maintenance action, or other metadata that would be used to determine the final classification. Since most raw data given is not labeled with health statuses, an secondary approach is to determine the relationships between the data to identify points that are anomalous. It is frequently true that most PHM data will not have a target variable as the data is typically fully healthy or unknown, so identifying the correlation of the data will show which features will be best to predict data in ML use (Kumar & Minz, 2014).

One correlation statistic that is in common use in data science and easy to use is the Pearson's Correlation Coefficient. The Pearson's Correlation Coefficient is a measure of the strength of a linear association between two variables as denoted by r. The range of r can be $+1$ to -1 and a value of 0 indicates that there is no association between the two variables, a value less than 0 is an inverse association (Profillidis & Botzoris, 2019).

Using entropy method as a means to analyze the uncertainty of the data for the feature selection is depicted in the data preparation process in Figure 2 and given by equation 2 where x_i is the $i-th$ element of a data set with N elements and $P(x)$ is the probability distribution of x_i under the condition that $\sum P(x_i) = 1$ and $0 < P(x_i) < 1$ (Shannon, 1948).

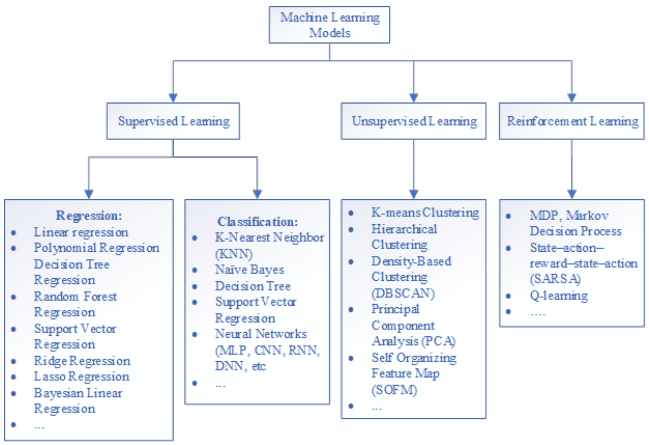

Figure 3. Tree of Machine Learning Models.

$$H(s) = -\sum_{i=1}^{N} P(x_i) log(P(x_i)) \tag{2}$$

Technically, uncertainty is a basic feature of automatic and semi-automatic processes in time series data (Keijzer, Keulen, & Dekhtyar, 2007). Uncertainty information arises from different resources such as process uncertainty, model uncertainty or environmental uncertainty, etc. Many solutions have been studied to reduce uncertainty due to risks of losing relevant information and misleading results(Radzuan, Othman, & Bakar, 2013). The objective of uncertainty analysis using Entropy is to determine the degrees of uncertain data to gain knowledge, fit low dimensional model, and improve prediction. A signal with high Entropy can be considered to *potentially* carry more information, although care must be taken in that random noise would have the highest possible Entropy in a given set of signals.

A final important method of feature engineering data before passing it to the model is to transfer the time domain into the frequency domain method. The transformation into the frequency domain is achieved by applying a mathematical transformation. The most common transformation used is the Fast Fourier Transformation (FFT). A FFT is a technique to visualize time series data in the frequency domain to obtain an additional feature for the ML process. For instance, a power spectrum density and spectrogram can be obtained using a FFT, which can be applied to provide the vibration profile for ML. Frequency order analysis can be conducted on the cyclical rotation of an engine. In addition, the short time FFT (STFT) can be used to provide the significant features of the vibration analysis in a time sequence for the given system.

4. TIME SERIES MODELS

The objective of ML/AI model(s) is to find the connections or correlations between input data and output data, and then support decision making. Theoretically, ML/AI models can be categorized into 3 categories (depicted in Figure 3) based on the type of the input data used to train the algorithms and the resulting objectives.

The supervised learning algorithms are provided an input dataset and then rewarded or optimized to meet a set of specific outputs. In unsupervised machine learning, the algorithm is provided an input dataset without being rewarded or optimized to specific outputs, and instead trained to group objects by their common characteristics. The Reinforcement learning algorithms are made to train itself using many trial and error experiments. Reinforcement learning happens when the algorithm interacts continually with the environment, rather than relying on training data.

