REPOSITORIO UNIVERSIDADE CORUÑA

UNIVERSIDADE DA CORUÑA

This is an ACCEPTED VERSION of the following published document:

Meira, J., Veloso, B., Bolón-Canedo, V., Marreiros, G., Alonso-Betanzos, A., & Gama, J. (2023). Data-driven predictive maintenance framework for railway systems. *Intelligent Data Analysis*, 27(4), 1087–1102. https://doi.org/10.3233/ida-226811

Link to published version: https://doi.org/10.3233/ida-226811

General rights:

This version of the article has been accepted for publication, after peer review. The Version of Record is available online at: https://doi.org/10.3233/ida-226811.

Intelligent Data Analysis (2023)
IOS Press

Data-Driven Predictive Maintenance Framework for Railway Systems

Jorge Meira [a,*], Bruno Veloso [b] Verónica Bolón-Canedo [c] Goreti Marreiros [a]
Amparo Alonso-Betanzos [c] and João Gama [b]

[a] *GECAD, Polytechnic Institute of Porto (ISEP/IPP), Porto, Portugal*
E-mails: janme@isep.ipp.pt, mgt@isep.ipp.pt

[b] *LIAAD, INESC TEC, Porto, Portugal*
E-mails: bruno.miguel.veloso@gmail.com, jgama@fep.up.pt

[c] *LIDIA - CITIC, University of Coruña, Coruña, Spain*
E-mails: veronica.bolon@udc.es, amparo.alonso.betanzos@udc.es

Abstract. The emergence of the Industry 4.0 trend brings automation and data exchange to industrial manufacturing. Using computational systems and IoT devices allows businesses to collect and deal with vast volumes of sensorial and business process data. The growing and proliferation of big data and machine learning technologies enable strategic decisions based on the analyzed data. This study suggests a data-driven predictive maintenance framework for the air production unit (APU) system of a train of *Metro do Porto*. The proposed method assists in detecting failures and errors in machinery before they reach critical stages. We present an anomaly detection model following an unsupervised approach, combining the Half-Space-trees method with One Class K Nearest Neighbor, adapted to deal with data streams. We evaluate and compare our approach with the Half-Space-Trees method applied without the One Class K Nearest Neighbor combination. Our model produced few type-I errors, significantly increasing the value of precision when compared to the Half-Space-Trees model. Our proposal achieved high anomaly detection performance, predicting most of the catastrophic failures of the APU train system.

Keywords: Anomaly Detection, Data Streams, Unsupervised Learning, One Class Classification, Predictive Maintenance

1. Introduction

Predictive maintenance (PdM) is a method that uses real-time analytic tools to assess collected data from various parts of one industrial machine [1]. The goal is to detect malfunctions as quickly as possible and fix them before they lead to a catastrophic failure. Anomaly detection lies at the core of PdM, with the primary focus on finding anomalies in the working components of machines at early stages and alerting supervisors to carry out maintenance activities [2].

This work describes a data-driven predictive maintenance system to detect anomalies on an Air Production Unit (APU) installed on trains of Metro of Porto. The goal is to identify as early as possible potential failures and notify the maintenance team of an anomaly (undetectable with traditional maintenance criteria), avoiding the inconvenience of removing a train from the operation and saving time and money for the company.

The data is collected from the APU using a set of analogic sensors and reading directly from the APU control system some digital signals that control the state of the APU. We receive the data in regular time

*Corresponding author. E-mail: janme@isep.ipp.pt.

1088-467X/23/ © 2023 – IOS Press and the authors. All rights reserved

intervals, and the learning process extracts information in near real-time to build a predictive model. The model can send an alarm to the maintenance teams, allowing timely intervention on the train.

In this work, we propose an online predictive model capable of dealing with incoming stream data with adaptive learning properties. Since the data incoming from the sensors is endless and received as a continuous flow, we choose to deepen the data stream mining topic, where the methods' computational resources are limited (memory, computational power, processing time). These methods are based on incremental learning as data is induced incrementally and contemplate a forgetting mechanism to deal with limited memory. They differ from batch learning models such as Deep Neural Networks, which are static, computational power is usually a must to get the best fitting in data, and the learning process is performed offline.

