Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Anomaly detection of vectorized time series on aircraft battery data

Moting Su[a], Wenjie Zhao[b], Ye Zhu[c,*], Donglan Zha[a], Yushu Zhang[b], Peng Xu[d,*]

[a] College of Economics and Management, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[b] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[c] Centre for Cyber Resilience and Trust, Deakin University, Geelong, 3125, Australia
[d] School of Energy and Environment, Southeast University, Nanjing 210096, China

ARTICLE INFO

Keywords:
Pattern recognition
Vectorization
Time series
Aircraft battery
Anomaly detection

ABSTRACT

The power supply system, as an indispensable electronic hardware module in most vehicles, needs the highest level of security and reliability to ensure the normal operation of the vehicle. Efficiently identifying any faulty battery at the earliest stage would prevent potential safety hazards. This paper aims to detect anomalous batteries using a time series analysis of their internal resistance. To identify the most meaningful patterns and extract their features, we propose a method named Pattern-based Vectorization for Time series (PVT). The PVT first encodes the local sequential shapes as unique symbols by sliding window, then maps each time series into a sequence of representative symbols, and finally generates the symbolic feature matrix for the whole time-series data via a TF-IDF statistical method. The effectiveness of PVT has been systematically evaluated with 7 classifiers on a large real civil aviation battery dataset collected from an uninterruptible power system. Our results show that PVT can significantly improve the performance of existing classifiers in detecting abnormal batteries. In particular, the combination with the RUBT classification model achieves the highest performance due to the random undersampling and boosting techniques that suit imbalanced data in anomaly detection scenarios.

1. Introduction

Nowadays, various types of batteries have been successfully applied to a large number of vehicles, from electric automobiles, forklifts, scooters to aircraft, spaceships and satellites (Cano et al., 2018; Rigo, Seman, Camponogara, Morsch Filho, & Bezerra, 2021; Wang, Qu, Yao, & Yang, 2020). The battery is an indispensable electronic hardware module in the vehicle power supply system, which needs regular charge and discharge to ensure the normal operation of the whole power system. Despite the continuous progress of battery manufacturing and storage technology, the battery cannot avoid failure due to reaching service life or short circuit caused by external factors (Hosseini, Farhadi, & Banisaeid, 2019). In order to assure and enhance the security and reliability of batteries in vehicles, there are a lot of research conducting on battery health monitoring (Burzyński & Kasprzyk, 2021; Chao & Chen, 2011; Lee, Kim, & Lee, 2022; Zhao, Zhang, Zhu, & Xu, 2021).

The life cycle of the battery is generally 3 to 5 years. Degradation is a gradual process, which requires a long period of time to be noticed. Therefore, it is a great challenge to identify the abnormal battery in the early stage. In the operation center, there are a variety of manual battery detection technologies to measure the health status of the battery (Wu, Ji, Liao, & Chang, 2019). For instance, a load test checks whether the battery can provide the specified power when in use. The authors in Křivík (2018) proposed to charge and discharge the battery repeatedly to confirm whether the battery meets the standard or exceeds the cycle life claimed by the manufacturer. However, few research directly deal with automatic identification of abnormal batteries based on battery characteristics. The traditional statistical methods usually set a threshold on some feature values, but they cannot identify the abnormal battery when those features are still within the allowable range. This poses a great threat to system security and reliability.

The uninterrupted power supply system can collect a large number of battery characteristics as time series data. Then how extracting the key features and patterns to discover different kinds of failures is a challenging task (Fu, Leung, Keogh, & Lin, 2006). In addition, there are usually a small number of abnormal batteries (ichi Fukui, Okada, Satoh, & Numao, 2019; Hu et al., 2020), which leads to an extremely unbalanced ratio of positive and negative samples. As a result, most traditional classifiers often perform unsatisfactorily on those datasets. Therefore, it is urgent to propose a more effective discriminative feature extraction method for handling unbalance time

* Corresponding author.
E-mail addresses: sumoting@nuaa.edu.cn (M. Su), zhaowj@nuaa.edu.cn (W. Zhao), ye.zhu@deakin.edu.au (Y. Zhu), zdl@nuaa.edu.cn (D. Zha), yushu@nuaa.edu.cn (Y. Zhang), 103200013@seu.edu.cn (P. Xu).

https://doi.org/10.1016/j.eswa.2023.120219
Received 27 October 2022; Received in revised form 15 April 2023; Accepted 18 April 2023
Available online 28 April 2023

series data, representing unique characteristics of different statuses of batteries. Then existing classifiers can benefit to differentiate abnormal from normal battery status.

This paper aims to detect anomalous batteries using a time series analysis of their internal resistance. To identify the most meaningful patterns and extract their features, we propose a method named Pattern-based Vectorization for Time series (PVT). This feature extraction method can significantly improve the performance of existing classifiers in detecting abnormal batteries based on their internal resistance of time series data.

The main contributions of this paper can be summarized as follows:

- Proposing a new time series feature extraction and transformation method PVT. It first encodes the local sequential shapes as unique symbols using a sliding window, and then maps each time series into a sequence of representative symbols, before conducting a TF-IDF statistical method to generate the final symbolic feature matrix for the whole time-series data.
- Evaluating the effectiveness of PVT with 7 classifiers on a large real civil aviation battery dataset collected from an uninterruptible power system. Our results show that PVT can significantly improve the performance of existing classifiers in detecting abnormal batteries.
- Revealing that the combination of PVT and the RUBT classification model achieves the highest anomaly detection performance since the random undersampling and boosting techniques are good at dealing with highly imbalanced data.

The rest of the paper is organized as follows: Section 2 discusses the related work of three types of feature extraction methods in time series, as well as the methods dealing with imbalance problems. Section 3 describes the process of the proposed PVT algorithm in detail. Section 4 conducts the case study in aircraft battery anomaly detection, and further analyzes the experimental evaluation results. The summary and future work are in Section 5.

2. Related works

In the domain of time series anomaly detection, feature extraction is a hot research topic in recent years. Discovering more meaningful features will significantly benefit the downstream learning tasks. Existing feature extraction methods can be mainly grouped into the following three categories:

(i) Shape-based Features. This is the most intuitive method in the time domain. The waveform of the sequence reflects the trend of variables, and the waveform itself is the distinguishing feature of the sequence. Due to the problems of non-alignment, local deformation, and noise interference in the time series, not every point in the series can provide useful information. For example, in the battery time series, the point when the resistance value changes is more valuable than the point that is constant for a period of time. Based on these problems, Ye and Keogh first proposed the concept of Shapelets (Ye & Keogh, 2011), which can represent a class of subsequences to the greatest extent. However, searching for the best Shapelets through enumeration and information gain is an extremely time-consuming process. Therefore, researchers continue to improve efficiency, such as Rakthanmanon and Keogh jointly proposed FS (Fast Shapelets) (Rakthanmanon & Keogh, 2013), Grabocka proposed LS (Learning Shapelets) (Grabocka, Schilling, Wistuba, & Schmidt-Thieme, 2014), and Lines proposed ST (Shapelets Transform) (Lines, Davis, Hills, & Bagnall, 2012). Nonetheless, although Shapelets can represent the key subsequence of a class, it is difficult to determine the number and length of shapelets.

