Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Early failure detection of paper manufacturing machinery using nearest neighbor based feature extraction

Wonjae Lee[1] | Kangwon Seo*[1,2]

[1]Department of Industrial and Manufacturing Systems Engineering, University of Missouri, Columbia, MO 65211, U.S.A.
[2]Department of Statistics, University of Missouri, Columbia, MO 65211, U.S.A.

Correspondence
*Kangwon Seo, Thomas & Nell Lafferre Hall, University of Missouri, Columbia, MO 65211, Email: seoka@missouri.edu

Abstract

In a paper manufacturing system, it can be substantially important to detect machine failure before it occurs and take necessary maintenance actions to prevent a detrimental breakdown of the system. Multiple sensor data collected from a machine provides useful information on the system's health condition. However, it is hard to predict the system condition ahead of time due to the lack of clear ominous signs for future failures, a rare occurrence of failure events, and a wide range of sensor signals which might be correlated with each other. In this paper, we present two versions of feature extraction techniques based on the nearest neighbor combined with machine learning algorithms to detect a failure of the paper manufacturing machinery earlier than its occurrence from the multi-stream system monitoring data. First, for each sensor stream, the time series data is transformed into the binary form by extracting the class label of the nearest neighbor. We feed these transformed features into the decision tree classifier for the failure classification. Second, expanding the idea, the relative distance to the local nearest neighbor has been measured, results in the real-valued feature, and the support vector machine is used as a classifier. Our proposed algorithms are applied to the dataset provided by IISE 2019 data competition, and the results show the better performance than the given baseline.

KEYWORDS:
Multi-stream time series classification, relative distance, rare event prediction, 1-nearest neighbor, feature extraction

1 | INTRODUCTION

The pulp and paper production requires highly complex and integrated processes by chemical or mechanical means, which include wood preparation, pulping, chemical recovery, bleaching, and papermaking[1]. In the advanced papermaking facilities, the systems are continuously monitored so that the operators can manage and control the processes, and detect any possible incidents that might cause an abrupt production break. To do this, a wide range of sensors are deployed in many different parts of manufacturing equipment to measure important process variables and monitor the system status. These sensors generate large amounts of multi-stream measurements. For instance, the motivated dataset for this research contains system monitoring measurements captured by 61 different sensors located in a paper manufacturing machinery[2]. These raw measurements ought to be processed and analyzed appropriately to obtain useful information regarding the system's health condition. The general

purpose of this paper is to develop a practical pipeline to process, analyze, and interpret the system monitoring data given as a form of the multi-stream time series to detect a system failure that may be occurred in the near future.

One challenge problem that we aim to resolve through this project is that the machine failure has to be prognosed ahead of a physical occurrence. The traditional system monitoring tools such as the control chart based quality control techniques focus on the detection of the assignable causes of the system abnormal status as soon after it occurs as possible. As such, the average run length (ARL) has been used as the main performance metric for comparing various types of control charts[3]. In the paper manufacturing process, however, once the machine failure occurs, the system instantly stops and there is no benefit to detect the failure afterward. Therefore, it is important to perceive any symptomatic signal followed by a machine breakdown even a few seconds earlier. To achieve this goal, we define our problem as a binary classification task where we aim to distinguish the precursory signs from the normal signals. This problem definition motivated us to use the terminology “multi-stream time series (MSTS)" rather than “multivariate time series" as the multivariate data implies multiple responses in the statistical literature.

Other difficulties for this task may be attributed to the multi-stream nature of the given data and a lack of failure-labeled observations. Although there exist several feature-based classification algorithms for the time series data, it is problematic to generate and select a proper set of features from high dimensional multi-stream data. As an alternative, the deep learning techniques are emerging as competent tools handling such data[4,5]. However, these techniques require a substantially large amount of data, which is not the case for our problem where the dataset only includes 124 machine breakdown points among more than 18, 000 time points. Such an extremely imbalanced dataset makes it even harder to build a model with high performance.

To solve the aforementioned problems, we rely on machine learning algorithms, which have been recognized as more powerful techniques for predictive tasks than traditional approaches which do not incorporate these techniques[5], with properly processed variables and informative features. Specifically, for each sensor or variable, we transform the time series instance into a scalar extracted from its nearest neighbor and feed the transformed variables into a proper off-the-shelf machine learning algorithms to make a classification. The nearest neighbor based algorithm has been recognized as one of the most effective classification methods for time series data[6]. In this paper, we exploit the advantages of 1-nearest neighbor but extend the method for MSTS data. The objectives of these algorithms are to extract suitable features for MSTS classification. First, we extract the class label of the nearest neighbor only considering a single variable, which results in the binary features for each variable. Second, the relative distance to the nearest neighbor is measured, which is anticipated to provide more useful information on an instance’s nearest neighbor. In this research, we demonstrate how to predict the paper machine failure before it occurs (i.e., early detection); and find the variables which have a significant effect on causing failures using these nearest neighbor based features.

The rest of this paper is organized as follows. Section 2 reviews related work in time series classification. Section 3 shows the overall procedure to implement, and describes dataset, preprocessing and two versions of algorithms we propose in this paper. In Section 4, we evaluate the performance of the proposed algorithms with the real-world dataset of the paper manufacturing sensor signals. Finally, we conclude our research in Section 5.

2 | RELATED WORK

A wide range of algorithms have been used and proposed to solve classification problems with univariate time series data. Sykacek and Roberts[7] propose an approach with a latent feature representation by applying Bayesian theory to hierarchical time series processing. Esmael et al.[8] suggest a hybrid approach to improve the accuracy of the time series classifier with hidden Markov models (HMM). Jović et al.[9] examine the capability of four common decision tree ensembles in biomedical time-series datasets. Eads et al.[10] employee a support vector machine (SVM) for time series classification with features extracted from the time series data. Cui et al.[11] demonstrate convolutional neural networks for time series classification problem to incorporate feature extraction and classification in a single framework. These algorithms have been employed as a single classifier or a combination of multiple methods, sometimes called an ensemble, to improve the performance of classification[12]. Although the ensemble-based classifier is known as a prominent algorithm for time series classification tasks[13], it requires much computation for training, which may not be suitable for a large dataset. Meanwhile, Tan et al.[14] describe that the nearest neighbor classifier based on the Euclidean distance is a fast and promising classification algorithm when it comes to the big dataset.

Recently, MSTS data has gained great attention, and many researchers have proposed new methods to solve the multi-stream based problem. Orsenigo and Vercellis[15] describe a classification method based on a temporal extension of discrete SVMs with the notions of warping distance and softened variable margin in the set of multivariate input sequences. Weng and Shen[16] implement a new approach for MSTS classification. The eigenvectors of row-row and column-column covariance matrices of

MSTS samples are calculated to extract features and 1-nearest neighbor classifier is used for the classification. The authors show that distance-based methods with 1-nearest neighbors are an effective way to classify MSTS. Other algorithms have also been used to deal with MSTS. Zhang et al.[17] address the challenges of MSTS data by presenting a real-time multiple profiles sensor-based process monitoring system.