There are two main types of supervised learning problems: *regression* to predict the numerical label; and *classification* to predict the class label. A number of unsupervised learning algorithms is depicted in Figure 3. Linear regression is used to identify relationships between the variable(s) of interest and the input data set, and predict its values based on the values of the input variables. Naive Bayes is used to classify objects using probability of features under the assumption of independence of variables. Decision trees are similarly classifiers used to determine the category by traversing the leaf's and nodes of a tree. Random forest models are a collection of many decision trees from random subsets of the data, resulting in a combination of trees that may be more accurate in prediction than a single decision tree. The K-nearest neighbors (KNN) technique involves grouping the closest objects in a dataset and finding the most frequent or average characteristics among the objects. Support Vector Machines (SVM) create coordinates for each object in an n-dimensional space and use a hyperplane to group objects by common features. The K-Means algorithm finds similarities between objects and groups them into K different clusters. Hierarchical clustering builds a tree of nested clusters without having to specify the number of clusters. Self-Organizing Feature Maps (SOFM) are an unsupervised machine learning technique to produce a low-dimensional (typically two-dimensional) representation (clusters) while preserving the topological structure of the data. Principal Component Analysis (PCA) is an unsupervised, non-parametric statistical technique primarily used for dimensionality reduction in machine learning. DBSCAN is a density-based clustering non-parametric algorithm. The Markov decision process (MDP) is a mathematical ML model used for modeling decision-making problems where the outcomes are partly random and partly controllable. State-action-reward-state-action (SARSA) and Q-learning are two reinforcement learning methods that do not require model knowledge, only observed rewards from many experiment runs.

The ARMA (Box & Jenkins, 1976) (Autoregression Mov-

ing Average) model is a stochastic model commonly used for time series forecasting problems. Typically, Autoregression (AR) and Moving Average (MA) models are effectively combined to form a general and useful class of time series models, where AR is to use observations from previous time steps as input to the regression equation to predict the value at the next time step. The role of the MA is to average the observations from the previous time steps to predict the direction of a trend.

5. EVALUATION

Classification performance metrics such as accuracy, precision, recall and classification error can be used to assess the performance of time series models introduced in 4. These metrics summarize the performance of the classifier while presuming that all classes are equally important and can be used to drive the selection of tuning parameters or the type of classifier implemented. Although valid single-point metrics, these metrics do not take into account the temporal aspect of the Remaining Useful Life (RUL) estimation problem or the fact that failing to correctly estimate low-value RULs is more critical than failing to predict high-value RULs.

A confusion matrix captures the error distribution of the classifier per class. It can be applied to both binary and multi-class classification problems when the true classification labels are known. For a binary classification problem, the confusion matrix shows four different classification counts, namely true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) as shown in Table 2. A TP (TN) indicates a sample in the positive (negative) class was classified correctly, and an FP (FN) a sample in the negative (positive) class that was classified as positive (negative). The multi-class classification model of the confusion matrix can then be extrapolated as follows (Krüger, 2016), see Figure 4. Per row $n \in \mathcal{C}$, the confusion matrix $\mathbf{E} \in \mathbb{N}^{(N+1)\times(N+1)}$ comprises a $1 \times (N+1)$ vector whose n'-th entry is $\sum_{m:c_m=n} \mathbf{1}_{\{\hat{n}_m=n'\}}$. The entries of the n-th row of $\mathbf{E}$, with the n-th entry removed, correspond to the FN count for class n. Similarly the entries of the n-th column of $\mathbf{E}$, with the n-th entry removed, correspond to the FP count for class n. Let $\mathbf{1}$ denote a vector of ones with appropriate dimensionality, $\text{diag}(\mathbf{E})$ as an $(N+1) \times (N+1)$ matrix comprising the main-diagonal entries of $\mathbf{E}$ on its main diagonal, and $(\cdot)'$ as the transpose operator. Thus, the $(N+1) \times 1$ vector $\boldsymbol{\alpha} := (\mathbf{E} - \text{diag}(\mathbf{E}))\mathbf{1}$ captures the FN count profile and the $(N+1) \times 1$ vector $\boldsymbol{\beta} := (\mathbf{E} - \text{diag}(\mathbf{E}))'\mathbf{1}$ captures the FP count profile.

Specific RUL estimators can be compared on the basis on these two profiles and their accuracy score $A \in [0, 1]$ through, e.g., the Euclidean distance between $\Theta := (\|\boldsymbol{\alpha}\|_2, \|\boldsymbol{\beta}\|_2, 1 - A)$ and the ideal score tuple $(0, 0, 0)$. This approach, however, ignores the temporal aspect of the RUL estimation problem and the fact that a false negative estimate that predicts a RUL that is smaller than true RUL is preferable to one that predicts a RUL that is larger than the true RUL. The former case would give the system a chance to react to an impending failure while the latter one would not.