(ii) Time-Dependent Features. In the fields of medical treatment and fault diagnosis, the time dependence of time series is a crucial feature, and the sequence of events is often the key to the occurrence of anomalies. Using recurrent neural networks such as LSTM (Long–Short Term Memory) (Hochreiter & Schmidhuber, 1997) and GRU (Gate Recurrent Unit) (Dey & Salem, 2017) is an idea to obtain the time-dependent features. Lin and Runger proposed the GCRNN (Group-constrained Convolutional Recurrent Neural Network) (Lin & Runger, 2017) to connect CNN and GRU in series. The time series is first input to CNN to learn pattern features, and then these features are sent to the RNN module for time-dependent feature modeling. In addition, Hu proposed the context hierarchical LSTM model (Hu, Li, Nie, Li, & Shao, 2017), Zhang proposed the attention based time aware LSTM network (Zhang, 2019) and Shi proposed the Convolutional LSTM (Xingjian et al., 2015) to better capture spatiotemporal correlation, which are all variants of LSTM. This neural network model may get satisfactory results, but they are always a black-box model, and the understanding of the extracted features is limited. In addition, they also have a series of disadvantages, such as complex parameters, requiring a large number of training samples and lengthy model training time.

(iii) Sequence Transformation Features. Through the transformation of different forms of time series, a new representation is obtained, retaining the features of the time series while changing the dimensions of time series. This type of feature can be obtained by converting the time series from the original form to another representation through a specific artificial neural network. ESN (Echo State Network) (Jaeger, 2007) is a novel recurrent network. It is an idea to transform the original high-dimensional time series into the corresponding model space. Yang et al. proposed polynomial ESN (Yang, Qiao, Han, & Wang, 2018) to solve the problem of ignoring high-order statistics in the process of ESN conversion from input space to feature space. Gong et al. proposed a multi-objective model measurement (Gong, Chen, Yuan, & Yao, 2018) using ESN to learn and generate a model for each time series, which improves the expression and segmentation ability of multi-objective optimization. Han and Xu proposed Laplacian ESN model (Han & Xu, 2017), which outputs time series based on low dimensional manifold features through the combination of Laplacian and ESN. These network models also have several problems mentioned in the part of time-dependent features.

Furthermore, there is another crucial challenge in time series anomaly detection, i.e., the imbalanced binary classification problem. Most traditional classifiers tend to emphasize the majority of normal samples while neglecting the minority anomalous samples, which causes the learning bias issue in the anomaly detection task (Liang et al., 2021). To overcome this bias, researchers have proposed different solutions to reduce the class imbalance problem (Japkowicz et al., 2000). Weiss (2004) made research on class imbalance problem and common technologies for how to reduce the adverse effect caused by problems, as well as presented common methods to solve this imbalance classification, including sampling and boosting.

There are two widely used sampling methods. One is oversampling technology, which adjusts the distribution of classes by adding samples to the minority class. Another technique called undersampling is to delete the samples of the majority class to improve the proportion of the minority class. Oversampling and undersampling have different advantages and disadvantages. The major disadvantage of undersampling is that the information of deleted samples will be lost (Batista, Prati, & Monard, 2004). On the contrary, its advantage is that it can reduce a lot of model training time, since the number of samples to be trained is greatly reduced. On the other hand, for oversampling, it can effectively avoid the loss of information by adding data instead of deleting data. However, oversampling will easily result in overfitting (Drummond, Holte, et al., 2003). In addition, due to the number of training samples increasing, the training time of the model increases accordingly.

The simplest undersampling method is random undersampling which randomly deletes the samples of the majority class until the desired proportion is reached. Although there is no generally accepted optimal proportion, a balanced uniform distribution (1:1) is usually considered optimal (Khoshgoftaar, Seiffert, Van Hulse, Napolitano, & Folleco, 2007; Weiss & Provost, 2003). Along with random sampling technology, various more intelligent sampling algorithms (Chawla, Bowyer,

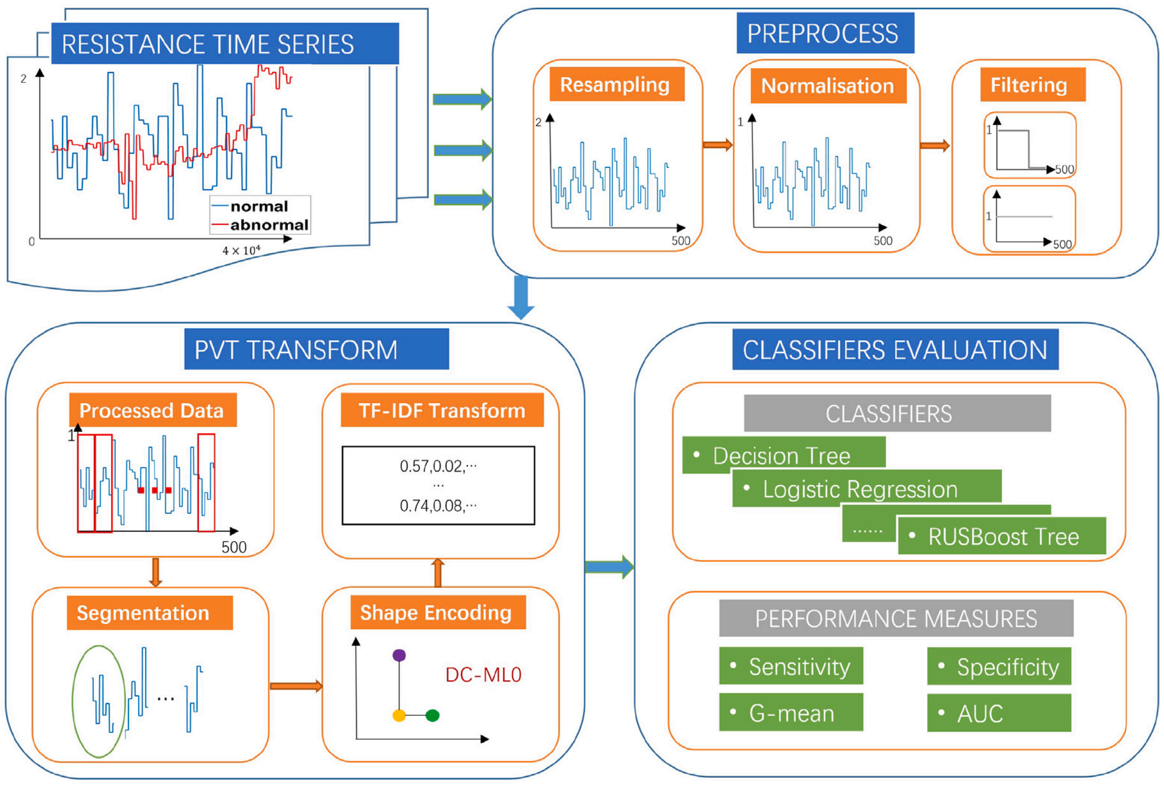


Fig. 1. The flow chart of the proposed method.

Hall, & Kegelmeyer, 2002; Lunardon, Menardi, & Torelli, 2014) are proposed. Barandela, Valdovinos, Sánchez, and Ferri (2004), Han, Wang, and Mao (2005) and Van Hulse, Khoshgoftaar, and Napolitano (2007) evaluated several intelligent sampling techniques, including intelligent and random methods, and found that random undersampling and SMOTE are effective data sampling methods.