Feature extraction is considered as one of the popular techniques for MSTS classification. Rodríguez and Alonso[18] use the boosting algorithm to generate new features and a SVM is applied with these metafeatures. Kadous and Sammut[19] seek to generate classifiers that are comprehensible and accurate with metafeatures. The authors describe applications of the sign language recognition and the electrocardiogram (ECG) signal classification. C. Li et al.[20] suggest feature vector selection approaches for MSTS classification using singular value decomposition (SVD).

Profile monitoring techniques with the use of the principal component analysis (PCA) method is another way to manage MSTS. Kim et al.[21] develop the method to detect profile changes of multi-stream tonnage signals for forging process monitoring and to classify fault patterns while Chang and Yadama[22] propose a statistical process control framework to monitor non-linear profiles to identify mean shifts in a profile with discrete wavelet transformation (DWT) and B-splines. Paynabar et al.[23] suggest a multi-way extension of the PCA technique to classify multi stream profile data. Grasso et al.[24] suggest multi-way PCA to deal with the reduction of data dimensionality and the fusion to all the sensor outputs. This paper carries out two main multi-way extensions of the traditional PCAs to handle MSTS.

Deep learning has provided prominent results for this application with the popularity of the neural networks. Zheng et al.[25] propose a deep learning framework for MSTS classification using features extracted by 1-nearest neighbor with dynamic time warping (DTW). Karim et al.[4] utilize the long short term memory fully convolutional network (LSTM-FCN) and attention LSTM-FCN for MSTS classification. Wang et al.[5] utilize recurrent neural network (RNN) and adaptive differential evolution (ADE) algorithm for the same task. Despite the popularity of deep learning, this technique requires high volume of dataset, and it is not suitable to our problem due to a lack of labeled data.

3 | MATERIALS AND METHODS

3.1 | Dataset description

The dataset was provided by the Institute of Industrial and Systems Engineers (IISE) 2019 data competition, which recorded real sensor observations from a paper manufacturing process[2]. Many different types of data are collected over a period of time using a variety of sensors located on the machines. Some sensors measure raw materials (e.g., amount of pulp fiber, chemicals, and so on) and the others represent process variables (e.g., blade type, couch vacuum, rotor speed, etc). Overall, 61 different sensor signals are collected, and one month of monitoring data are recorded at every 2 minute for a paper manufacturing machine, which results in the dataset of 61 streaming signals at 18,398 time points. In addition, for each time point, the system condition (i.e., normal or break) has been recorded in a binary response variable. Despite such a large number of measurements, the failures only occur at 124 time points (0.67% of total observations) during operation and this characteristic of the rare event makes it hard to predict the failure before it occurs. Table 1 summarizes the dataset. Data-driven approach is used to for this problem instead of incorporating physical models, since no information was given regarding sensor information and domain knowledge.

TABLE 1 Dataset description

Element		Value	Remark
Number of variables	Continuous variables	59	$\mathbf{s}_1 \sim \mathbf{s}_{27}, \mathbf{s}_{29} \sim \mathbf{s}_{60}$
	Categorical variables	2	$\mathbf{s}_{28}$ (8 categories), $\mathbf{s}_{61}$ (2 categories)
Number of measurements	Normal	18,274	recorded by every 2 minute
	Abnormal (failure)	124	

Predicting failures for a pulp-and-paper mill is critical because a break has a significant impact on the entire process. Even though paper breaks rarely take place during operation, only one failure causes a significant loss of time and labor for identifying a cause of the failure and replacing any broken parts. Once the machine fails, the entire process should be stopped since the operation needs to be halted until the problem is found and fixed. This maintenance procedure would take more than an hour

which would incur a substantial amount of cost. It indicates that only a small amount of failure reduction through early detection could give a significant amount of cost savings for industries.

3.2 | Procedure

The overall procedure of the proposed algorithms in this paper is presented in Figure 1 consisting of preprocessing, CL-LNN and RD-LNN with other machine learning techniques. The original MSTS dataset is prepocessed before carrying out two types of feature extraction methods and these features are fed into decision tree or SVM based on the extracted data types for early failure detection. More detailed information is described in the following sections.

