ChieF : A Change Pattern based Interpretable Failure Analyzer

Dhaval Patel Lam M. Nguyen Akshay Rangamani Shrey Shrivastava Jayant Kalagnanam
IBM Thomas J. Watson Research Center
1101 Kitchawan Rd, Yorktown Heights, NY 10598
{*pateldha@us.,LamNguyen.MLTD@,akshay.rangamani@,shrey@,jayant@us.*}*ibm.com*

***Abstract*—Discovering the underlying dynamics leading up to an industrial asset failure is an important problem to be solved for successful development of Predictive Maintenance techniques. Existing work has largely focused on building complex ML/AI models for developing Predictive Maintenance solution patterns, but has largely avoided developing methods to explain the underlying failure dynamics. In this paper, we use an old but significantly improved change-pattern based technique to analyze IoT sensor data and failure information to generate useful and interpretable failure-centric insight. We discuss a solution pattern that we call ChieF, which when applied on multi-variate time series datasets, discover the leading failure indicators, generate associative patterns among multiple features, and output temporal dynamics of changes. Experimental analysis of ChieF on four datasets uncovers insights that may be valuable for predictive maintenance.**

***Keywords*-Change Pattern Algorithms; Failure-Centric Knowledge Extraction; Data Analysis**

I. Introduction

The immense growth in Industrial Internet of Things (IIoT) software platforms[1], such as Watson IoT platform, Azure IoT, AWS IoT, has led to the automated collection of large volumes of digitized data from smart devices in industrial and manufacturing environment. Platforms such as Maximo[2], OSIPi[3], etc., are providing smarter options to connect people, data and systems in a seamless way. One of the key problem these IIoT software platforms try to address is to build an operational intelligence using data analytics. Operational intelligence requirements for IoT applications range from proactive maintenance to predictive and prescriptive maintenance. Proactive maintenance finds anomalies and identifies root causes using interactive and automatic observation of IoT data. Whereas predictive maintenance techniques for industrial applications aims to predict the top risky assets, i.e., assets (or its component) that will fail in the near future if preventive maintenance is not conducted.

Early work on predictive maintenance techniques use SCADA[4] data to monitor assets, and manual thresholds are set based on subject matter expert's knowledge. These techniques trigger an alert when sensor data breach thresholds and hence signal potential machine fault. With recent advancements in Machine Learning and Artificial Intelligence (AI) techniques, industries are bringing various AI enabled predictive maintenance solution patterns [1][2]. These solution patterns utilize novel algorithms such as Remaining Useful Life estimation [3][4], Failure Prediction in the near future [5], and Anomaly Detection [6][7], among others for predictive maintenance. Although these solution patterns are scalable and deployable across various industries, the issue of interpretability and actionable knowledge extraction for predictive maintenance is largely unanswered and limited to [8][9]. More specifically, end users are interested in failure-centric data analysis such as the features responsible for failures, how early these features show behavioral changes, and what are the important relationship among features, etc.

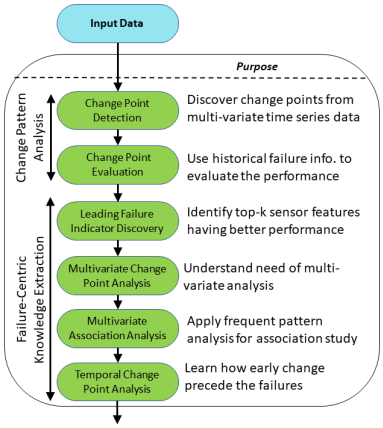


Figure 1: The architecture flow of the system

In this paper we discuss a solution pattern, namely **ChieF** (Change Pattern based interpretable Failure Analyzer), that analyzes IoT sensor data and failure information from multiple assets and provides a interpretable insight for data-driven failure analysis. ChieF uses multiple state-of-the-art change point detection methods on multi-variate time series data [10][11][12], and associates the discovered change points with failure labels for discovering interpretable failure-centric knowledge. The computational procedure of the ChieF approach is presented in Figure 1.

ChieF starts with *application* and *evaluation* of change point detection methods to univariate and multivariate time

[1]The Forrester Wave: Industrial IoT Software Platforms, Q3 2018
[2]https://www.ibm.com/products/maximo
[3]https://www.osisoft.com/
[4]Supervisory Control And Data Acquisition

series data. The complexity of this task mainly depends on the number of assets under consideration, number of sensor variables and quality of the collected raw data. The next step is *Failure-centric Knowledge Extraction*, which comprises of leading indicator discovery, temporal analysis of change points, and frequent pattern analysis for association study. ChieF, as the name suggests, is designed to provide useful information to the other existing solution patterns, as an example, ChieF can help to identify a golden set of failure samples to be used for building supervised model [1], etc.