Rather than sampling, boosting technology can also be used to enhance the classification performance of the base classifier (Seiffert, Khoshgoftaar, Van Hulse, & Napolitano, 2008). Many variants of AdaBoost have been presented to deal with unbalanced data (Guo & Viktor, 2004; Joshi, Kumar, & Agarwal, 2001). Among them, Chawla proposed SMOTEBoost (Chawla, Lazarevic, Hall, & Bowyer, 2003), which has produced promising results in unbalanced data. However, the defects of the oversampling technology in SMOTEBoost (e.g., high time complexity and troublesome parameter setting) have deteriorated when meeting the boosting, thus it is often not suitable for large datasets.

In this paper, we propose a new feature extraction and transformation method PVT that encodes each series based on local sequential shapes using different symbols and then use a statistical method to generate the distribution of symbols as the representative features. It is different from Shapelet-based method, since there is no search for representative subsequences. Furthermore, unlike most neural network-based methods, the feature extraction process of PVT is totally unsupervised and it only has one parameter, i.e., the sliding window size. Thus, PVT is more efficient and easier to be implemented than existing feature extraction methods.

3. Methodology

Fig. 1 presents the flow chart of our proposed method for anomaly detection on battery data. We first preprocess the original resistance time series and then transform the data using our proposed PVT algorithm. The point pattern and fragment pattern bases will be extracted before symbolization. Finally, the symbols are transformed into digital feature forms with TF-IDF method. After obtaining the features, we can use existing classifiers to learn the data distribution and conduct anomaly detection.

Before formally introducing PVT algorithm, we first define basic notations of time series.

Definition 1. Time series T is a real valued digital sequence arranged in time order, which is expressed as $T = (t_1, x_1), (t_2, x_2), \dots, (t_n, x_n)$, where n represents the length of T.

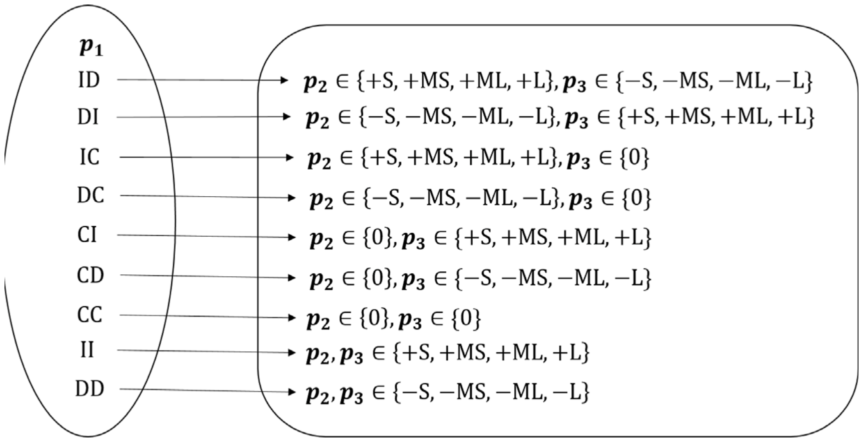
Definition 2. A non-overlapping subsequence S_i of time series T is a continuous subset generated by sliding window W with size m on time series T in step m. There is no overlap between subsequences, starting from position $(i-1)\times m+1$ and ending at position $i\times m$, where $i = 1 : \lceil \frac{n}{m} \rceil$.

If $\frac{n}{m}$ is not an integer, we can extend the time series until it is can be perfectly divided.

By using a sliding window on a time series T, we can easily obtain the set of non-overlapping subsequences $S = \{S_1, S_2, \dots, S_{\lceil \frac{n}{m} \rceil}\}$, where the length of each subsequence is m. Then we can select one of the subsequences S_i to create a point pattern (PP) and fragment pattern base (FPB) as follows.

3.1. Point pattern

Definition 3. Let (t_i, x_i) be a point in the non-overlapping subsequence S_i, (t_{i-1}, x_{i-1}) and (t_{i+1}, x_{i+1}) are the left and right adjacent points of (t_i, x_i) respectively. We use PP_i to represent the symbolic representation of (t_i, x_i). PP_i is connected by three parts p_1, p_2 and p_3. p_1 records the positive or negative properties of the difference between (t_i, x_i) and adjacent points, reflecting the fluctuation trend of these three continuous points. p_2 and p_3 represent the range of the difference between (t_i, x_i) and adjacent points, reflecting the fluctuation amplitude of these three continuous points.



p_1
ID
DI
IC
DC
CI
CD
CC
II
DD
$p_2 \in \{+S, +MS, +ML, +L\}, p_3 \in \{-S, -MS, -ML, -L\}$
$p_2 \in \{-S, -MS, -ML, -L\}, p_3 \in \{+S, +MS, +ML, +L\}$
$p_2 \in \{+S, +MS, +ML, +L\}, p_3 \in \{0\}$
$p_2 \in \{-S, -MS, -ML, -L\}, p_3 \in \{0\}$
$p_2 \in \{0\}, p_3 \in \{+S, +MS, +ML, +L\}$
$p_2 \in \{0\}, p_3 \in \{-S, -MS, -ML, -L\}$
$p_2 \in \{0\}, p_3 \in \{0\}$
$p_2, p_3 \in \{+S, +MS, +ML, +L\}$
$p_2, p_3 \in \{-S, -MS, -ML, -L\}$

Fig. 2. A total of 81 different combinations for p_1, p_2, and p_3.

Comparing with its previous point, each point can be described as one of the three trends, i.e., decrease, constant, or increase. When using the symbol to describe two adjacent points for p_1, we have 9 combinations, as shown in Eq. (1), including ID (Increase Decrease), DI (Decrease Increase), IC (Increase Constant), DC (Decrease Constant), CI (Constant Increase), CD (Constant Decrease), CC (Constant Constant), II (Increase Increase), and DD(Decrease Decrease). For example, when d_1 is greater than 0 and d_2 is less than 0, it means that the three consecutive points (t_{i-1}, x_{i-1}), (t_i, x_i) and (t_{i+1}, x_{i+1}) present a trend of increasing first and then decreasing, thus we mark the p_1 of (t_i, x_i) as ID. When d_1 and d_2 equal to 0, their values are constant. In this case, we mark the p_1 as CC.

Once obtaining the gap values between two adjacent points, we can categorize them into different levels and assign them to p_2 and p_3, based on the value range. It is possible to define less or more symbols for categorization. Eq. (2) elaborates the 9 symbols used as default in our paper, including 0, $+S$ (+ Small), $+MS$ (+ Middle Small), $+ML$ (+ Middle Large), $+L$ (+ Large), $-S$ (− Small), $-MS$ (− Middle Small), $-ML$ (- Middle Large), and $-L$ (- Large).

When normalizing all data points to $[0, 1]$, then the gap values between any two adjacent points are in the range of $[-1, 1]$. Therefore, we divide the value into 9 different intervals and use different symbols to represent p_2 and p_3.