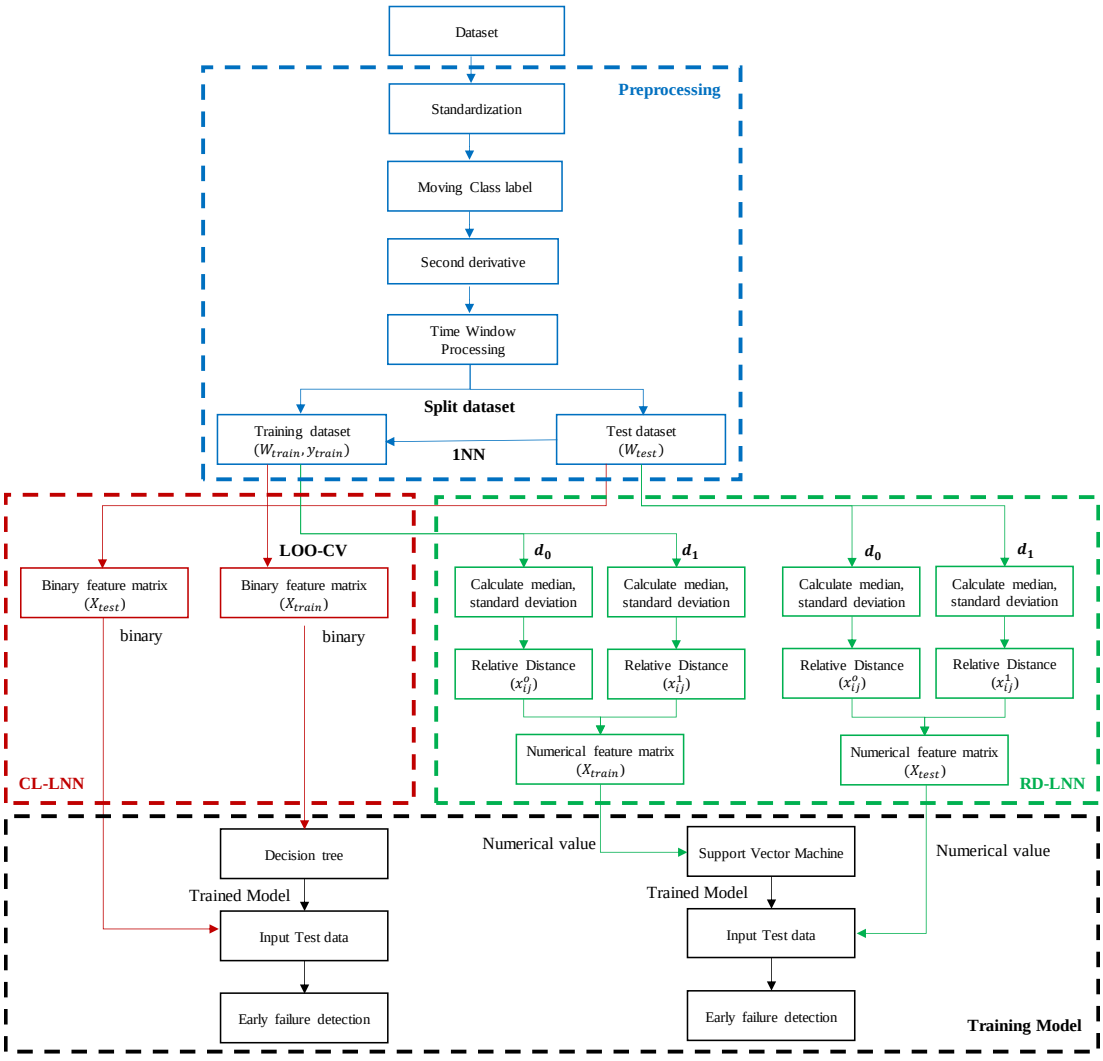

FIGURE 1 Flow chart of the proposed method for early failure detection

3.3 | Data preprocessing

The MSTS data obtained from the paper manufacturing machinery is given as

$$\mathbf{MSTS} = (\mathbf{s}_1\ \mathbf{s}_2\ \cdots\ \mathbf{s}_p\ \mathbf{c}) = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,p} & c_1 \\ s_{2,1} & s_{2,2} & \cdots & s_{2,p} & c_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ s_{T,1} & s_{T,2} & \cdots & s_{T,p} & c_T \end{pmatrix} \tag{1}$$

where $s_{t,j}$'s, $t = 1, \ldots, T, j = 1, \ldots, p$, are sensor signals measured at the time point t from the j-th sensor, $T = 18,274$ is the number of measurement time points, $p = 61$ is the number of variables by different sensors, and c_t's are records of the system condition for each time point of measurement (i.e., $c_t = 0$ for normal, and $c_t = 1$ for break). This sensor information is preprocessed to implement the classification algorithms. First, data standardization is conducted for each variable. To do this, each measurement is scaled by subtracting the corresponding mean and then being divided by the standard deviation so that the mean becomes 0 and the standard deviation 1, as follows.

$$s_{t,j} \leftarrow \frac{s_{t,j} - mean(\mathbf{s}_j)}{std(\mathbf{s}_j)}, \quad j = 1, \ldots, p \tag{2}$$

where the notation $\leftarrow$ indicates that the variable in the left-hand side is replaced with the new variable of the right-hand side, $mean(\mathbf{s}_j)$ and $std(\mathbf{s}_j)$ are the mean and standard deviation, respectively, of the original measurement data from the j-th sensor.

In this project we aim to detect the failure earlier before it occurs. One simple way to achieve this goal is to use the class label of k time points ahead for the current instance's class label so that classifiers are able to learn to predict c_{t+k} the system condition at k time units ahead[2].

$$c_t \leftarrow c_{t+k}, \text{ where } k = 1, 2, \ldots \tag{3}$$

We set $k = 1$, which implies that we build a model to detect a failure two minutes earlier than its occurrence. Figure 2 depicts this process.

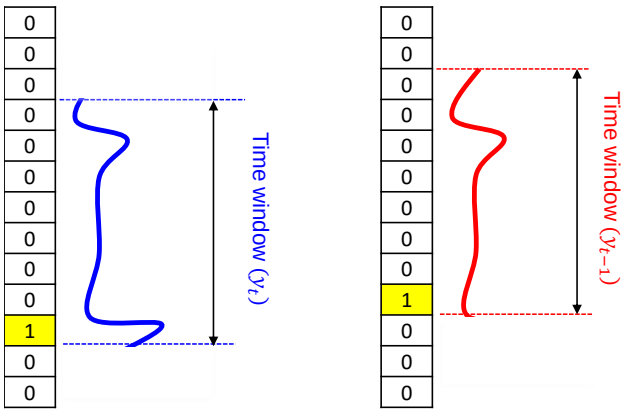

FIGURE 2 Early detection by moving column **c** by 1 row

The derivative then is applied to sense sudden changes in the sensor signals. The derivative in the time series is the difference between all neighboring points in one dimension. That is,

$$s'_{t,j} = |s_{t,j} - s_{t-1,j}|, \quad s''_{t,j} = |s'_{t,j} - s'_{t-1,j}|, \quad j = 1, \ldots, p \tag{4}$$

where $s'_{t,j}$ and $s''_{t,j}$ represent the first and second derivatives of $s_{t,j}$, respectively. The first derivative is related to a gradual change in time series which may not be sensitive to a sudden machine breakdown, while the second derivative is more useful to detect sharp changes appeared in the streaming signals. For the rest of this paper, we use the second derivative to seize precursors of an immediate failure.

In classification problems with streaming data, temporal sequence data can normally secure more information compared to the data point sampled at a single time step[26]. Accordingly we extract small fragments of sequences by conducting, namely, time window processing. For a given window size m, a window instance consists of the last m sensor measurements up to time t which corresponds to the rows of MSTS data given in Eq. (1) with time indices $t-m+1, \dots, t$. The class label of the window instance is given as c_t so that it represents the system condition at the last time point of the window. These window instances provide features to be used in a machine learning algorithm. In addition, we address the problem of severely imbalanced class labels of the original MSTS data while constructing the window instances by making a balance between two labels to some extent. That is, for time window processing we select all the time points t where $c_t = 1$ and only randomly select t where $c_t = 0$ such that it makes difference between the number of class labels not too large. The constructed window instances and those class labels are given as the following form.

$$(\mathbf{W} \;\; \mathbf{y}) = \begin{pmatrix} \mathbf{w}'_{1,1} & \mathbf{w}'_{1,2} & \cdots & \mathbf{w}'_{1,p} & y_1 \\ \mathbf{w}'_{2,1} & \mathbf{w}'_{2,2} & \cdots & \mathbf{w}'_{2,p} & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{w}'_{n,1} & \mathbf{w}'_{n,2} & \cdots & \mathbf{w}'_{n,p} & y_n \end{pmatrix} \tag{5}$$

where each row represents each window instance. That is, $\mathbf{w}_{i,j}$ is the sequence of length m which is the second derivatives of the j-th sensor signals, and y_i is the class label of the i-th window instance. Note that the row index $i = 1, \dots, n$ merely distinguishes each window instance, not necessarily implies the time point.

The time window processed dataset is then splitted into training and test datasets, which are called $(\mathbf{W}_{train} \;\; \mathbf{y}_{train})$ and $(\mathbf{W}_{test})$. These multi-stream window instances are used as input of Algorithm 1 and 2 respectively.