To the best of our knowledge, a systematic framework for analyzing and extracting failure-centric interpretable information extraction using unsupervised change-pattern analysis has not been proposed earlier. Our key contributions:

- Design and development of *ChieF* (Section II)
- Uses of univariate and multivariate change point detection models for performing analysis (Section III)
- Extension of univariate CUSUM to multi-variate CUSUM (Section III-A)
- Adapted precision, recall and accuracy definition to incorporate the temporal window constraint (Section IV)
- Use of frequent pattern analysis for discovering *Transactional* and *Sequential* patterns (Section VI-C)
- Conducted extensive experiment on four datasets from different industries (Section VI)

II. Change Point based Interpretable Failure Analyzer

A. Input Data

This section discusses the detail of input data (See Figure 2) and the common notations used in the paper. Let $A = \{A_1, A_2, \ldots, A_N\}$ be a set of multi-variate sensor data for N Assets, where A_i represents the multi-variate time series for Asset i. We use A_i^j to refer the time series of sensor variable j for Asset i, and $A_i^{j,t}$ is a value of sensor variable j at time t for Asset i. Let $F = \{F_1, F_2, \ldots, F_N\}$ be an asset wise recorded failures, where F_i is a list of time points when asset A_i fails. The failure information are mostly extracted from maintenance records. Note that, the granularity of failures varies from dataset to dataset, i.e., there are different types of failures for one component, or there are different component failures. It is expected that the time point of failures in F are aligned with sensor data A.

B. System Architecture

The ChieF system mainly performs the following three things- discover change points in sensor data A, validate the change point results with failures information F, and generate change pattern analysis results.

The *Change Point Detection* module takes in the sensor data and leverages various algorithms to generate the changes points either at individual sensor level or at a time

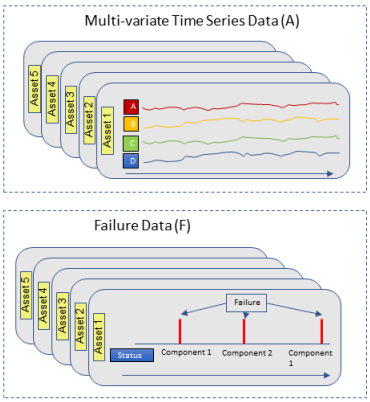


Figure 2: Example of Input Data

series level. We selected three change point detection algorithms namely CUSUM [10], Ruptures [11] and Gaussian graphical model (GGM) [12] for this comparison.

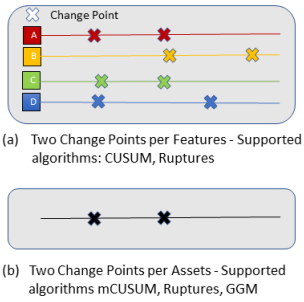
The *Change Point Evaluator* module selects the change point detection algorithm which performs best and examines whether the discovered change points coincide with documented failure points. We evaluate this performance using three evaluation measures discussed in Section IV. These measures are adjusted to accommodate the temporal nature of change point detection output.

The final *Failure-Centric Knowledge Extraction* module post-processes the outputs of change point detection algorithms to generate the insightful and interpretable results. In particular, we aim to discover leading sensor indicators using univariate analysis, identify associative and sequential failure patterns using frequent pattern mining approach, and conduct multivariate change point analysis. Moreover, this module also conducts the temporal analysis of discovered change point in order to identify the appropriate time window for failure detection.

III. Change Point Detection Module

Change points detection algorithms can either be univariate or be multi-variate. The univariate change point methods operate on time series of single feature, and outputs the change point only for that feature. For a multi-variate time series data, the univariate algorithms can be applied on each feature independently. On the other hand, the multi-variate change point detection methods incorporate the relationship between multiple features, and generate the change point at time series level or at attribute-level. Given a multi-variate data with K variables and length L, Figure 3 describes two different outputs that can be generated by different change point detection algorithms. The output is binary matrix of size either K x L (attribute-wise change point) or 1 x L (time-series change point), with value 1 indicating a change point.

We include three methods in our framework- GGM, Ruptures, and CUSUM. GGM can only work with multi-



(a) Two Change Points per Features - Supported algorithms: CUSUM, Ruptures

(b) Two Change Points per Assets - Supported algorithms mCUSUM, Ruptures, GGM

Figure 3: Comparing output of the methods

variate dataset while the other two methods can work with both univariate and multi-variate datasets. In Figure 3, we also include the name of algorithms based on the kind of output they can produce. In the further sections, we will show how our framework provides flexibility to model the multi-variate data as it is or as a set of univariate time series.

A. CUSUM

The *CUSUM* method [10] is a univariate change point detection method based on statistical process control charts. CUSUM is widely used technique to monitor the change point in univariate time series data that show a subtle shift in the mean relative to the context of the time series itself. As name suggest, CUSUM uses cumulative sum of changes to monitor the change in time series. In Equation 1, $Score_i^{j,t}$ is a CUSUM signal value at time point t for Asset i and sensor feature j. When the value of cumulative change exceeds a certain threshold value, denoted as $cusum\_threshold$, we reset $Score_i^{j,t}$ to zero, and declare a change has been found at time point t.

$$Score_i^{j,t} = Score_i^{j,t-1} + (A_i^{j,t} - A_i^{j,t-1}). \quad (1)$$

The value of $cusum\_threshold$ decides the number of discovered change points. In this paper, we provide three alternatives to choose $cusum\_threshold$: "q75", "max", and "min". The option "q75" is the value at 75_{th} percentile of given time series; option 'max'('min') is the the maximum (minimum) value of the absolute difference between two consecutive points in time series. Since, we use the entire time series to decide the value of the above three options, we refer current CUSUM method as an *off-line approach*.