Fig. 2 shows 81 different combinations of p_1, p_2 and p_3. Points in S_i can be connected into a kind of PP according to its corresponding p_1, p_2 and p_3. A special example is shown in Fig. 3, which vividly covers 9 different combinations of p_1, p_2 and p_3. When each point is symbolized, there will be redundancy of information, such as the white point between the red point and the orange point, whose p_2 is equal to p_3 of the red point and p_3 is equal to p_2 of the orange point. Therefore, the points in the sequence are symbolized by every other point. For the red point, we can calculate that $p_2 = 0.5 \in (0.25, 0.5] > 0$, $p_3 = -0.25 \in (0, -0.25] < 0$, and then we can infer that the PP of the red point is $ID + MS - S$ according to Eqs. (1) and (2). Similarly, we can get the PP of other color points.

$$p_1 = F_1(d_1, d_2) = \begin{cases} ID, & \text{if} \quad d_1 > 0 \quad \text{and} \quad d_2 < 0 \\ DI, & \text{if} \quad d_1 < 0 \quad \text{and} \quad d_2 > 0 \\ IC, & \text{if} \quad d_1 > 0 \quad \text{and} \quad d_2 = 0 \\ DC, & \text{if} \quad d_1 < 0 \quad \text{and} \quad d_2 = 0 \\ CI, & \text{if} \quad d_1 = 0 \quad \text{and} \quad d_2 > 0 \\ CD, & \text{if} \quad d_1 = 0 \quad \text{and} \quad d_2 < 0 \\ CC, & \text{if} \quad d_1 = 0 \quad \text{and} \quad d_2 = 0 \\ II, & \text{if} \quad d_1 > 0 \quad \text{and} \quad d_2 > 0 \\ DD, & \text{if} \quad d_1 < 0 \quad \text{and} \quad d_2 < 0 \end{cases}, \tag{1}$$

where $d_1 = x_i - x_{i-1}$; $d_2 = x_{i+1} - x_i$.

$$p_i = F_2(d_i) = \begin{cases} 0, & \text{if} \quad d_i = 0 \\ +S, & \text{if} \quad d_i \in (0, 0.25] \\ +MS, & \text{if} \quad d_i \in (0.25, 0.5] \\ +ML, & \text{if} \quad d_i \in (0.5, 0.75] \\ +L, & \text{if} \quad d_i \in (0.75, 1] \\ -S, & \text{if} \quad d_i \in (0, -0.25] \\ -MS, & \text{if} \quad d_i \in (-0.25, -0.5] \\ -ML, & \text{if} \quad d_i \in (-0.5, 0.75] \\ -L, & \text{if} \quad d_i \in (-0.75, -1] \end{cases}, \tag{2}$$

where $i = 1, 2; d_1 = x_i - x_{i-1}$; $d_2 = x_{i+1} - x_i$.

3.2. Fragment pattern base

Definition 4. Let S_i be an element in the non-overlapping subsequence set S with length m in time series T. According to $Definition$ 3 with (1) and (2), we calculate the PP of $\lfloor \frac{m}{2} \rfloor$ points in S_i. As shown in (3), we concatenate these PP to obtain the fragment pattern base (FPB_i) of S_i.

$$FPB_i = PP_1 + PP_2 + \cdots + PP_{\lfloor \frac{m}{2} \rfloor}, \tag{3}$$

where + means the connection of two PP.

For example, let us refer to Fig. 3 again, when the sliding window size $m = 5$, we take the first five points as S_1, and calculate the PP of red and orange points, before concatenating their PP according to Eq. (3) to obtain that FPB_1 of S_1 is $ID + MS - SDI - ML + S$. Similarly, we can calculate the PP of yellow point and green point, blue point and purple point, black point and gray point, and then concatenate their PP in pairs to get FPB_2, FPB_3, and FPB_4 of S_2, S_3 and S_4, respectively.

3.3. ID-IDF vectorization

Once converting continuous time series into a discrete string representation, We can encode a time series based on those representations. The third step of our PVT algorithm is to calculate the term frequency (TF) and inverse document frequency (IDF) of these FPB by using the TF-IDF statistical method, which is very popular used in text processing (Beel, Gipp, Langer, & Breitinger, 2016). The character features extracted from the sequence are transformed into digital features, so that we can use the classifier to train the model. At this time, the digital feature is not a sampling point or a sampling value of the original data, but the importance of a FPB in a time series, in other words, it determines how much the sequence plays a key role in the normal or abnormal class. The TF-IDF is defined as follows (Salton & Buckley, 1988).

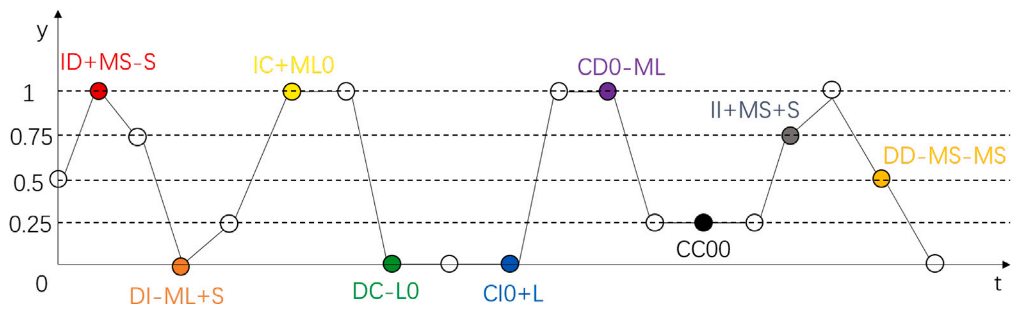


Fig. 3. Types of point pattern.

Definition 5. Term Frequency (TF) describes the frequency of a term in a document. In section III-A2, we calculate FPB set of a time series T. In this paper, we use $TF_{FPB,T}$ to represent the frequency of a FPB in the FPB set of T. The larger TF is, the more FPB appears. The calculation formula is shown as:

$$TF_{FPB,T} = \frac{N_{FPB,T}}{AN_{FPB,T}}, \tag{4}$$

where $N_{FPB,T}$ refers to the number of this FPB appears in the FPB set of T, and $AN_{FPB,T}$ means the number of FPB in the FPB set of T.

We cannot judge the importance of a term only from the high frequency, since some recurring terms in a document set are not of interest to us. For instance, in the two sentences, "This cat is cute." and "This dog is lazy.", we are not interested in the repeated copula (is) and pronoun (this). On the contrary, the subject (cat, dog) and adjective (cute, lazy) need our more attention. The same is true for FPB. If an FPB appears in all sequences in the time series set, that is, in both normal and abnormal data, it is not beneficial to anomaly detection. In contrast, even if an FPB does not appear most frequently in the whole dataset, but it appears frequently in normal or abnormal data, it also plays a representative significance for the whole sequence set. Therefore, we need to introduce inverse document frequency to give these meaningful FPB more weight.

Definition 6. Inverse Document Frequency (IDF) is an index to measure the importance of a term in the document set. In this paper, we use IDF_{FPB} to represent the importance of a FPB to the whole time series set. The larger the IDF, the more crucial the FPB is. The calculation formula is shown in Eq. (5).

$$IDF_{FPB} = \log \frac{N_T}{TSS_{FPB}}, \tag{5}$$

where N_T is the number of time series set, and TSS_{FPB} is the number of sequences with FPB in the time series set.