3.4 | 1-nearest neighbor for time series classification

In the field of data mining and machine learning, one of the most frequently studied problems is classification[27]. The classification process is to evaluate the similarities in a dataset to classify them into designated classes. One of the differences between time series classification problem and traditional classification problems is that the attributes are arranged in order and input features may be correlated. The 1-nearest neighbor (1-NN) is a popular classifier for the time series classification as its performance can compete with the most complex classifiers[6]. When a new observed time series instance comes out, the 1-NN classifier looks for the instance in the training dataset which has the shortest distance with the new instance, and predict the class of the new instance as the class label of the closest instance. A distance measure such as the Euclidean distance is used to compare two time series instances. For a one dimensional time series data, the Euclidean distance between two time series instances $\mathbf{w}_i$ and $\mathbf{w}_k$ is measured by

$$D_{ED}(\mathbf{w}_i, \mathbf{w}_k) = \sqrt{\sum_{t=1}^{m} (w_{it} - w_{kt})^2} \tag{6}$$

where $\mathbf{w}_i$ and $\mathbf{w}_k$ are window instances with $t = 1, \dots, m$ measurements to be compared each other.

The other renowned distance measure for time series data is dynamic time warping (DTW), which is the method to find the optimal alignment between two time-dependent sequences. It has been widely used in the field of pattern recognition and broadly tested on the benchmark time series data. DTW is originally designed to compare different speech patterns for the purpose of automatic speech recognition to solve the problem of distortions in the time axis[28]. It makes a time series stretched and realigned to better match the other time series[29]. To find the DTW distance, the matrix $\mathbf{M}$ is built where the (t, t')-th element of $\mathbf{M}$ is $d(w_{it}, w_{kt'}) = (w_{it} - w_{kt'})^2$. Then a warping path is defined as the monotonically increasing sequences of indices

$\mathbf{p} = \{(0,0), \ldots, (t,t'), \ldots, (m,m)\}$. The DTW distance can be found by the warping path which has the minimum cumulative distance between two sequences.

$$D_{DTW}(\mathbf{w}_i, \mathbf{w}_k) = \min_{\mathbf{p}} \sqrt{\sum_{h=1}^{H} \mathbf{M}_h} \tag{7}$$

where H is the length of the warping path, $\mathbf{M}_h$ is the matrix element corresponding to the h-th element of a warping path $\mathbf{p}$[30]. Figure 3 depicts how Euclidean matching and DTW matching compare similarities between two time series instances. In brief, Euclidean distance measures the distance between the two waves regardless of the shapes, while the DTW measures the distance by taking into account the shapes of two sequences. However, due to its computational complexity of DTW, the distance measurement with the DTW may not be suitable to be applied for the real time sensor streaming data in which it is required to find the nearest neighbor instance quickly.

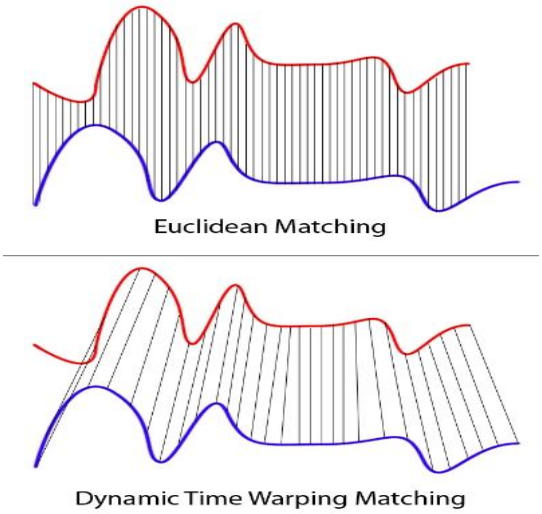

FIGURE 3 1-NN comparison between Euclidean and DTW matching

To select the appropriate distance measure between Euclidean distance and DTW distance, a separate experiment is conducted to compare the performance, of which the result is shown in Section 4. For our proposed algorithms, Euclidean distance is used to measure the distance between two time series instances as the experiment shows that Euclidean distance requires much less time than DTW without a significant difference of performances between two methods.

3.5 | Nearest neighbor based feature extraction

One possible way to extend the 1-NN for a single-stream time series data to the case of multi-stream signals could be to use the sum of the Euclidean distance measured by Eq. (6) for each variable to measure a similarity between two multi-stream window instances. In this case, however, information of all variables is aggregated, that results in the loss of each variable's information and relationships between variables. Instead, we look for the nearest neighbor considering each variable only, which we call the *local* nearest neighbor (i.e., the nearest neighbor in an embedded space of a single stream), and extract scalar features from it. These features are fed into different classification algorithms depending on the types of extracted features.

The first feature we propose is the class label of the local nearest neighbor (CL-LNN), which is given as 0 or 1 for each variable. Algorithm 1 outlines the procedure of the CL-LNN feature extraction in which the MSTS data is converted into binary feature matrices $\mathbf{X}_{train}$ and $\mathbf{X}_{test}$. The local nearest neighbor is found by leave-one-out cross validation (LOO-CV) for each variable on the training dataset $\mathbf{W}_{train}$. That is, for an instance of the training dataset, LOO-CV searches all the other instances in the training dataset except itself and chooses the one that gives the highest matching with it, which is simple but effective for 1-NN[31]. On the

Algorithm 1 CL-LNN feature extraction

Input: Multi-stream window instances $\mathbf{W}_{train}$ for training, and $\mathbf{W}_{test}$ for testing, class labels of training instances $\mathbf{y}_{train}$, index set of training data $Train$, index set of test data $Test$
Output: Binary feature matrices $\mathbf{X}_{train}$ with elements $x_{ij}, i \in Train$, and $\mathbf{X}_{test}$ with elements $x_{ij}, i \in Test$

for $i \in$ Train **do**
  for $j \in \{1, \dots, p\}$ **do** $d^* = L$
    L is a large number used for a initialization
    for $k \in Train \setminus i$ **do** $d = D_{ED}(\mathbf{x}_{ij}, \mathbf{x}_{kj})$
      for all instances in training data except itself (LOO-CV)
      if $d \leq d^*$ **then** $k^* \leftarrow k$, $d^* \leftarrow d$
      end if
    end for$x_{ij} \leftarrow y_{k^*}$
    store class label of the local nearest neighbor as a feature
  end for
end for

for $i \in$ Test **do**
  for $j \in \{1, \dots, p\}$ **do** $d^* = L$
    L is a large number used for a initialization
    for $k \in Train$ **do** $d = D_{ED}(\mathbf{x}_{ij}, \mathbf{x}_{kj})$
      for all instances in training data
      if $d \leq d^*$ **then** $k^* \leftarrow k$, $d^* \leftarrow d$
      end if
    end for$x_{ij} \leftarrow y_{k^*}$
    store class label of the local nearest neighbor as a feature
  end for
end for

other hand, for an instance of the test dataset, the algorithm simply searches the nearest neighbor from the training dataset. The nearest neighbor is found by computing Euclidean distance based on the time window for each variable as in Eq. (6). Note that, for a given instance, the CL-LNN features for different variables may be varied because the nearest neighbor for each variable could be different. These binary features are fed into the decision tree classifier for the model training and prediction, which will be described in more detail in Section 3.6. The features can keep the original information of each sensor signals by considering each variable separately, and correlations between different variables are expected to be handled by the decision tree algorithm.