		Estimated	
		positive	negative
Actual	positive	**TP** True Positive	**FN** False Negative
	negative	**FP** False Positive	**TN** True Negative

Table 2. Binary classification confusion matrix.

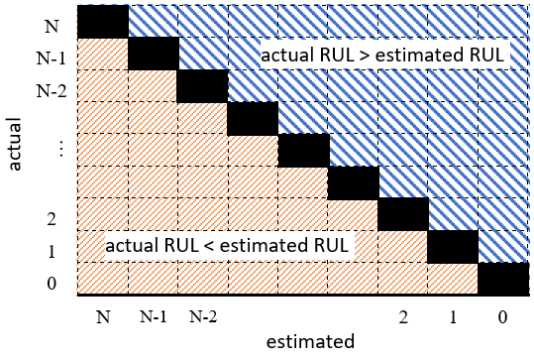

Figure 4. N-ary classification confusion matrix.

In the context of RUL estimation, given a class n all FN values assigned to classes $n' > n$ should be weighed more than those assigned to classes $n' < n$. This can be achieved for each n by using a masking function defined entry-wise as:

$$g_n(n') = \begin{cases} \lambda_1 & n' < n \\ \lambda_2 & n' \geq n \end{cases} \tag{3}$$

with scalars $0 < \lambda_1 < \lambda_2$. Let $\mathbf{G} := [\mathbf{g}_1, \ldots, \mathbf{g}_{N+1}]'$, with $\mathbf{g}_n := [g_n(1), \ldots, g_n(N+1)]'$, denote the resulting masking matrix. Then one can define an adjusted profile $\boldsymbol{\alpha}_{\text{adj}} := [\mathbf{G} \circ (\mathbf{E} - \text{diag}(\mathbf{E}))]\mathbf{1}$, where $\circ$ denotes the element-wise product. A similar argument can be used to argue that for a given class n FPs assigned to classes indexed by n' with $n' < n$ should be weighed more since they will convey an unnecessary sense of urgency for action to system. With these observations, it is possible to define adjusted FP $\boldsymbol{\alpha}_{\text{adj}}$ and FN $\boldsymbol{\beta}_{\text{adj}}$ profiles. Then, the tuple $(\|\boldsymbol{\alpha}_{\text{adj}}\|_2, \|\boldsymbol{\beta}_{\text{adj}}\|_2, 1-A)$ can be used to assess the quality of the RUL estimator by assessing its Euclidean distance from the tuple $(0, 0, 0)$ as before (Baumann, Forero, Selby, & Hsu, 2021).

6. IMPLEMENTATION

To demonstrate the processes above, data was collected from a Honda CR-V driven during normal commuting cycles. The collection was performed using a CanEdge2 from CSS Electronics connected to the OBDII port on the car. The data was collected at fully rate and then down sampled to 1 Hz using the first time stamp for each data type in each second. After collection, the data was converted to a comma seperated value (CSV) file for ingestion into common ML tools, including MATLAB and Python. The unprocessed data can be accessed at: https://www.kaggle.com/datasets/hayley01/honda-car-sensor-readings.

The initial step is to load the dataset into the development environment of choice and perform an initial data review looking for missing values, null values, or other obvious artifacts. In the case of the CR-V, 162 independent signals were present in the data. A cursory glance showed that the number of signals was unrealistically large. Signals such as *time, config valid,* and *output disabled* all clearly show they will not be correlated to an actual signal or platform problem. However, they were all kept in to demonstrate the efficiencies of decreasing the signal count at the start.

Once a general knowledge has been built on the data, it can be pared down with the taxonomy presented earlier. A number of signals registered as constants or binary. Some examples are the ID of the logger (which never changes), and door indicator status lights (which only change when a door is opened). Well over half of the data points could have been eliminated as part of those two categories. Without knowing more about the low, mid, and high-state signals, none of those were dropped at this time. Again, these values were maintained to demonstrate the efficiencies of better preprocessing.

The entropy value for each signal was computed next in order to draw a threshold for split between signals. The top signals were then processed through the steps shown above.

The Entropy measurement for the 160 numerical measurement data was calculated to determine the degrees of knowledge gained in support of the feature selection to fit a low dimensional model and improve prediction. The entropy distribution of the 160-measurement data with 4 resolution bins is depicted in Figure 5.