Finally, after calculating the TF and IDF of FPB, the TF-IDF score is obtained by multiplying them, as shown in Eq. (6). We use $TF-IDF_{FPB,T}$ to indicate the importance of an FPB to time series T. TF-IDF uses the multiplication of TF and IDF to obtain a matrix for FPB vectorization, which represents the new digital characteristics of sequence set, and comprehensively considers the occurrence frequency and importance of FPB. Furthermore, it balances the impact of FPB on the classifier and facilitates the training of classifiers that can recognize anomalies.

$$TF-IDF_{FPB,T} = TF_{FPB,T} \times IDF_{FPB}. \tag{6}$$

The PVT algorithm is shown in Algorithm 1, the inputs are the normalized matrix of time series set T and the sliding window size m. Meanwhile, the output is the TF-IDF matrix M after feature transformation. The algorithm first initializes the input parameter $T-FPB$ of TF-IDF subalgorithm corresponding to time series set T. The 8–12 lines of the innermost loop initialize the PP set, calculate the differences d_1 and d_2, and then obtain the three parts of the PP according to Eqs. (1) and (2). Finally, it connects them to get the PP and adds it to the PP set until the end of the loop. The second layer loop obtains the PP set according to the inner layer loop operation, and then connects each PP to obtain the FPB, which is incorporated into the FPB set. The outermost loop control traverses the whole time series set T to obtain the FPB set $T-FPB$ of all sequences.

The obtained $T-FPB$ will be used as the input of the subalgorithm TF-IDF. The first line of Algorithm 2 obtains all non-repeated FPBs. Lines 4–7 respectively calculate the number of occurrences of an FPB corresponding to a sequence and all sequences in the FPB set, and obtain TF and IDF according to Eqs. (4) and (5). Finally, $TF-IDF$ matrix M is obtained by multiplying TF and IDF. The time complexity of PVT algorithm is $O(k(m+size(term)))$. In order to reduce the complexity, the sliding window size m is usually a small number, which reduces the number of terms at the same time.

Algorithm 1 PVT

```
Input: Time series set $T = \{T_1, T_2, ...T_k\}$,
       Sliding window size $m$
Output: TF-IDF matrix $M$
1:  $T-FPB = \emptyset$;
2:  for a = 1:k do
3:      $FPB = \emptyset$;
4:      $temp = T_i$;
5:      for b=1:$\lceil \frac{n}{m} \rceil$ do
6:          $S = T_i[(i-1)*m+1 : i*m]$
7:          for c=1:2:m do
8:              $PP = \emptyset$;
9:              $d_1 = x_i - x_{i-1}$; $d_2 = x_{i+1} - x_i$;
10:             $p_1 = F_1(d_1, d_2)$; $p_2 = F_2(d_1)$; $p_3 = F_2(d_2)$;
11:             $PP_j = p_1 + p_2 + p_3$;
12:             $PP = PP \cup PP_c$;
13:         end for
14:         $FPB_b = PP_1 + PP_2 + ... + PP_m$;
15:         $FPB = FPB \cup FPB_b$;
16:     end for
17:     $T-FPB = T-FPB \cup FPB$;
18: end for
19: $M = TF-IDF(T-FPB)$;
20: return $M$
```

4. Experiments

In this section, we conduct an empirical evaluation to verify the effectiveness of the proposed PVT algorithm proposed for anomaly detection on real-world aircraft battery dataset.

Algorithm 2 TF-IDF

```
Input: FPB of time series set $T - FPB$
Output: TF-IDF matrix $M$
 1: $term = unique(T - FPB)$;
 2: for $a = 1 : size(T - FPB, 1)$ do
 3:     for $b = 1 : size(term, 1)$ do
 4:         $num_{one} = find(term_b, T - FPB_a)$;
 5:         $num_{all} = find(term_b, T - FPB)$;
 6:         $TF = \frac{num_{one}}{size(T-FPB,2)}$;
 7:         $IDF = log_{10}\frac{size(T-FPB,1)}{num_{all}}$;
 8:     end for
 9: end for
10: $M = TF * IDF$;
11: return $M$
```

4.1. Dataset and preprocessing

We collect the status of resistance data from UPS (Uninterruptible Power System) in the big data center of civil aviation in China. It contains the information of 9416 lead–acid batteries for one month with 180 abnormal labels. Due to the various sampling frequency and inconsistent starting time of each battery, we conduct a preprocessing as follows.

First, we align each time series and unify the length to obtain the same 500 time stamps for each battery. It effectively reduces data redundancy as well. According to the principle of PVT algorithm, when calculating the mapping between difference and symbols in point pattern, it is necessary to control the range of difference to $[-1, 1]$, Thus, we also normalize the data to make the value fall within the range of $[0, 1]$.

4.2. Evaluation criteria

In this paper, we choose four different measurements for classification result evaluation, including $Sensitivity$, $Specificity$, G-mean and AUC (Area under the ROC Curve).

The first evaluation criterion is $Sensitivity$, also known as true positive rate, which measures the proportion of positive samples correctly classified. The second evaluation criterion to weigh with $Sensitivity$ is $Specificity$, also known as true negative rate, which measures the proportion of negative samples correctly classified. Their calculation formulas are shown in (7) and (8).

We compare the predicted labels with the ground true labels and calculate the true positives (TP), false negatives (FN), true negatives (TN) and false positives (FP). A higher score of each measure indicates a better classification performance. The first three measures can be obtained as follows:

$$Sensitivity = \frac{TP}{TP + FN}, \tag{7}$$

$$Specificity = \frac{TN}{TN + FP}. \tag{8}$$

$$\begin{aligned} G - mean &= \sqrt{Sensitivity * Specificity} \\ &= \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}}. \end{aligned} \tag{9}$$

In addition, ROC takes sensitivity as the ordinate and 1-specificity (false positive rate) as the abscissa. AUC is defined as the area below the ROC.

4.3. Experimental setup

All experiments were carried out using MATLAB with the processor, CPU: Inter(R) Core(TM) i7-8700 @ 3.2 GHz, RAM: 16 GB. We selected four traditional classifiers: Decision Tree (DT) (Loh, 2011), Logistic Regression (LR) (Wright, 1995), Support Vector Machine (SVM) (Pisner & Schnyer, 2020) and K-nearest neighbor (KNN) (Peterson, 2009); and three ensemble classifier models: AdaBoost Tree (ABT) (Freund, Schapire, et al., 1996), Bagging Tree (BT) (Breiman, 1996) and RUboost Tree (RUBT) (Seiffert, Khoshgoftaar, Van Hulse, & Napolitano, 2009). We run all classifiers on both the original resistance data and the data transformed by PVT algorithm. The details of each classifier are provided in Appendix.

The following is the description of parameter abbreviations and search range: MSN is the maximum splitting number, SC is the splitting criteria, ADS is the alternative decision splitting, KF is the kernel function, K is the neighbor number, DM is the distance measure, DW is the distance weight, LN is the learner number, and LR is the learning rate. The parameters of the seven classifiers take the built-in settings of MATLAB machine learning toolbox by default, and run on the original resistance data and the data transformed by the PVT algorithm, respectively. In order to test the parameter sensitivity of PVT algorithm, five different sliding window sizes 4, 6, 8, 10 and 12 were set for experiments. Ten-fold-cross-validation was used in all experiments. The data was randomly divided into ten parts, of which 9 parts were used to train the model and the remaining part was used to test the model. The evaluation result of each model is the average over the 10 test parts.

4.4. Results analysis

Table 1 shows the classification performance of seven classifiers on both the original data and the feature data after the PVT transformation. It can be seen from these results that all classifiers have been improved with varying degrees on the PVT features with respect to almost all measurements.

It is interesting to mention that BT, ABT and RUBT are all decision-tree based methods. Both ABT and RUBT use a boosting method, but RUBT includes an unbalanced data sampling strategy. The performance of the RUBT classifier has been greatly improved and the effect is remarkable if running on PVT features. What deserves more attention is the change of sensitivity. Except for the decline of BT classifier, other classifiers have been maintained and enhanced. In particular, the sensitivity of the RUBT classifier has been significantly improved. Since RUBT (Seiffert et al., 2009) integrates boosting learning and unbalanced data sampling technology, it greatly improves the accuracy of unbalanced data classification. In the classification problem with large sample size, it shows better prediction results and faster prediction performance.