Multi-variate CUSUM: We also extend CUSUM method to work with multivariate data. Figure 4 gives a high level overview of involved steps. First, we detect the change points using univariate CUSUM for each feature. Next, we observe whether the change points from multiple features are detected in temporal window of pre-defined length. In our example, we have shown four temporal windows along with the number of change points discovered across the features. If at least $num\_feature\_choice$ (integer value) features have change points in given temporal window, we mark that a change point is detected, and then choose the latest change as the common change point of all features. In the Figure 4 example, we detected two changes points.

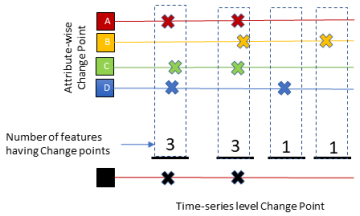


Figure 4: Working of multivariate CUSUM

B. Ruptures

Ruptures is a Python library for *off-line change point detection* [11]. This method can work with univariate as well as multivariate data. Ruptures requires user to specify the number of change points to be detected beforehand.

The working of our proposed approach for multi-variate data A_i goes as follow: we apply a window-based change point detection method on A_i with the fixed temporal window length 30 and ℓ_2 penalty. In window-based change detection, two adjacent temporal windows of equal lengths are compared to calculate the discrepancy score. Intuitively, the discrepancy score compares how similar the observations in both the windows. In (2), the discrepancy score at time point t is calculated using two adjacent windows $A_i^{:,t-w:t}$ and $A_i^{:,t:t+w}$ is given, where $A_i^{:,t-w:t}$ be the temporal window of length w starting from time point $t-w$ and involve all the features of asset A_i.

$$Score_i^{:,t} = Score(A_i^{:,t-w:t}, A_i^{:,t:t+w}) = F(A_i^{:,t-w:t+w}) - F(A_i^{:,t-w:t}) - F(A_i^{:,t:t+w}). \quad (2)$$

The function $F(A_i^{:,t-w:t+w})$ for $l2$ penalty function is defined as follow, where $\overline{A_i^{:,t:t+w}}$ is a mean value of time segment $A_i^{:,t:t+w}$.

$$F(A_i^{:,t:t+w}) = \sum_{t\in[t,t+w]} ||A_i^{:,t} - \overline{A_i^{:,t:t+w}}||_2^2. \quad (3)$$

The above discrepancy score is calculated at various time points on the time series of each assets. Once the discrepancy score is calculated for entire time series, Rupture optimally select the pre-specified number of locations, given by $rupture\_threshold$, using peak detection algorithm as a change points. Note that, $Score_i^{:,t}$ is at time series level. Similarly, we can also calculate the score $Score_i^{j,t}$ using univatiate time series A_i^j

C. Gaussian Graphical Model (GGM)

Gaussian Graphical Model, in short GGM [12], is a multivariate technique. This method uses the multidimensional Gaussian distribution to learn a graphical model that explains

the relationship between sensor variables. In a graphical model, the nodes are sensor variables and the edges encode the conditional dependence structure between random variables. In order to capture the most salient dependencies among variables, GGM discovers a sparse dependency matrix, i.e., many entries in the matrix will be zero, and non-zero entries represents the strength of dependency between two variables. To generate the sparse matrix, there exists various sparsifying formulations such as Graphical Lasso (ℓ_1) method, ℓ_0 method, etc. In this paper, we use Graphical Lasso (ℓ_1) method [12] to generate the sparse matrix.

We use GGM method to detect the change points as follows: we first apply GGM method on input multi-variate data A_i to discover the sparse precision matrix M_i. Next, we use Mahalanobis distance measure to find out the distance between every sample $A_i^{:,t}$ in input data A_i with respect to the learned matrix M_i. The following equation gives a formula to calculate the score for asset A_i at time t based on Mahalanobis distance, where symbol $\odot$ refers to element-wise product of vectors, and $\overline{A_i^{:,t}}$ is mean normalized vector.

$$Score_i^{:,t} = Score(A_i^{:,t}, M_i) = \sum((\overline{A_i^{:,t}} \cdot M_i) \ \odot \ \overline{A_i^{:,t}}).$$

It is expected that the score is very high near the change point. Thus, we introduce the user defined parameter, $ggm\_threshold$, to control the number of change points to be discovered. Our method selects $ggm\_threshold$ time points in input data where the calculated score is high and marks them as a change points.

IV. Change Point Evaluator

The objective of the *Change Point Evaluator* module is to evaluate the output of different change point algorithms with respect to the ground truth failure F. Let $C = \{C_1, C_2, \ldots, C_N\}$ be asset wise predicted change points from a method M, where C_i is a list of time points where change for asset A_i is observed. The change point evaluator compares the detected change point C_i with reported failures F_i, and obtain an average value over all the assets as follow:

$$\Phi(m) = \frac{1}{N} \times \sum_{i=1}^{N} Eval(C_i, F_i, \delta). \tag{4}$$

The definition of $Eval(C_i, F_i, \delta)$ is based on the following intuition. The change points should precede the failure events, i.e., change point method should detect changes "slightly early" than the real failures. We formulate three different evaluation score namely, Window Precision, Window Recall, and Window F1-Score. All these score definition use temporal window (δ) to account the "slightly early" nature of detected change points while comparing with the ground truth failures.