In Figure 5, 4 entropy groups with 1.2 bin width are formed, which is simply the total width divided into 4 groups, where EntropyGroup1 is between [0 1.2], EntropyGroup2, EntropyGroup3, EntropyGroup4 are in the range [1.2 2.4], [2.4 3.6], and [3.6 4.8] respectively. The research and analysis are conducted in EntropyGroup2 and EntropyGroup3. EntropyGroup1 with lower entropy values contains constants in the measurement data, and EntropyGroup4 with higher entropy values indicates more noise-like signals in the measurement

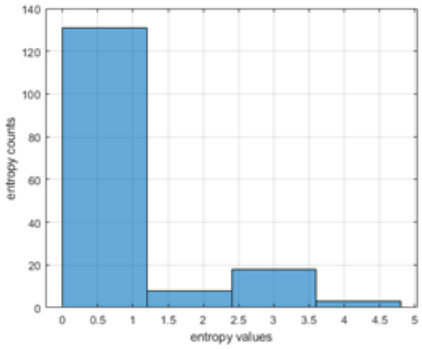

Figure 5. Entropy Distribution of Time Series Measurement Points

Algorithm 1 Time Series Preprocessing

1: Load data into memory.
2: Find the NULL values in the data.
3: Interpolate the NULL values through the calculated mean of the prior values.
4: Find the IQR and remove outliers
5: Compute Entropy and drop signals from the lowest and highest group.
6: **return** Preprocessed signals for ML model.

data. There were 8 signals in EntropyGroup2 and 18 signals in EntropyGroup3. The features of EntropyGroup2 and EntropGroup3 were selected and used for the unsupervised ML using SOFM (Self-Organizing Feature Map). The number of models inputs has been significantly reduced from 160 to 26 for the ML model to learn. A SOFM with Kohonen map is an unsupervised machine learning technique used to produce a low-dimensional representation (clusters) while preserving the topological structure of the data.

There are 160 measurement data (features) in the data set used for the demonstration, and every measurement data contains 24263 time series data. To effectively evaluate the performance, the full 24263 time series data are applied for the unsupervised learning, and the clustering outcome is used as a reference comparison. With the reference, the half of the time series data (12142 data) are used for training, and the rest half of the time series data (12141 data) is tested and its result is compared to the clustering outcome with training using the full time series data.

In this paper, given a data set of a commercial vehicle, the SOFM unsupervised learning algorithm is used to learn the feature map from the input space and cluster the discrete output space. The stages of the SOFM unsupervised learning algorithm can be summarized as follows.

- Initialization – Choose random values for the initial weight vectors w_j
- Sampling – Draw a sample training input vector x from

Table 3. Comparison of signal features as separated by entropy, accuracy rate, processing time, and number of features

Data Group	Accuracy Rate	Processing Time	Features
Group 1	21.78	9.52 sec	131
Group 2	32.22	2.44 sec	8
Group 3	99.82	4.03 sec	18
Group 4	28.64	2.72 sec	3

the input space, where the training input vector x contains selected features using entropy.

- Matching – Find the winning neuron $I(x)$ that has weight vector closest to the input vector, i.e. the minimum value of $d_j(x) = \sum_{(i=1)}^{D}(x_i - w_{ji})^2$.
- Updating – Apply the weight update equation $\Delta w_{ji} = \eta(t)T_{(j,I(x))}(t)(x_i - w_{ji})$, where $T_{(j,I(x))}(t)$ is a Gaussian neighborhood and $\eta(t)$ is the learning rate.
- Continuation – keep returning to step 2 until the feature map stops changing.

The unsupervised SOFM is implemented using the Deep Learning Toolbox in MATLAB V.9.13 (R2022b). The SOFM network is created using MATLAB built-in selforgmap function with a [2 2] dimensions (4 clusters), coverSteps = 170; initNeighbor = 20; topologyFunc = 'gridtop'; and distanceFunc = 'linkdist'. The result is depicted in terms of data group, accuracy rate, processing time, and the number of features in Table 3.

7. CONCLUSION AND FUTURE WORK

This paper provided an overview of some of the problems with using time-series sensor data for PHM problem sets, specifically those related to vehicles with combustion engines. The data itself can be or seem to be incomplete, be overshadowed by noise, contain out of range data, be out of sync with similar data points, or be monitored at different sampling rates than other data points; all of which are problematic for a ML model that wants to predict future states or identify a difference from a set baseline. While many models have migrated into an easy to use package with Python, R, or MATLAB, they are typically accompanied by basic documentation built on synthetic or easy simulation data, without much discussion on how to apply them to more difficult problem sets.