Based on the sensitivity and specificity, we can observe that G-mean has a similar improvement when using PVT. The results of most classifiers hover at 0.5, but RUBT classifier achieves a wonderful effect of 0.85. Moreover, from the perspective of AUC, all classifiers have been improved, among which DT, ABT and RUBT classifiers have improved significantly, and the RUBT still maintains the highest value. To sum up, the PVT algorithm has a good improvement effect on the battery time series data. Overall, the combination of the PVT and RUBT classifier can archive the highest classification performance.

In order to test the influence of the PVT algorithm parameter m (sliding window size) on the performance of RUBT classification model, we set $m = 4, 6, 8, 10, 12$ for sensitivity analysis. It can be concluded from Fig. 4 that the four performance measurements have little influence caused by the change of the sliding window size.

5. Discussion and conclusion

In this paper, we have proposed a new feature extraction algorithm, called Pattern-based Vectorization for Time series (PVT). The experimental results on real civil aircraft battery data demonstrate the promising performance of PVT to improve existing classifiers. The proposed PVT method can effectively help to detect abnormal batteries

Table 1
The best performance of the seven classifiers on the original data and PVT transformed data, which are the average of ten runs. The best results are given in boldface.

Classifier	Parameter	Sensitivity		Specificity		G-mean		AUC	
		Original	PVT	Original	PVT	Original	PVT	Original	PVT
DT	MSN = 100 SC = Gini ADS = Close	0.175	**0.246**	0.994	**0.996**	0.417	**0.495**	0.600	**0.760**
LR	None	0.281	**0.298**	0.986	**0.988**	0.526	**0.543**	0.650	**0.730**
SVM	KF = Gaussian	**0.246**	**0.246**	**1.000**	**1.000**	**0.496**	**0.496**	0.740	**0.790**
KNN	K = 1 DM = Euclidean DW = Equal	0.281	**0.298**	0.995	**0.996**	0.529	**0.545**	0.640	**0.650**
BT	MSN = 7480 LN = 30 LR = 0.1	**0.228**	0.149	**1.000**	**1.000**	**0.477**	0.386	0.665	**0.678**
ABT	MSN = 20 LN = 30 LR = 0.1	**0.246**	**0.246**	**1.000**	**1.000**	**0.496**	**0.496**	0.710	**0.870**
RUBT	MSN = 20 LN = 30 LR = 0.1	0.516	**0.828**	0.704	**0.876**	0.603	**0.852**	0.658	**0.915**

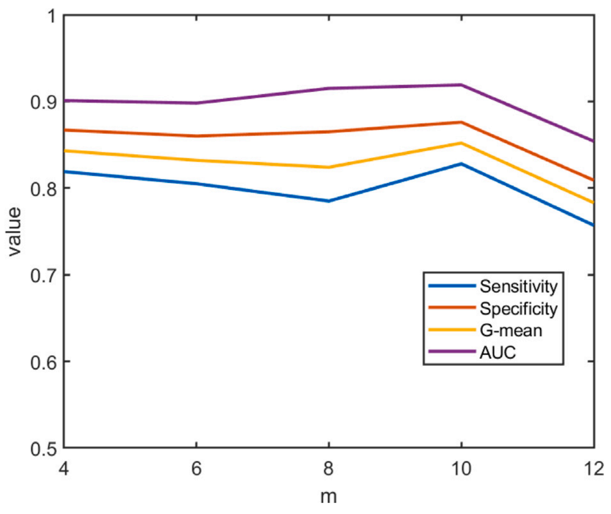


Fig. 4. Sensitivity analysis of PVT using the RUBT classifier with different *m* (sliding window size) setting.

based on the time series of resistance data, and improve the quality and reliability of civil aviation aircraft energy systems.

The practical implication of this paper is to provide an effective way of encoding time series without the expensive search. PVT is also an unsupervised feature extraction method for time series datasets without prior domain knowledge. Thus, it can be extended to different unsupervised learning tasks such as time series clustering. Our evaluation shows that the combination of PVT and the RUBT classification model achieves the highest performance due to the random undersampling and boosting techniques that suit imbalanced data in anomaly detection scenarios. This combination also can be applied in cybersecurity, finance and manufacturing industries to identify abnormal events.

It is worth mentioning that we demonstrated the effectiveness of PVT with only 4 traditional and 3 ensemble classifiers, as they are easy to implement and interpret. Although deep learning has advanced significantly in time series anomaly detection, we did not compare it with any deep learning method as the case study dataset is relatively small. Another limitation of this study is that the PVT algorithm relies on pre-defined 9 symbols for categorization, based on the gap value between two adjacent points. It is possible to design an adaptive categorization method that automatically determines the number of symbols based on the most representative gap values.

In the future, we will investigate the extension of PVT and RUBT to a broader field of applications. In addition, we will investigate how to incorporate PVT with a deep learning-based method to identify anomalies on large and complex time series datasets.

CRediT authorship contribution statement

Moting Su: Conceptualization, Methodology, Software. **Wenjie Zhao:** Data curation, Software, Writing – original draft. **Ye Zhu:** Supervision, Writing – review & editing. **Donglan Zha:** Supervision, Project administration. **Yushu Zhang:** Formal analysis, Validation, Funding acquisition. **Peng Xu:** Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by Basic Research Program of Jiangsu Province (BK20201290) and Natural Science Foundation of Jiangsu Province of China (BK20190374).

Appendix. Evaluation classifiers

A.0.1. Decision tree

Decision Tree (DT) (Loh, 2011) is a simple but widely used classifier. As the name suggests, the decision tree is based on the tree structure, which is equivalent to a series of if-then statements. When judging the class of a sample, first judge which of the left and right subtrees is satisfied from the first node, and then judge downward in the same way until the category of the leaf node is reached. CART (Loh, 2011) decision tree uses the Gini index instead of the information gain ratio as the splitting criterion, which represents the impure of the model. The smaller the value, the lower the impure, and the better the features, which is opposite to the information gain ratio. The calculation formula of the Gini coefficient is shown in Eq. (A.1).

$$\text{Gini(Y)} = \sum_{i=1}^{|Y|}\sum_{i'\neq i} p_i p_{i'} = 1 - \sum_{i=1}^{|Y|} p_i^2, \tag{A.1}$$

where Y is the sample set and p_i is the proportion of class i samples in Y.

A.0.2. Logistic regression

LR (Wright, 1995) is a linear model used to deal with binary classification problems. It linearly combines the attributes of samples, and then outputs the function value through Sigmoid function transformation. The calculation formula for judging the class of a sample is shown in (11). When the value of the linear combination is greater than 0, the function output is 1; conversely, the output is 0.

$$p(T) = \frac{1}{1+e^{-(\theta_0+\theta_1 t_1+\theta_2 t_2+\cdots+\theta_n t_n)}}, \tag{A.2}$$

where θ_i is the parameter of attribute t_i in T.