Another feature we propose in this paper is called the relative distance of the local nearest neighbor (RD-LNN). While the CL-LNN can be thought of as features of hard classification where the outcome is certainly given as 0 or 1, RD-LNN provides features of soft classification, which can be seen as probability-like features. Although the binary feature extracted from the CL-LNN provides information on which one is the most closest to the instance under consideration, it is not able to measure the degrees of significance or strength of the extracted feature. Let us consider 3 cases to classify the label of instances with nearest neighbor based feature extraction in Figure 4. In the first case, there is a clear decision boundary which makes it easy to separate two distinct groups where CL-LNN might show the superior performance. However, outliers in the second example make it more challenging to classify the target instance. Suppose we know the class label of the local nearest neighbor for a given instance is, say, 1 a break signal. To build a robust prediction model, we may also want to know how much reliable and accurate this signal is. For the second case, even if the nearest neighbor is the break signal, this nearest neighbor is an outlier with respect to the majority of the other break signals. In this case, relying solely on the class label of the nearest neighbor may be risky. To complement this pitfall of binary features, we may consider to measure distances from the nearest neighbor to the other instances respectively. If the distance value is large, the nearest neighbor is thought to be located far from the majority of its same class and does not provide reliable information. Whereas if the distance is small, the nearest neighbor is thought to represent the group of the same class and the information provided by this instance is more accurate. In lieu of direct distance

measure, we use probability measure which is similar to the computation of p-value for a statistical hypothesis testing. Rare events (i.e., breaks) in our dataset, however, appear to be indistinguishable from the other which is ambiguous to differentiate those two groups as in the third case. In this situation, we found that it is more effective to measure the relative distance for each group respectively instead of applying the same nearest neighbor to different groups. Specifically, given an instance of which the class label has to be predicted, Euclidean distances to all the other instances in the training dataset are computed. For each class label ($y = 0, y = 1$), the nearest neighbors are found. Let d_0^* and d_1^* be distances to nearest neighbors with class label 0 and 1, respectively. We can also find the approximated normal distribution for each class. Let X_0 and X_1 be random variables with these approximated normal distributions. The RD-LNN features are computed by $P(X_0 \leq d_0^*)$ and $P(X_1 \leq d_1^*)$ for each class, which can be interpreted as the probability that an observation is located farther than the nearest neighbor.

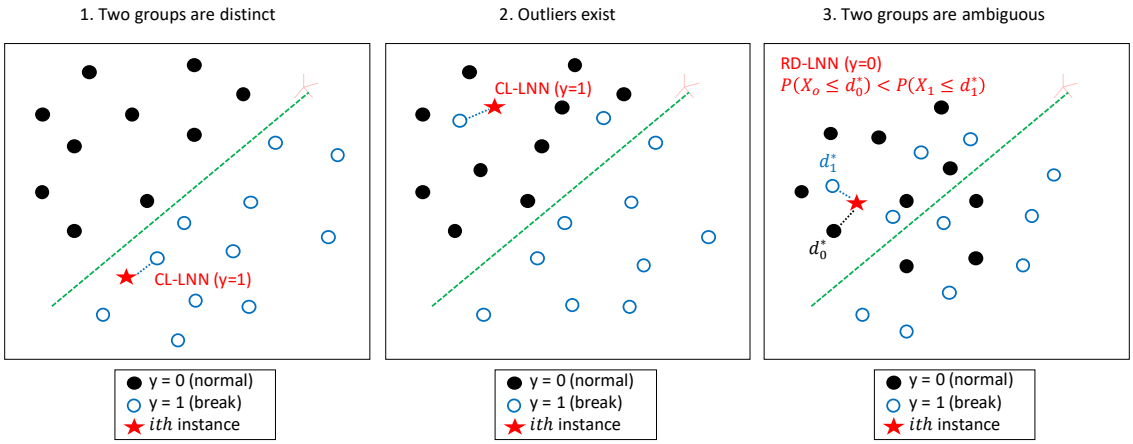

FIGURE 4 Three different cases based on nearest neighbor based feature extraction

Algorithm 2 describes the procedure in which the algorithm generates the numerical values by measuring the probability representing the relative position of the nearest neighbor for each class compared to the other instances with the same class label. We found that this unique feature extraction technique improves the performance of classification when it comes to another machine learning algorithm, SVM. Note that two features are extracted from each variable corresponds to each class label ($y = 0, y = 1$). The normal distribution is approximated to distance data where the mean and standard deviation are set as the sample median and the standard deviation of distances included in the interquartile range (i.e., data ranging from the first quartile $Q1$ to the third quartile $Q3$) to minimize the effect of outliers.

3.6 | Training model

Different machine learning techniques are used for CL-LNN and RD-LNN respectively to train the model and predict failures two minute earlier based on the data type that algorithms produce. First, C5.0 decision tree algorithm which is an improved version of its predecessor C4.5 is applied to the CL-LNN algorithm for the classification between normal and abnormal condition. In order to improve model performance, we implemented $adaptiveboosting$ which is the process in which many trees are built and trees vote for the best class. A $costmatrix$ is also employed by assigning a penalty to different types of errors to improve the accuracy where 1 is assigned for the false positive, and 5 is chosen for the false negative since failing to detect breaks can be more expensive mistake. Second, the numerical feature matrix ($\mathbf{X}_{train}$) from the training dataset ($\mathbf{W}_{train}$) is fed into SVM to generate the model, and the other matrix ($\mathbf{X}_{test}$) is used to evaluate the performance of the model generated with training dataset ($\mathbf{W}_{test}$) from RD-LNN. To train SVM model, the function of $kernel$ which takes data as input and transforms it into the required form for training and predicting is chosen to be $radial$. The $cost$ is assigned to 1 to trade off the correct classification of training examples against maximization of the decision function's margin. 0.5 is also used for $gamma$ parameter which defines how far the influence of a single training example reaches. These parameters are selected heuristically by experiments based on our dataset.