As the first step in the data pipeline for a time-series model, Pre-Processing was discussed in detail as it relates to time-series data. Notably, the need to perform effective Data Cleaning, Data Transformation, and Feature Selection was explored with examples of how this impacts the model results and how someone new to time series data analysis can classify time series data types based on a defined taxonomy. Next, specific time series models were discussed briefly to provide an overview of the resulting capabilities of the captured data. Evaluation metrics for these models were reviewed to identify which models performed the best. Finally, the work discussed above was demonstrated against real-world data from a Honda CR-V to showcase the ability for these steps to reduce the time to implementation for a time series model.

Future work is recommended to present a series of peer-reviewed publications demonstrating the real-world application of time-series model application for PHM problems. Extending the number and complexity of models with demonstrated use cases and non-synthetic data sets will equip and encourage data scientists, engineers, and others seeking to either join the industry or learn new tool sets. Finally, by organizing papers around specific parts of the machine learning pipeline, these can server as tutorials at a greater depth than what is typically available.

NOMENCLATURE

AI	Artificial Intelligence
AR	Autoregression
$ARIMA$	Autoregressive Integrated Moving Average
$ARMA$	Autoregression Moving Average
CAN	Controller Area Network
CBM	Condition-Based Maintenance
FFT	Fast Fourier Transform
FMI	Failure Mode Indicator
FN	False Negative
FP	False Positive
IQR	Inter-Quartile Range
$LSTM$	Long Short Term Memory
MA	Moving Average
MAP	Maximum A Posteriori
ML	Machine Learning
MLP	Multilayer Perceptron
$NASA$	National Aeronautics and Space Administration
NN	Nearest Neighbor
$OBDII$	On-Board Diagnostics 2
$PCoE$	Prognostics Center of Excellence
PMF	Probability Mass Function
PHM	Prognostics and Health Management
PCA	Principal Component Analysis.
$RMSE$	Root Mean-Squared Error
RNN	Recurrent Neural Network
RUL	Remaining Useful Life
$SOFM$	Self-Organizing Feature Map
TN	True Negative
TP	True Positive
TTF	Time to Failure

REFERENCES

Ariew, R. (1976). *Ockham's razor: A historical and philosophical analysis of ockham's principle of parsimony* (Unpublished doctoral dissertation). University of Illi-

nois at Urbana-Champaign.

Baumann, E., Forero, P. A., Selby, G., & Hsu, C. (2021). Methods to improve the prognostics of time-to-failure models. In *Annual conference of the phm society.*

Box, G. E. P., & Jenkins, G. M. (1976). *Time series analysis: Forecasting and control*. Holden-Day.

Esling, P., & Agon, C. (2012, dec). Time-series data mining. *ACM Comput. Surv.*, *45*(1). Retrieved from https://doi.org/10.1145/2379776.2379788 doi: 10.1145/2379776.2379788

Han, J. (2011). *Data mining: Concepts and techniques, 3rd ed.* Morgan Kaufmann.

Jones, P. R. (2019). A note on detecting statistical outliers in psychophysical data. *Attention, Perception & Psychophysics*, *5*(81), 1189–1196.

Keijzer, D. A., Keulen, V. M., & Dekhtyar, A. (2007). *Report on the first vldb workshop on management of uncertain data (mud).* (Tech. Rep.).

Krüger, F. (2016). *Activity, context, and plan recognition with computational causal behaviour models* (Unpublished doctoral dissertation). Universitat Rostock.

Kumar, V., & Minz, S. (2014, Jun). Feature selection: A literature review. *Smart Computing Review*, *4*(3), 211-229.

Lines, J., & Bagnall, A. (2015). Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, *29*(3), 565-592.

Profillidis, V., & Botzoris, G. (2019). *Modeling of transport demand: Analyzing, calculating, and forecasting transport demand*. Elsevier.

Pukelsheim, F. (1994). The three sigma rule. *The American Statistician*, *48*(2), 88–91. Retrieved 2023-06-04, from http://www.jstor.org/stable/2684253

Radzuan, N. F. M., Othman, Z., & Bakar, A. A. (2013). Uncertain time series in weather prediction. In *Procedia technology.*

Shannon, C. E. (1948, July). A mathematical theory of communication. *The Bell System Technical Journal*, *27*, 379-423.

Teng, C. M. (1999). Correcting noisy data. In *16th international conference on machine learning.*