A. Window Precision

This score determines how accurate the detected change points. It is the ratio of the number of change points in C_i that can be associated with ground truth failure F_i using threshold δ to the total number of change points in C_i. It is given as follow:

$$\begin{aligned} WPrecision_i &= \tfrac{|PRel(C_i,F_i,\delta)|}{|C_i|}, \\ PRel(C_i, F_i, \delta) &= \{c \in C_i \mid (\exists f \in F_i) \ \wedge \ (f - c) > 0 \\ &= \ \wedge \ 0 < (f - c) \leq \delta\}. \end{aligned}$$

B. Window Recall

This score calculates failure coverage, i.e., % of detected failures. It is the ratio of the number of failures in F_i that are discovered by change point output C_i using threshold δ to the total number of failures in F_i. It is given as follow:

$$\begin{aligned} WRecall_i &= \tfrac{|RRel(C_i,F_i,\delta)|}{|F_i|}, \\ RRel(C_i, F_i, \delta) &= \{f \in F_i \mid (\exists c \in C_i) \ \wedge \ (f - c) > 0 \\ &= \ \wedge \ (f - c) \leq \delta\}. \end{aligned}$$

C. Window $F1$ Score

Window $F1$ Score is the harmonic average of window precision and window recall, i.e.

$$WF1_i = 2\tfrac{WPrecision_i \cdot WRecall_i}{WPrecision_i + WRecall_i}.$$

V. Experimental Dataset

In this paper, we utilize four multi-asset datasets of multivariate time series. Table I gives a short summary of their properties. Datasets are taken from diverse industries, with different sizes, frequency of failures and data quality (missing data, etc).

Table I: Datasets Description

Name (Industries)	**Assets**	**Sensors**	**Failures**	**Timesteps per Asset**
Dataset-I (Telemachine)	1000	4	6368	8761
Dataset-II (Jet Engines)	709	21	709	128-543
Dataset-III (Gas Industries)	676	20	519	1000
Dataset-IV (Hard drives)	363	63	363	2-299

Dataset-I is a large-scale simulated dataset[5], and contains measurements from machines taken during the course of its operation for a year. There are four sensor variables namely *voltage, pressure, rotation, and vibration*. This dataset does not have any missing readings. There are a total of 1000 machines, and each machine has hourly measurements taken over one year, resulting in 8761 timesteps per machine. Each machine undergoes a failure if one or more of it's four components fail. There are a total of 6368 failures in the dataset. The component level failure distribution is as follow: 1886 (Component 1), 2587 (Component 2),

[5]https://github.com/Microsoft/SQL-Server-R-Services-Samples/tree/master/PredictiveMaintanenceModelingGuide/Data

1012 (Component 3), and 1241 (Component 4). We analyze failures of the individual components as well as that of each machine as a whole.

Dataset-II is publicly available Turbofan Engine Degradation Simulation Data Set[6]. This data is widely used in many academic and industrial studies [3]. This dataset has total 24 features: 3 variable related to engine "settings", and remaining 21 are sensor features. Also, data does not have any missing values. There are a total of 709 engines, and each engine has between 128 and 543 measurements. Most engines have around 200 measurements. Each engine undergoes a failure at the end of a cycle of measurements, making up a total of 709 failures in the dataset. The 709 engines are actually distributed among four different sub-datasets, which differ in the types of testing conditions that the engines are subjected to during the operation. Sub-Datasets Dataset-II_1 and Dataset-II_3 contain 100 engines each and tested under a single condition. Sub-Datasets Dataset-II_2 and Dataset-II_4 contain 260 and 249 engines respectively, and tested under six conditions during its operations.

Dataset-III contains measurements from oil pumps over a period of approximately three years with measurements being made once a day, which means there are about 1000 measurements per assets. The measurements are multivariate recordings from 20 different sensors. The dataset also has missing values. Each sensor is missing around 22%-26% of the total number of samples. This may be due to a variety of reasons, including asset failure. In our change point detection process, we deal with the missing values in a naive way, by simply dropping the timesteps which are not recorded. This dataset has 676 different assets, and there are more than 500 instances of failures. This means that a number of assets in the dataset do not undergo failure.

Dataset-IV contains 63 smart features collected from 300+ hard drives[7]. The recording of smart features were started at arbitrary time points, and stopped when the hard drives fails entirely. The recording of smart features were originally varying temporal granularity. We re-sampled data at an interval of one hours, and the dataset does not have any missing values. On an average, 180 recordings are performed for each hard drives, and we only retain those hard drives for which we have atleast 48 recordings after re-sampling.

VI. Failure-Centric Knowledge Extraction

We conduct experiments to discover insights using univariate and multivariate change point detection methods.

A. Leading Failure Indicator Discovery

We perform univariate analysis on each sensor features using four different methods: $Ruptures$, $CUSUM_{min}$, $CUSUM_{max}$ and $CUSUM_{q75}$. This analysis generates feature wise evaluation measures for each method and for each dataset. For example, Table IIa-IIe are sample output generated for Dataset-I having failures from different components. We selected Dataset-I, as it has only four sensor variables for detail analysis.