A.0.3. Support vector machine

SVM (Pisner & Schnyer, 2020) establishes an optimal hyperplane to realize sample classification. In order to solve the problem of linear indivisibility, the function of kernel function (KF) is to refract the low-dimensional samples into the high-dimensional space so that they can be classified correctly. Gaussian KF has good performance for large samples or small samples which can be used in nonlinear situations. At the same time, it has only one parameter, so Gaussian KF is preferred. The parameter of Gaussian KF σ determines the generalization ability of Gaussian kernel. The support vector machine model function using Gaussian KF is shown in (12).

$$f(T) = \sum_{i=1}^{k} \alpha_i Y_i \mathcal{K}(T, T_i) + b, \tag{A.3}$$

where α is the Lagrange multiplier, $\mathcal{K} = exp(-\frac{\|T-T_i\|^2}{2\sigma^2})$ is the Gaussian kernel function, Y_i is the class of sample T_i, and b is the offset.

A.0.4. K-nearest neighbor

KNN (Peterson, 2009) can be used for both supervised learning and unsupervised sample distance calculation. The process is to calculate the distance between the sample and the sample in the dataset based on a specific distance measurement method, select the K samples with the smallest distance, and then predict based on their labels. The result is usually the class that appears the most. In addition, the weighted average can be carried out based on the distance, and the weight increases according to the decrease of the distance. The most commonly used distance measurement method is European distance, and the calculation formula is shown in (13).

$$dist(T, T') = \sqrt{(t_1 - t_1') + (t_2 - t_2') + \cdots + (t_n - t_n')}, \tag{A.4}$$

where T, T' are two time series, t_i is data point, and n is sequence length.

A.0.5. AdaBoost tree

ABT (Freund et al., 1996) is an algorithm that can promote the decision tree as a weak learner to a strong learner of multiple decision tree sets. There is a strong dependency between each weak learner, thus the next learner needs to be generated serially based on the previous learner. The algorithm first trains a base learner from the original training set, and then adjusts the weight of the training samples according to the prediction results of the base learner, so that the samples judged wrong by the previous classifier get greater weight in the next training. After the weight adjustment, the next base learner is trained. In this way, the cycle is executed until the specified number of iterations I is reached, and finally the prediction result P is output. The calculation formula is shown in (14).

$$P(T) = sign(\sum_{i=1}^{I} \beta_i P_i(T)), \tag{A.5}$$

where β_i and $P_i(T)$ are the weight of the ith base classifier and the prediction result of the ith base classifier for sample T respectively.

A.0.6. Bagging tree

BT (Breiman, 1996) is a parallel integration algorithm that takes the decision tree as a weak learner. Different from boosting, each base learner can be generated at the same time without relying on the previous learner. The basic process of the bagging tree is to select I sets containing k' training samples through bootstrap sampling (Efron & Tibshirani, 1994), then train a learner for each sampling set, and finally combine the output of these learners. The simple voting method is usually used, that is, the winner who votes more is the winner. The final prediction result is shown in (15).

$$P(T) = \underset{Y\in\{-1,1\}}{\arg\max} \sum_{i=1}^{I} \mathcal{I}(P_i(T) = Y), \tag{A.6}$$

where $\mathcal{I}$ is the indicator function.

A.0.7. RUSBoost tree

RUSBoost (Seiffert et al., 2009) tree (RUBT) is a method of integrating boosting learning and unbalanced data sampling technology. Based on AdaBoost theory, the model adds random undersampling technology, which greatly improves the accuracy of unbalanced data classification. In the classification problem with a large sample size, it shows better prediction results and faster prediction performance.

Random undersampling is one of the techniques adopted in RUBT classification model. When the original dataset is pretty large, it attempts to adjust the class distribution of the training set to mitigate the issue of data unbalance. In the anomaly detection task, random undersampling will delete the majority of negative samples, so that the ratio of positive and negative samples tends to a preset ratio after deletion. On the one hand, the benefit of random undersampling is that it reduces the training time required to build the model. On the other hand, the major disadvantage of random undersampling is that it will lose partial information contained in the deleted data. However, this shortcoming is well overcome through the combination of boosting technology. Although certain information may be lost during one iteration of boosting, it will be included during other iterations.

Different from many sampling techniques that are specially designed to address the issue of data unbalance, in fact, the original intention of boosting is to enhance the classification performance of base classifiers. The reason why this technology can solve the problem of unbalanced data in the field of anomaly detection is that the minority positive samples are likely to be misclassified at the beginning of the iteration, thus higher weights will be given in the subsequent iteration. After reweighting, the modified sample weight can not only be directly transmitted to the next round of base classifier for decision reference, but also be used as the basis of each round of random undersampling.

The RUBT Algorithm is shown in Algorithm 3. To combine with PVT, the input of the algorithm is the TF-IDF matrix M obtained in the feature extraction, where M_i means the TF-IDF feature vector of the time series. T_i, Y_i denotes the corresponding class label, i.e., −1 represents the negative sample as the normal sample, and +1 represents the positive sample as the abnormal sample. I indicates the number of iterations, which can also be understood as the number of base classifiers. The output of the algorithm is the sum of the results of the base classifier multiplied by the corresponding weights. If the sum is greater than 0, it is judged as abnormal, and if it is less than 0, it is judged as normal.

The first line of the algorithm initializes the weight to $\frac{1}{k}$, where k is the total number of training samples. The algorithm iteratively trains the base classifier in lines 2–12. Firstly, the random undersampling and the weight of the sample (the lower the weight, the easier it is to be deleted) are used to adjust the proportion of the sample to obtain the subsample D_j. Then the decision tree is selected as the base classifier to train the classification model B_j. The prediction result P_j obtained by B_j model is used to calculate the error L_j. When L_j is less than 0.5, the parameter β_j and sample weight δ_{j+1} are updated. At the end of the iteration, the decision function $P(T)$ is obtained.

Algorithm 3 RUBT

```
Input: TF-IDF $M = \{(M_1, Y_1), (M_2, Y_2), ..., (M_k, Y_k)\}$,
       $Y_i \in \{-1, 1\}$, Iteration times $I$
Output: $P(T) \in \{-1, 1\}$
 1: $\delta_1(i) = \frac{1}{k}, i = 1, ..., k$;
 2: for $j = 1 : I$ do
 3:     $D_j = Undersampling(M, \delta_j)$;
 4:     $B_j = DecisionTree(D_j, \delta_j)$;
 5:     $P_j = B_j(M)$;
 6:     $L_j = \sum_{i=1}^{k} \delta_j(i) Error(P_j(M_i) \neq Y_i)$;
 7:     if $L_j > 0.5$ then
 8:         break;
 9:     end if
10:     $\beta_j = \frac{1}{2} ln(\frac{1-L_j}{L_j})$;
11:     $\delta_{j+1}(i) = \frac{\delta_j(i)}{Z_j} e^{-\beta_j Y_i P_j(M_i)}$, $Z_j = \sum_{i=1}^{k} \delta_{j+1}(i)$;
12: end for
13: return $P(M) = sign(\sum_{j=1}^{I} \beta_j P_j(M))$
```

References

Barandela, R., Valdovinos, R. M., Sánchez, J. S., & Ferri, F. J. (2004). The imbalanced training sample problem: Under or over sampling? In *Proc. joint IAPR int. workshops* (pp. 806–814).

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, *6*(1), 20–29.

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Paper recommender systems: a literature survey. *International Journal on Digital Libraries*, *17*(4), 305–338.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Burzyński, D., & Kasprzyk, L. (2021). A novel method for the modeling of the state of health of lithium-ion cells using machine learning for practical applications. *Knowledge-Based on Systems*, *219*, Article 106900.