Algorithm 2 RD-LNN feature extraction

Input: Multi-stream window instances $\mathbf{W}_{train}$ for training, and $\mathbf{W}_{test}$ for testing, class labels of training instances $\mathbf{y}_{train}$, index set of training data $Train$, index set of test data $Test$
Output: Numeric feature matrices $\mathbf{X}_{train}$ with elements x_{ij}^0 and x_{ij}^1, $i \in Train$, and $\mathbf{X}_{test}$ with elements x_{ij}^0 and x_{ij}^1, $i \in Test$
for $i \in$ Train **do**
  for $j \in \{1, \ldots, p\}$ **do** $\mathbf{d}_0 = \mathbf{d}_1 = NULL$
    initialize arrays to store distance values
    for $k \in Train \setminus i$ **do** $d = D_{ED}(\mathbf{x}_{ij}, \mathbf{x}_{kj})$
      for all instances in training data except itself (LOO-CV)
      if $y_k = 0$ **then** append d to $\mathbf{d}_0$
      end if
      if $y_k \neq 0$ **then** append d to $\mathbf{d}_1$
      end if
    end for
    extract features from distances with label 0
    $d_0^* \leftarrow \min(\mathbf{d}_0)$
    distance to the nearest neighbor with label 0
    $\mathbf{q}_0 \leftarrow interquartile(\mathbf{d}_0)$
    $\mu_0 \leftarrow median(\mathbf{q}_0)$, $s_0 \leftarrow stdev(\mathbf{q}_0)$
    $x_{ij}^0 \leftarrow P(X \leq d_0^*)$, where $X \sim N(\mu_0, s_0^2)$
    extract features from distances with label 1
    $d_1^* \leftarrow \min(\mathbf{d}_1)$
    distance to the nearest neighbor with label 1
    $\mathbf{q}_1 \leftarrow interquartile(\mathbf{d}_1)$
    $\mu_1 \leftarrow median(\mathbf{q}_1)$, $s_1 \leftarrow stdev(\mathbf{q}_1)$
    $x_{ij}^1 \leftarrow P(X \leq d_1^*)$, where $X \sim N(\mu_1, s_1^2)$
  end for
end for
for $i \in$ Test **do**
  for $j \in \{1, \ldots, p\}$ **do** $\mathbf{d}_0 = \mathbf{d}_1 = NULL$
    initialize arrays to store distance values
    for $k \in Train$ **do** $d = D_{ED}(\mathbf{x}_{ij}, \mathbf{x}_{kj})$
      for all instances in training data except itself (LOO-CV)
      if $y_k = 0$ **then** append d to $\mathbf{d}_0$
      end if
      if $y_k \neq 0$ **then** append d to $\mathbf{d}_1$
      end if
    end for
    extract features from distances with label 0
    $d_0^* \leftarrow \min(\mathbf{d}_0)$
    distance to the nearest neighbor with label 0
    $\mathbf{q}_0 \leftarrow interquartile(\mathbf{d}_0)$
    $\mu_0 \leftarrow median(\mathbf{q}_0)$, $s_0 \leftarrow stdev(\mathbf{q}_0)$
    $x_{ij}^0 \leftarrow P(X \leq d_0^*)$, where $X \sim N(\mu_0, s_0^2)$
    extract features from distances with label 1
    $d_1^* \leftarrow \min(\mathbf{d}_1)$
    distance to the nearest neighbor with label 1
    $\mathbf{q}_1 \leftarrow interquartile(\mathbf{d}_1)$
    $\mu_1 \leftarrow median(\mathbf{q}_1)$, $s_1 \leftarrow stdev(\mathbf{q}_1)$
    $x_{ij}^1 \leftarrow P(X \leq d_1^*)$, where $X \sim N(\mu_1, s_1^2)$
  end for
end for

4 | RESULTS OF EXPERIMENT

4.1 | Performance analysis

In this research, four metrics are used to evaluate the performance of the proposed classification algorithms. *Precision* (also known as the positive predictive value) is defined as the proportion of correctly predicted instances over the total number of instances predicted as being positive. *Recall* (also known as sensitivity) is the proportion of positive instances detected among the total number of positive instances. Also, *False positive rate* refers to the probability of falsely rejecting the null hypothesis for a particular test. Since, however, the distribution of class labels is highly skewed, another performance metric *F − measure* has been used to measure the performance of a rare classification problem. F-measure (also sometimes called the F1 score or F-score) is the combination of precision and recall using the harmonic mean, a type of average being used for rates of change. Table 2 shows the prediction result of CL-LNN and RD-LNN respectively in the form of a confusion matrix. This table indicates that there are 1,796 normal states, while only 25 failures are included in the whole test dataset. Based on the confusion matrix, RD-LNN presents better performance than CL-LNN which detects one more failure while false positive is decreased by eight instances.

TABLE 2 Confusion Matrix

	CL-LNN		RD-LNN		Remark	
Prediction	0	1	0	1	0	1
0	1750	21	1758	20	(True Negative)	(False Negative)
1	46	4	38	5	(False Positive)	(True Positive)

A comparison of performance metrics in Table 3 makes it even more clear. Note that all four metrics are improved in RD-LNN compared to CL-LNN. Moreover, F-measure has been significantly improved by 36% which is a major breakthrough considering F-measure 0.11 is one of the top scores in the IISE 2019 data competition. This indicates that only a small number of increase in failure detection can make enormous contributions to the performance in the rare event scenario.

TABLE 3 Performance with four metrics

Item		Description	CL-LNN	RD-LNN
Precision	(Positive predictive value)	$TP/(TP+FP)$	0.08	0.12
True positive rate	(Sensitivity or Recall)	$TP/(TP+FN)$	0.16	0.2
False positive rate	(Specificity)	$FP/(FP+TN)$	0.03	0.02
F-measure		$(2\times TP)/(2\times TP+FP+FN)$	0.11	0.15

Another metric used to measure the performance is a receiver operating characteristic curve, or ROC curve, which represents the diagnostic ability of a binary classifier. This tool is suitable to visualize and compare the performance of our proposed algorithms. The true positive rate (TPR or sensitivity) is plotted in the ROC curve against the false positive rate (FPR or specificity) at different threshold settings to exhibit how much model is able to distinguish classes. In Figure 5, ROC curves of CL-LNN and RD-LNN are plotted together to compare the performance. It shows that there is a significant difference in prediction power between them. The ROC curve of RD-LNN represents way better performance compared to CL-LNN which is approaching the 45-degree diagonal line as FPR increases. It shows that RD-LNN is more robust to the noise in the rare event situation.

Additional experiment is conducted separately to choose distance measure algorithm between Euclidean distance and DTW distance. In the experiment comparing two methods to measure the distance, we found that DTW distance spends much more time to complete the same task than Euclidean distance which takes less than 5 minutes while it shows the almost same performance. Based on the result in Table 4, Euclidean distance correctly detected four failures and DTW distance found six failures while Euclidean distance shows the higher F-measure than DTW distance. The reason why Euclidean distance performs better

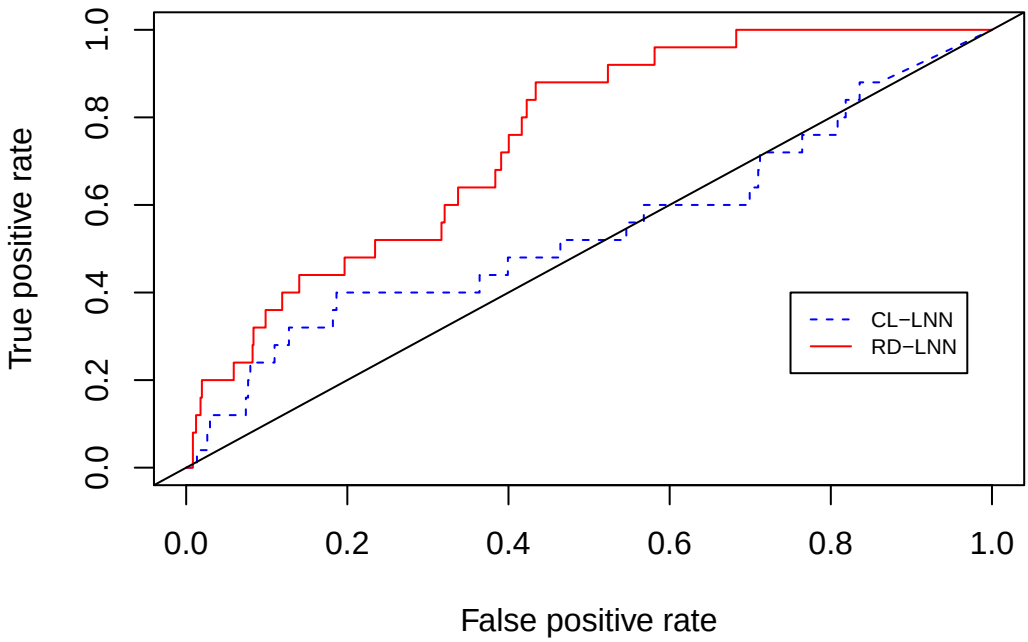

FIGURE 5 ROC curves of CL-LNN & RD-LNN

in this experiment is that DTW distance is particularly well suited for the application of automatic speech recognition in which speaking speeds vary based on time. However, time-series data that has been used here has the same time difference.