Dataset-I: Using experimental results shown in Table IIa, we inferred that variable "voltage" is a candidate for leading indicator for Component 1 failures. Other features do not have strong signal in prefailure window, i.e., the time before the failure happens. Although, CUSUM based approaches have higher $WRecall$, the performance of Ruptures is considered better due to higher $WPrecision$. Similarly, Tables IIb-IId are prepared using Dataset-I for different component failures. In Dataset-I an interesting phenomenon that we observed was that the best leading indicator was different for each component, with each of the sensors seeming to monitor a specific component. For example, feature "rotation" is leading failure indicators for Component 2 related failures, whereas feature "pressure" is for Component 3 related failures, etc. Ruptures is best suited to discovering change points with reasonable precision and recall. In Table IIe, we obtain the performance evaluation with respect to all the failures. Since, assets may encounter multiple component failures, CUSUM algorithm is also competitive to Ruptures when we put multiple failures together.

Dataset-II: Table III presents the summary of results obtained for Jet engine data Dataset-II. To better utilize the space, we highlighted the best algorithm for each feature with respect to the evaluation parameter. For example, for feature "s2", Ruptures has the best $WPrecision$(0.82), $CUSUM_{max}$ has the best $WRecall$(1.0), etc. We noted that, Ruptures based change point detection method performs best for Dataset-II_1 (and also Dataset-II_3). But, CUSUM works better on Dataset-II_2 (and also Dataset-II_4). Note that, the performance on Dataset-II_2 and Dataset-II_4 is poor compared to Dataset-II_1 and Dataset-II_3. The poor performance for Dataset-II_2 and Dataset-II_4 can be validated based on dataset description given in Section V. Briefly, the engines were operated under different testing conditions for Dataset-II_2 and Dataset-II_4. This is also evident from the value of feature "setting1". The value of "setting1" stay same for Dataset-II_1 and Dataset-II_3, while in Dataset-II_2 and Dataset-II_4, the same setting changes, leading to a lot more potential change points being identified. This might hurt the precision of the Ruptures algorithm that we use. It seems to be the case that when the time series one is dealing with has a high frequency, CUSUM is a more desirable change point detection method with fewer false positives

Other Datasets: Experiments on Dataset-III suggests Ruptures achieve best results and identified three leading indicators out of 20 features. The evaluation result for the top-most leading indicator is 0.105(WPrecision), 0.494(WRecall), and 0.134(WF1). In Dataset-IV however, $CUSUM_{q75}$ based analysis seems to perform best. The top leading indicator is "Servo2", which has a WPrecision 0.204,

[6]https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/
[7]https://github.com/badlogicmanpreet/htm-drivefailures

Table II: Univariate Feature Analysis: Dataset-I

(a) Component 1 Failures

Sensor	Ruptures			CUSUM$_{min}$			CUSUM$_{max}$			CUSUM$_{q75}$		
	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1
voltage	**0.212**	**0.516**	**0.275**	0.052	0.9	0.095	0.051	0.875	0.094	-	0	0
pressure	0.036	0.078	0.044	0.034	0.631	0.063	0.035	0.626	0.064	0.043	0.003	0.003
rotation	0.033	0.084	0.042	0.033	0.646	0.06	0.033	0.654	0.062	0.047	0.001	0.001
vibration	0.039	0.086	0.048	0.036	0.667	0.066	0.035	0.657	0.065	0.033	0.013	0.013

(b) Component 2 Failures

Sensor	Ruptures			CUSUM$_{min}$			CUSUM$_{max}$			CUSUM$_{q75}$		
	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1
voltage	0.054	0.089	0.063	0.049	0.661	0.088	0.049	0.641	0.088	-	0	0
pressure	0.051	0.084	0.058	0.047	0.635	0.084	0.047	0.632	0.085	0.022	0.001	0.001
rotation	**0.276**	**0.503**	**0.33**	0.074	0.92	0.133	0.073	0.92	0.131	0.31	0.006	0.008
vibration	0.044	0.069	0.049	0.047	0.652	0.085	0.048	0.64	0.086	0.048	0.014	0.016

(c) Component 3 Failures

Sensor	Ruptures			CUSUM$_{min}$			CUSUM$_{max}$			CUSUM$_{q75}$		
	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1
voltage	0.018	0.079	0.019	0.018	0.665	0.032	0.019	0.667	0.033	-	0	0
pressure	**0.109**	**0.472**	**0.114**	0.03	0.969	0.053	0.03	0.973	0.053	0.215	0.029	0.035
rotation	0.015	0.069	0.016	0.019	0.664	0.034	0.019	0.686	0.034	0	0	0
vibration	0.017	0.074	0.017	0.019	0.676	0.034	0.019	0.698	0.034	0.014	0.013	0.007

(d) Component 4 Failures

Sensor	Ruptures			CUSUM$_{min}$			CUSUM$_{max}$			CUSUM$_{q75}$		
	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1
voltage	0.022	0.078	0.023	0.024	0.671	0.042	0.023	0.654	0.04	-	0	0
pressure	0.025	0.082	0.025	0.023	0.637	0.041	0.022	0.616	0.038	0.022	0.002	0.002
rotation	0.024	0.083	0.025	0.023	0.659	0.041	0.023	0.669	0.04	0.023	0.001	0.001
vibration	**0.126**	**0.456**	**0.133**	0.036	0.952	0.063	0.036	0.95	0.063	0.099	0.073	0.049