Cano, Z. P., Banham, D., Ye, S., Hintennach, A., Lu, J., Fowler, M., et al. (2018). Batteries and fuel cells for emerging electric vehicle markets. *Nature Energy*, *3*(4), 279–289.

Chao, K.-H., & Chen, J.-W. (2011). State-of-health estimator based-on extension theory with a learning mechanism for lead-acid batteries. *Expert Systems with Applications*, *38*(12), 15183–15193.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of the Artifical Intelligence Research*, *16*, 321–357.

Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003). SMOTEBoost: Improving prediction of the minority class in boosting. In *Proc. euro. conf. princ. data mining knowl. disc.* (pp. 107–119).

Dey, R., & Salem, F. M. (2017). Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)* (pp. 1597–1600). IEEE.

Drummond, C., Holte, R. C., et al. (2003). C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop learn. imbalanced data, Vol. 11* (pp. 1–8). Citeseer.

Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC Press.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *Proc. int. conf. machine learn., Vol. 96* (pp. 148–156).

Fu, A. W.-C., Leung, O. T.-W., Keogh, E., & Lin, J. (2006). Finding time series discords based on haar transform. In *Proc. int. conf. advanced data mining app.* (pp. 31–41).

ichi Fukui, K., Okada, Y., Satoh, K., & Numao, M. (2019). Cluster sequence mining from event sequence data and its application to damage correlation analysis. *Knowledge-Based on Systems*, *179*, 136–144.

Gong, Z., Chen, H., Yuan, B., & Yao, X. (2018). Multiobjective learning in the model space for time series classification. *IEEE Transactions on Cybernetics*, *49*(3), 918–932.

Grabocka, J., Schilling, N., Wistuba, M., & Schmidt-Thieme, L. (2014). Learning time-series shapelets. In *Proc. ACM int. conf. knowl. disc. data mining* (pp. 392–401).

Guo, H., & Viktor, H. L. (2004). Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM Sigkdd Explorations Newsletter*, *6*(1), 30–39.

Han, H., Wang, W.-Y., & Mao, B.-H. (2005). Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *Proc. int. conf. intell. comput.* (pp. 878–887).

Han, M., & Xu, M. (2017). Laplacian echo state network for multivariate time series prediction. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(1), 238–244.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hosseini, S., Farhadi, K., & Banisaeid, S. (2019). Improving particle size of BaSO4 with a unique glycerol base method and its impact on the negative active material of the lead-acid battery. *Journal of Energy Storage*, *21*, 139–148.

Hu, L., Li, J., Nie, L., Li, X.-L., & Shao, C. (2017). What happens next? future subevent prediction using contextual hierarchical lstm. In *Proc. AAAI conf. artif. intell.*.

Hu, X., Zhang, K., Liu, K., Lin, X., Dey, S., & Onori, S. (2020). Advanced fault diagnosis for lithium-ion battery systems: A review of fault mechanisms, fault features, and diagnosis procedures. *IEEE Industrial Electronics and Magazine*, *14*(3), 65–91. http://dx.doi.org/10.1109/MIE.2020.2964814.

Jaeger, H. (2007). Echo state network. *Scholarpedia*, *2*(9), 2330.

Japkowicz, N., et al. (2000). Learning from imbalanced data sets: a comparison of various strategies. In *Proc. AAAI workshop, Vol. 68* (pp. 10–15).

Joshi, M. V., Kumar, V., & Agarwal, R. C. (2001). Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Proc. IEEE inter. conf. data mining* (pp. 257–264).

Khoshgoftaar, T. M., Seiffert, C., Van Hulse, J., Napolitano, A., & Folleco, A. (2007). Learning with limited minority class data. In *Proc. int. conf. machine learn. app.* (pp. 348–353).

Křivík, P. (2018). Methods of SoC determination of lead acid battery. *Journal of Energy Storage*, *15*, 191–195.

Lee, G., Kim, J., & Lee, C. (2022). State-of-health estimation of Li-ion batteries in the early phases of qualification tests: An interpretable machine learning approach. *Expert Systems with Applications*, *197*, Article 116817.

Liang, X., Song, X., Qi, K., Li, J., Liu, J., & Jian, L. (2021). Anomaly detection aided budget online classification for imbalanced data streams. *IEEE Intelligent Systems*, *36*(3), 14–22.

Lin, S., & Runger, G. C. (2017). GCRNN: Group-constrained convolutional recurrent neural network. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(10), 4709–4718.

Lines, J., Davis, L. M., Hills, J., & Bagnall, A. (2012). A shapelet transform for time series classification. In *Proc. ACM int. conf. knowl. disc. data mining* (pp. 289–297).

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining Knowledge Discovery*, *1*(1), 14–23.

Lunardon, N., Menardi, G., & Torelli, N. (2014). ROSE: A package for binary imbalanced learning. *R Journal*, *6*(1).

Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, *4*(2), 1883.

Pisner, D. A., & Schnyer, D. M. (2020). Support vector machine. In *Machine Learning* (pp. 101–121). Elsevier.

Rakthanmanon, T., & Keogh, E. (2013). Fast shapelets: A scalable algorithm for discovering time series shapelets. In *Proc. SIAM int. conf. data mining* (pp. 668–676).

Rigo, C. A., Seman, L. O., Camponogara, E., Morsch Filho, E., & Bezerra, E. A. (2021). A nanosatellite task scheduling framework to improve mission value using fuzzy constraints. *Expert Systems with Applications*, *175*, Article 114784.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, *24*(5), 513–523.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2008). Building useful models from imbalanced data with sampling and boosting.. In *Proc. FLAIRS conf.* (pp. 306–311).

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2009). RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems Man Cybernetics-Part A: Systems and Humans*, *40*(1), 185–197.

Van Hulse, J., Khoshgoftaar, T. M., & Napolitano, A. (2007). Experimental perspectives on learning from imbalanced data. In *Proc. int. conf. machine learn.* (pp. 935–942).

Wang, D., Qu, X., Yao, Y., & Yang, P. (2020). Hybrid inductive-power-transfer battery chargers for electric vehicle onboard charging with configurable charging profile. *IEEE Transactions on Intelligent Transportation Systems*, *22*(1), 592–599.

Weiss, G. M. (2004). Mining with rarity: a unifying framework. *ACM SIGKDD Explorations Newsletter*, *6*(1), 7–19.

Weiss, G. M., & Provost, F. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of the Artificial Intelligence Research*, *19*, 315–354.

Wright, R. E. (1995). Logistic regression.

Wu, T., Ji, F., Liao, L., & Chang, C. (2019). Voltage-SOC balancing control scheme for series-connected lithium-ion battery packs. *Journal of Energy Storage*, *25*, Article 100895.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-c. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Proc. advances neur. info. process. syst.* (pp. 802–810).

Yang, C., Qiao, J., Han, H., & Wang, L. (2018). Design of polynomial echo state networks for time series prediction. *Neurocomputing*, *290*, 148–160.

Ye, L., & Keogh, E. (2011). Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining Knowledge Discovery*, *22*(1), 149–182.

Zhang, Y. (2019). ATTAIN: Attention-based time-aware LSTM networks for disease progression modeling. In *Proc. inter. joint conf. artif. intell.*.

Zhao, W., Zhang, Y., Zhu, Y., & Xu, P. (2021). Anomaly detection of aircraft lead-acid battery. *Quality and Reliability Engineering International*, *37*(3), 1186–1197.