TABLE 4 Comparison between Euclidean and DTW distance

Item	Euclidean distance	DTW distance	Remark
TN (True Negative)	1746	1672	
FN (False Negative)	21	19	
TP (True Positive)	4	6	Detect failures
FP (False Positive)	50	124	
F-measure	0.10127	0.07742	$(2 \times TP)/(2 \times TP + FP + FN)$

4.2 | Root cause analysis

Root cause analysis is implemented by measuring the importance of each variable to find the critical ones which cause the failure of paper manufacturing machinery based on the decision tree algorithm. The variable importance is estimated based on the percentage of training dataset samples that fall into all the terminal nodes after the split to find the root cause. In Table 5, 61 variables are listed in the order of importance from 1 to 61. Four variables ($\mathbf{s}_3$, $\mathbf{s}_{32}$, $\mathbf{s}_{40}$, and $\mathbf{s}_{43}$) which have the importance 100% are the most important variables to detect failures earlier than its occurrence. In other words, these four variables have the most impact on the classification model. One interesting fact is that a categorical variable ($\mathbf{s}_{28}$) and a binary variable ($\mathbf{s}_{61}$) do not make any contributions to this model.

TABLE 5 Root cause analysis through decision tree algorithm

Rank	Variable	Importance	Rank	Variable	Importance	Rank	Variable	Importance
1	$\mathbf{s}_3$	100%	21	$\mathbf{s}_{26}$	62%	41	$\mathbf{s}_{48}$	27%
2	$\mathbf{s}_{32}$	100%	22	$\mathbf{s}_{12}$	58%	42	$\mathbf{s}_{25}$	25%
3	$\mathbf{s}_{40}$	100%	23	$\mathbf{s}_{47}$	58%	43	$\mathbf{s}_{17}$	23%
4	$\mathbf{s}_{43}$	100%	24	$\mathbf{s}_{23}$	49%	44	$\mathbf{s}_{42}$	20%
5	$\mathbf{s}_{34}$	97%	25	$\mathbf{s}_{13}$	49%	45	$\mathbf{s}_{49}$	20%
6	$\mathbf{s}_{37}$	97%	26	$\mathbf{s}_{16}$	46%	46	$\mathbf{s}_{11}$	16%
7	$\mathbf{s}_{51}$	96%	27	$\mathbf{s}_6$	44%	47	$\mathbf{s}_{35}$	15%
8	$\mathbf{s}_{15}$	87%	28	$\mathbf{s}_{27}$	44%	48	$\mathbf{s}_{36}$	14%
9	$\mathbf{s}_{33}$	85%	29	$\mathbf{s}_7$	40%	49	$\mathbf{s}_{46}$	11%
10	$\mathbf{s}_{59}$	82%	30	$\mathbf{s}_8$	39%	50	$\mathbf{s}_{10}$	10%
11	$\mathbf{s}_{44}$	82%	31	$\mathbf{s}_{56}$	38%	51	$\mathbf{s}_{22}$	9%
12	$\mathbf{s}_2$	77%	32	$\mathbf{s}_{21}$	36%	52	$\mathbf{s}_{14}$	9%
13	$\mathbf{s}_{41}$	73%	33	$\mathbf{s}_{54}$	34%	53	$\mathbf{s}_{19}$	8%
14	$\mathbf{s}_{20}$	73%	34	$\mathbf{s}_9$	34%	54	$\mathbf{s}_{50}$	7%
15	$\mathbf{s}_1$	70%	35	$\mathbf{s}_{57}$	32%	55	$\mathbf{s}_{24}$	6%
16	$\mathbf{s}_4$	68%	36	$\mathbf{s}_{30}$	31%	56	$\mathbf{s}_{53}$	5%
17	$\mathbf{s}_{38}$	67%	37	$\mathbf{s}_{55}$	31%	57	$\mathbf{s}_{28}$	0%
18	$\mathbf{s}_{31}$	65%	38	$\mathbf{s}_{52}$	29%	58	$\mathbf{s}_{29}$	0%
19	$\mathbf{s}_{18}$	64%	39	$\mathbf{s}_{58}$	29%	59	$\mathbf{s}_{45}$	0%
20	$\mathbf{s}_{39}$	63%	40	$\mathbf{s}_5$	28%	60	$\mathbf{s}_{60}$	0%
						61	$\mathbf{s}_{61}$	0%

The decision tree which consists of three types of nodes (i.e., root nodes, decision nodes, and terminal (or leaf) nodes) and branches also shows similar results. As we can expect from variable importance, the most important variable is on the root node which is located on the top of the decision tree.

4.3 | Cost benefit analysis

Based on the experiment, it suggests RD-LNN is able to detect 5 failures 2 minutes earlier among 25 paper breaks. In this section, we will analyze how much this proposed algorithm could make a contribution to the industries even though the performance does not look high enough to detect every failure before it occurs. Table 6 shows that even a small number of failure reduction improved by this algorithm can save significant costs for the industries every year. The gain is calculated based on the recall 20 percent, and the loss caused by the false alarm is estimated to find the total cost that we can save throughout a year. Ranjan et al.[2] implies that it will cost more than 10,000 dollars for a break. We assumed that failure would occur 124 times for one month based on our dataset. Since the classification algorithm can detect 20 percent of failure, almost 3 million dollars can be saved per year by preventing 297 possible failures. However, we also need to consider the other side, a negative effect caused by a false alarm which gives the warning even though the machine is in the normal state. We assumed that this false alarm would cost 100 dollars because people might stop working and need to check the machine status to find out the problem. Based on the fact that data is captured by every 2 minutes, 1,488, the number of failures occurred every year, is subtracted from the total number of failures. The total loss caused by false alarm would be more than 1 million dollars due to the FPR which is 2 percent. If both positive and negative factors are considered together to find the total cost, we can conclude that the algorithm we propose here can save more than 1 million dollars in total for a year.