(e) All Component Failures

Sensor	Ruptures			CUSUM$_{min}$			CUSUM$_{max}$			CUSUM$_{q75}$		
	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1	WPrecision	WRecall	WF1
voltage	0.282	0.2111	0.224	0.134	0.739	0.217	0.134	0.715	0.216	-	0	0
pressure	0.203	0.123	0.145	0.127	0.673	0.203	0.126	0.667	0.203	0.285	0.005	0.008
rotation	**0.322**	**0.247**	**0.262**	**0.141**	**0.765**	**0.229**	**0.14**	**0.773**	**0.228**	0.333	0.002	0.004
vibration	0.207	0.127	0.149	0.13	0.705	0.211	0.13	0.698	0.21	0.179	0.022	0.032

Table III: Leading Feature Indicator Discovery on Jet Engine (Dataset-II)

(a) Dataset-II$_1$

Feature	WPrecision	WRecall	WF1
s1	-	0	0
s2	Ruptures1(0.82)	CUSUM$_{max}$(1.00)	Ruptures1(0.82)
s3	Ruptures1(0.79)	CUSUM$_{max}$(1.00)	Ruptures1(0.79)
s4	Ruptures1(0.95)	CUSUM$_{max}$(1.00)	Ruptures1(0.95)
s5	-	0	0
s6	Ruptures(0.018)	Ruptures(0.03)	Ruptures(0.015)
s7	Ruptures1(0.89)	CUSUM$_{max}$(1.00)	Ruptures1(0.89)
s8	Ruptures1(0.89)	CUSUM$_{max}$(1.00)	Ruptures1(0.89)
s9	Ruptures1(0.76)	CUSUM$_{min}$(1.00)	Ruptures1(0.76)
s10	-	0	0
s11	Ruptures1(0.99)	CUSUM$_{max}$(1.00)	Ruptures1(0.99)
s12	Ruptures1(0.96)	CUSUM$_{max}$(1.00)	Ruptures1(0.96)
s13	Ruptures1(0.88)	CUSUM$_{max}$(1.00)	Ruptures1(0.88)
s14	Ruptures1(0.88)	Ruptures(0.99)	Ruptures1(0.88)
s15	Ruptures1(0.94)	CUSUM$_{min}$(1.00)	Ruptures1(0.94)
s16	-	0	0
s17	Ruptures1(0.81)	CUSUM$_{max}$(0.97)	Ruptures1(0.81)
s18	-	0	0
s19	-	0	0
s20	Ruptures1(0.85)	CUSUM$_{max}$(1.00)	Ruptures1(0.85)
s21	Ruptures1(0.91)	CUSUM$_{max}$(1.00)	Ruptures1(0.91)

(b) Dataset-II$_2$

Feature	WPrecision	WRecall	WF1
s1	CUSUM$_{min}$(0.667)	Ruptures(0.85)	Ruptures(0.403)
s2	CUSUM$_{max}$(0.507)	CUSUM$_{min}$(0.992)	CUSUM$_{min}$(0.645)
s3	CUSUM$_{max}$(0.444)	CUSUM$_{min}$(0.992)	CUSUM$_{min}$(0.591)
s4	CUSUM$_{max}$(0.497)	CUSUM$_{max}$(0.996)	CUSUM$_{max}$(0.642)
s5	CUSUM$_{min}$(0.667)	CUSUM$_{q75}$(1.00)	CUSUM$_{q75}$(0.527)
s6	Ruptures1(0.285)	CUSUM$_{q75}$(1.00)	CUSUM$_{q75}$(0.527)
s7	CUSUM$_{q75}$(0.371)	CUSUM$_{q75}$(1.00)	CUSUM$_{q75}$(0.528)
s8	CUSUM$_{min}$(0.499)	Ruptures(0.869)	CUSUM$_{min}$(0.561)
s9	CUSUM$_{max}$(0.47)	CUSUM$_{min}$(0.988)	CUSUM$_{max}$(0.611)
s10	CUSUM$_{min}$(0.634)	Ruptures(0.862)	Ruptures(0.39)
s11	CUSUM$_{max}$(0.506)	CUSUM$_{max}$(1.000)	CUSUM$_{max}$(0.65)
s12	CUSUM$_{q75}$(0.371)	CUSUM$_{q75}$(1.00)	CUSUM$_{q75}$(0.527)
s13	CUSUM$_{max}$(0.679)	CUSUM$_{min}$(0.973)	CUSUM$_{min}$(0.748)
s14	CUSUM$_{max}$(0.42)	CUSUM$_{min}$(0.962)	CUSUM$_{max}$(0.559)
s15	CUSUM$_{max}$(0.482)	CUSUM$_{max}$(1.000)	CUSUM$_{max}$(0.633)
s16	Ruptures1(0.412)	Ruptures(0.858)	Ruptures(0.426)
s17	CUSUM$_{max}$(0.516)	CUSUM$_{max}$(0.985)	CUSUM$_{max}$(0.646)
s18	CUSUM$_{min}$(0.384)	Ruptures(0.869)	Ruptures(0.411)
s19	Ruptures(0.258)	Ruptures(0.785)	Ruptures(0.377)
s20	Ruptures1(0.296)	Ruptures(0.842)	Ruptures(0.4)
s21	Ruptures1(0.288)	Ruptures(0.835)	Ruptures(0.394)

WRecall 0.971 and WF1 0.285.