5 | CONCLUSION

It is crucial to detect the failure earlier to save cost and labor in a paper manufacturing facility. However, it is challenging to detect machine failure in advance due to the fact that data is comprised of multi-stream time series, and failures which

TABLE 6 Cost benefit Analysis based on RD-LNN

Item	Gain (by Recall)	Loss (by FPR)	Remark
Cost / Occurrence	\$10,000 / True Positive	\$100 / False Positive	Assumed based on the relevant articles
Number of Occurrence	124 x 12 month = 1,488	(2 min x 30 x 24 hr x 365 day) - 1,488 = 524,112	1 year
Occurrence Rate	Recall = 20%	FPR = 2%	Test result
Cost / year	\$2,976,000	- \$1,108,923	
Total cost	\$1,867,077		

rarely occur during operation without any clear symptom. In this paper, two types methods called CL-LNN, RD-LNN are proposed based on the nearest neighbor to extract proper features for failure early detection. The data is preprocessed with several different steps: standardization, moving class label, second derivative, and time window processing. CL-LNN measures Euclidean distance to extract the class label of the nearest neighbor which will be fed into the decision treee classifier for the failure classification. Another algorithm called RD-LNN extracts relative distance generating numerical values which are suitable to be trained with SVM. Experiments are implemented on the dataset provided by the IISE 2019 data competition to show the competitiveness of our proposed methods. Dataset is preprocessed and proposed algorithms are implemented with other machine learning techniques. Through the experiment, it finds that RD-LNN is able to extract features effectively to detect the abnormal condition in the multi-stream time series dataset which would make a considerable contribution to industries by saving cost.

Even though experimental results show the good performance, $F-measure$ 0.15, further research to overcome the rare event situation is still necessary, since improving performance is limited with insufficient labeled data which most of machine learning algorithms normally suffer from. More efforts should be made to overcome the lack of failure data which we normally encounter when collecting the data in industries.

References

1. Bajpai P. Basic overview of pulp and paper manufacturing process. In: Springer. 2015 (pp. 11–39).
2. Ranjan C, Mustonen M, Paynabar K, Pourak K. Dataset: Rare Event Classification in Multivariate Time Series. *arXiv preprint arXiv:1809.10717* 2018.
3. Montgomery DC. *Introduction to statistical quality control*. John Wiley & Sons . 2012.
4. Karim F, Majumdar S, Darabi H, Harford S. Multivariate lstm-fcns for time series classification. *Neural Networks* 2019; 116: 237–245.
5. Wang L, Wang Z, Liu S. An effective multivariate time series classification approach using echo state network and adaptive differential evolution algorithm. *Expert Systems with Applications* 2016; 43: 237–249.
6. Christ M, Kempa-Liehr AW, Feindt M. Distributed and parallel time series feature extraction for industrial big data applications. *arXiv preprint arXiv:1610.07717* 2016.
7. Sykacek P, Roberts SJ. Bayesian time series classification. In: ; 2002: 937–944.
8. Esmael B, Arnaout A, Fruhwirth RK, Thonhauser G. Improving time series classification using Hidden Markov Models. In: IEEE. ; 2012: 502–507.
9. Jović A, Brkić K, Bogunović N. Decision tree ensembles in biomedical time-series classification. In: Springer. ; 2012: 408–417.
10. Eads DR, Hill D, Davis S, et al. Genetic algorithms and support vector machines for time series classification. In: . 4787. International Society for Optics and Photonics. ; 2002: 74–85.

11. Cui Z, Chen W, Chen Y. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995* 2016.

12. Baesens B, Van Gestel T, Viaene S, Stepanova M, Suykens J, Vanthienen J. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the operational research society* 2003; 54(6): 627–635. https://doi.org/10.1057/palgrave.jors.2601545.

13. Lines J, Taylor S, Bagnall A. Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. In: IEEE. ; 2016: 1041–1046.

14. Tan CW, Petitjean F, Webb GI. FastEE: Fast Ensembles of Elastic Distances for time series classification. *Data Mining and Knowledge Discovery* 2019: 1–42. https://doi.org/10.1007/s10618-019-00663-x.

15. Orsenigo C, Vercellis C. Combining discrete SVM and fixed cardinality warping distances for multivariate time series classification. *Pattern Recognition* 2010; 43(11): 3787–3794.

16. Weng X, Shen J. Classification of multivariate time series using two-dimensional singular value decomposition. *Knowledge-Based Systems* 2008; 21(7): 535–539.

17. Zhang C, Yan H, Lee S, Shi J. Multiple profiles sensor-based monitoring and anomaly detection. *Journal of Quality Technology* 2018; 50(4): 344–362.

18. Rodríguez JJ, Alonso CJ. Support vector machines of interval-based features for time series classification. In: Springer. ; 2004: 244–257.

19. Kadous MW, Sammut C. Classification of multivariate time series and structured data using constructive induction. *Machine learning* 2005; 58(2-3): 179–216.

20. Li C, Khan L, Prabhakaran B. Feature selection for classification of variable length multiattribute motions. In: Springer. 2007 (pp. 116–137).

21. Kim J, Huang Q, Shi J, Chang TS. Online multichannel forging tonnage monitoring and fault pattern discrimination using principal curve. 2006. DOI: 10.1115/1.2193552.

22. Chang SI, Yadama S. Statistical process control for monitoring non-linear profiles using wavelet filtering and B-spline approximation. *International Journal of Production Research* 2010; 48(4): 1049–1068. https://doi.org/10.1080/00207540802454799.

23. Paynabar K, Jin J, Pacella M. Analysis of multichannel nonlinear profiles using uncorrelated multilinear principal component analysis with applications in fault detection and diagnosis. *IIE Transactions* 2013; 45(11): 1235–1247.

24. Grasso M, Colosimo BM, Pacella M. Profile monitoring via sensor fusion: the use of PCA methods for multi-channel data. *International Journal of Production Research* 2014; 52(20): 6110–6135. https://doi.org/10.1080/00207543.2014.916431.

25. Zheng Y, Liu Q, Chen E, Ge Y, Zhao JL. Time series classification using multi-channels deep convolutional neural networks. In: Springer. ; 2014: 298–310.

26. Li X, Ding Q, Sun JQ. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety* 2018; 172: 1–11.

27. Baradwaj BK, Pal S. Mining educational data to analyze students' performance. *arXiv preprint arXiv:1201.3417* 2012.

28. Srikanthan S, Kumar A, Gupta R. Implementing the dynamic time warping algorithm in multithreaded environments for real time and unsupervised pattern discovery. In: IEEE. ; 2011: 394–398.

29. Sakoe H. Dynamic-programming approach to continuous speech recognition. In: ; 1971.

30. Górecki T, Łuczak M. Non-isometric transforms in time series classification using DTW. *Knowledge-based systems* 2014; 61: 98–108.

31. Dau HA, Silva DF, Petitjean F, Forestier G, Bagnall A, Keogh E. Judicious setting of Dynamic Time Warping's window width allows more accurate classification of time series. In: IEEE. ; 2017: 917–922.

How to cite this article: Lee W., Seo K. (2020), Early failure detection of paper manufacturing machinery using nearest neighbor based feature extraction, *Qual Reliab Engng Int.*, *20XX;00:X–X*.