In summary, some datasets are easier, i.e., we are able to detect some signal in timeseries in a fixed window before a failure occurs. For example, Dataset-II$_1$ and Dataset-II$_3$, we are able to achieve nearly 99% precision with 99% recall. But, signal seems to be weaker in other datasets. As a result, we should expect that failure prediction in Datasets-I, III, and IV will be harder than in Dataset-II. We note that the change point detection algorithm that performs best depends on the characteristics of the dataset, and there seems to be no one algorithm that performs best in all situations.

B. Multivariate Change Point Analysis

We perform multivariate analysis on each dataset using GGM, CUSUM$_{min}$, CUSUM$_{max}$, CUSUM$_{q75}$ and Ruptures. To conduct detailed in-depth experimental analysis,

we prepared three different set of features (when feasible) for multi-variate analysis: (a) All-features, (b) Top-5 leading indicators, and (c) Top-10 leading indicators. The multivariate analysis generates evaluation measures for each method and for each feature subsets. To highlights the benefits of conducting multi-variate analysis over univariate analysis, we also used result of Top-1 leading indicators as a baseline.

Dataset-I:. We skip Top-5 and Top-10 analysis for Dataset-I, as it has only four features. To provide the summary of experimental outcome using All-features option, we selected the best method for each component failures. The best method is selected using $WF1$ score. Table IV shows the result. We noticed that the performance for Component 1, Component 3 and Component 4 is not as good as the one we used to obtain using univariate feature analysis. This is expected as the univariate analysis suggested a one to one correspondence between failure and the sensor feature. Mixing an extra additional information does not improve the result. However, for Component 2 and All-failure ground truth, the performance are comparable with univariate analysis. This surprise behavior is possibly due to dominant role of feature "rotation" or due to the majority failures are from Component 2. Note that, there is a greater correspondence between the univariate and multivariate results. In particular, there is a nearly 90% overlap between the change point generated using univariate and multivariate features.

Table IV: Multi-variate Analysis on Dataset-I

Failure Types	Best Method	WPrecision	WRecall	WF1
Component 1	CUSUM_{max}	0.051	0.116	0.053
Component 2	Ruptures/GGM	**0.276**	**0.501**	**0.329**
Component 3	CUSUM_{max}	0.034	0.159	0.032
Component 4	CUSUM_{min}	0.038	0.132	0.034
All Component	Ruptures/GGM	**0.357**	0.215	0.244

Dataset-II:. Table V presents summary of multi-variate analysis for Dataset-II_1 and Dataset-II_2. We observed that, Top-10 feature option performs better than the Top-5 and All-Feature options. Recall, the univariate analysis on Dataset-II (Table III) also suggests that nearly 10 features has good performance out of 21 sensor features. As a result, mixing top-10 leading indicators and conducting multi-variate analysis has helped to improve the performance. The performance of best multi-variate analysis method on Dataset-II_1 is comparable to univariate analysis, whereas for Dataset-II_2 the performance is improved (WF1 = 0.721). We also noticed that, GGM based method achieve 0.811 WF1 score using Top-5 features when we set the number of change points to be detected to 5.

Other Datasets: For Dataset-III, the multi-variate analysis using Top-5, Top-10 and All-Feature options does not bring any improvement compared to univarate analysis. This is due to fact that, the leading indicators are highly dependents on each other based on domain input. In multi-variate experiments on Dataset-IV, there is a marked

Table V: Multi-variate Analysis on Dataset-II

Failure Types	Best Method	WPrecision	WRecall	WF1
Dataset-II_1				
All	Ruptures	0.98	0.98	0.98
Top-5	Ruptures	0.95	0.95	0.95
Top-10	Ruptures	**0.99**	**0.99**	**0.99**
Dataset-II_2				
All	CUSUM_{max}	0.369	1	0.53
Top-5	CUSUM_{min}	0.454	1	0.611
Top-10	CUSUM_{max}	**0.592**	**1**	**0.721**

Table VI: Summary of Association Patterns

Association Patterns		Dataset	Support
Transactional	['s11', 's12']	Dataset-II_1	32
	['s21', 's5', 's6']	Dataset-II_2	62
Sequential	{'s11' $\rightarrow$ 's17'}	Dataset-II_1	43
	{['s12', 's7'] $\rightarrow$ ['s15']}	Dataset-II_3	33

improvement in the performance over univariate features. Using CUSUM_{q75}, we get a WF1 0.265 with WPrecision 0.187 and WRecall 0.99. The GGM based change point detection method is also competitive with a WF1 0.21. In summary, the complex datasets like Dataset-II and Dataset-III, the multivariate GGM method seems to boost the overall change point detection performance, suggesting that the time series cannot be simply modeled using linear method.

C. Multivariate Association Analysis

We perform frequent pattern analysis on change points generated by univariate method. First, we discuss the format of input data needed for analysis. For each failure F_i in a given dataset, we identify a list of univariate change points that precede failure F_i. To restrict the temporal locality, we impose an additional constraint saying that "duration before failure" should be less than a predefined threshold.

The frequent pattern analysis is a multivariate technique that aims to generate association relationship between discovered change points across multiple features. The two most common forms of association patterns are "**Transactional**" and "**Sequential**". In Transactional pattern, the change points in various features are observed at the same time whereas the sequential pattern has change points that are ordered in time. Table VI provides an examples of association patterns. The transactional pattern $[s11, s12]$ with Support 32 means, we observed 32 different failures in Dataset-II_1 that exhibits change in $s11$ and $s12$ at the same time. On the other hand, sequential pattern {'s11' $\rightarrow$ 's17'} with Support 43 means, we observed 43 different failures in Dataset-II_1 that exhibits change first in $s11$ and followed by change in $s17$. We have used Ruptures method to generate the change points for each feature, and then PrefixSpan algorithm [13] to generates the association patterns.

Dataset-I to Dataset-IV: We applied multivariate association analysis on all four datasets. We discover all frequent association patterns having support higher than 30 as well involve at-least two different features. Table VII shows the number of patterns that are generated per dataset. We noticed that Dataset I and IV do not generate any patterns. This

Table VII: Multivariate Analysis using All-Features

Dataset	Dat-II$_1$	Dat-II$_2$	Dat-II$_3$	Dat-II$_4$	Dat-III
# of Trans. Patterns	11	1623	30	305	9
# of Seq. Patterns	79	0	70	0	0

is expected as for this study since they are not suitable for multivariate analysis. We also noticed that, the number of transactional patterns are significantly higher than the sequential patterns. Indirectly, it suggests that change points across multiple features happen at same time.

Since association patterns are easy to understand, the multivariate association analysis provides an interpretable output of change point method.

D. Temporal Change Point Analysis

We study how early change points are discovered with respect to the failures. For example, it is possible that a leading indicator exhibits changes just few minutes prior to the failures, but another feature can detect changes much earlier but at the cost of performance. In this section, we perform temporal analysis of univariate change points. For each sensor feature S_i in dataset, we calculate average, count and deviation using "duration before failure" value.

Dataset-I to Dataset-IV: In Dataset-I with Component 1 Failure, feature "Voltage" normally change around 32 hours prior to failures. Change in other features also happen around same time, but their count is very less. Figure 5 show temporal analysis on Dataset-II$_2$. We noticed that the Setting related variables are at centered around 35 cycles before failures and remaining features are between 20-30 cycles before failures. For Dataset-III, we noticed that the leading indicator changes around 7 day before the failures, but, the another variable with slightly less accuracy change around 10 day ahead. Dataset-IV have several features change around 30 hours prior to failures. In summary, leading indicators are accurate as they are close to failures.

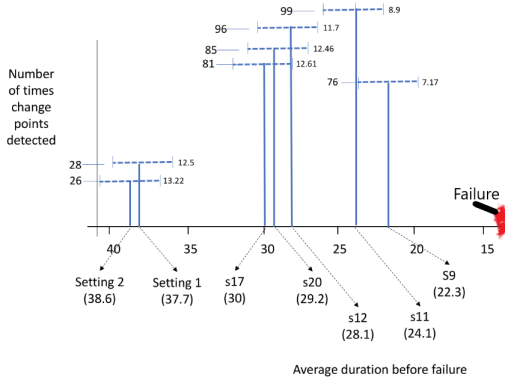


Figure 5: Temporal Analysis : Dataset-II$_2$

VII. Conclusion

In this paper, we discussed the design and development of ChieF solution pattern to be used as a part of Predictive Maintenance techniques. We explained three change point detection algorithms, namely CUSUM, Ruptures, and GGM in detail and also provided a way to extract the useful knowledge by associating the failure information. The solution is extensively tested on four different datasets and experimental results are encouraging. In the future, we would like to utilize the system to conduct the user study to validate the real usefulness of the proposed system in building knowledge model.

References

[1] "Predictive maintenance solution accelerator overview," https://docs.microsoft.com/en-us/azure/iot-accelerators/iot-accelerators-predictive-walkthrough, 2018-08-18.

[2] "GE's predictive solution," https://www.predix.io/catalog/analytics, 2018-08-18.

[3] L. Yongxiang, S. Jianming, W. Gong, and L. Xiaodong, "A data-driven prognostics approach for rul based on principle component and instance learning," in *Prognostics and Health Management (ICPHM), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–7.

[4] K. Medjaher, D. A. Tobon-Mejia, and N. Zerhouni, "Remaining useful life estimation of critical components with application to bearings," *IEEE Transactions on Reliability*, vol. 61, no. 2, pp. 292–302, 2012.

[5] S. Nussbaum, N. Liberman, and Y. Trope, "Predicting the near and distant future." *Journal of Experimental Psychology: General*, vol. 135, no. 2, p. 152, 2006.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[7] A. Zimek, E. Schubert, and H.-P. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 5, pp. 363–387, 2012.

[8] "LIME - local interpretable model-agnostic explanations," https://homes.cs.washington.edu/~marcotcr/blog/lime/, 2018-08-18.

[9] J. Krause, A. Perer, and E. Bertini, "Using visual analytics to interpret predictive machine learning models," in *2016 ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[10] A. Pettitt, "A simple cumulative sum type statistic for the change-point problem with zero-one observations," *Biometrika*, vol. 67, no. 1, pp. 79–84, 1980.

[11] C. Truong, L. Oudre, and N. Vayatis, "Ruptures: change point detection in python," *arXiv preprint arXiv:1801.00826*, 2018.

[12] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[13] "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *Proceedings of the 17th International Conference on Data Engineering*, ser. ICDE '01. IEEE Computer Society, 2001, pp. 215–.