Furthermore, we followed a semi-supervised learning approach since we did not know when train failures occurred at the beginning of the project.Therefore, we have combined two methods, the *Half Space Trees (HS-Trees)* algorithm for one-class anomaly detection in evolving streams [3] and an adaptation of the *K-Nearest Neighbour* [4, 5] capable of doing one-class classification in streaming data.

The main idea of our proposal is to use *HS-Trees* as the primary anomaly detector method to filter the incoming data. *HS-Trees* sends the observations detected as anomalies to the *One-Class K-Nearest Neighbour* method to reduce false positives. Our model presented high-performance results, detecting most of the catastrophic failures and producing fewer false positives compared to the *HS-Trees* method.

The paper is organized as follows: we provide an overview of the related work in the context of anomaly detection in Section 2. Section 3 describes the algorithms implemented in our proposal for fault detection using an semi-supervised learning approach. Section 4 describes the data used, the problem definition and the detailed description of our proposal. Section 5 presents the anomaly detection results of our model. Finally, Section 6 points out the conclusions, remarks, and future works.

2. Related Work

Using sensors to monitor industrial equipment combined with the emergence of high-speed networks like 5G and computational systems allowed the development and adaptation of machine learning techniques to anomaly detection and predictive maintenance. In this section, we will present some studies regarding these two topics.

Maintenance in industrial equipment and repair procedures are typically responsive to a not-predicted issue. Since malfunctions in equipment affect the safety, availability, and environment, the authors in [6] proposed a real-time monitor to schedule monitoring tasks. These tasks obtain sensor information, measure the state and condition of several components, and determine when the most appropriate moment is to apply a preventive maintenance action (predictive maintenance) on the equipment. The predictive maintenance topic has been attracting growing interest over the last years with several proposals exploring different machine learning methods for predictive maintenance or anomaly detection [6–18]. More recently, a survey proposed by [19] analyses all the related work regarding the usage of machine learning techniques for predictive maintenance on the railway industry.

Industrial equipment often lacks sufficient and diverse anomalous data to build a binary classification system. Thus many of the predictive maintenance models rely on unsupervised anomaly detection algorithms, which are responsible for determining whether an observation of the sensor deviates from the normal state of the equipment [20, 21]. Detecting the presence of anomalies in real-time provides valuable insights and knowledge about the equipment to make a rigorous assessment of possible maintenance

interventions. There are several works in the literature related to the topic of predictive maintenance in railway systems, and they can be organized into supervised or unsupervised learning approaches:

Supervised Learning

Rabatel et al. [7] explored the application of sequential patterns to correctly identify normal and abnormal data generated by a set of sensors installed in three key train components.

Li et al. [8] proposed a five-step predictive maintenance framework. The first step is the feature extraction of the dataset containing information about bearings on the train. The second step is reducing dimensional space using the Principal Component Analysis. The model adopted was the Support Vector Machine. Finally, a confidence level for alarm prediction was defined, and a rule simplification divides the feature space into non-overlapping small grids.

In terms of predicting failures on door trains, Manco et al. [11] developed an application to predict and explain door failures using an outlier detection method. Pereira et al. [9] developed a failure detection system for classifying irregular open/close cycles within trains based on the difference between the inlet and outlet pressure in specific intervals of the cycle. More recently, Ribeiro et al. [10] explored data-driven PdM based on anomaly and novelty detection implemented to predict failure in the automatic door system. The results showed that a low-pass filter could significantly reduce the number of false alarms.

Fumeo et al. [22] described a condition-based maintenance algorithm that explores the online support vector regression algorithm to predict the remaining useful life of the railway vehicle. In particular, the authors aim to detect failures on the axle bearings as soon as possible.

Wan-Jui Lee [12] used the Linear Regression model to describe two different compressor operations (idle and running time). The authors used logistic functions to define the boundaries of the two classes or compressor operations modes. The system is used for air leakage detection by anomaly detection in a train's braking pipes. They used a density-based clustering method with a dynamic threshold to distinguish anomalies.

Bukhsh et al. [15] explored the usage of tree-based models like Random Forest, Decision trees, or XGBoost to predict the status of railway switches. Additionally, the authors explored the Local Interpretable Model-Agnostic Explanations (LIME) to explain the possible reasons for the malfunction. Kalathas and Papoutsidakis [18] applied two well-known classification algorithms, the J48 and M5P, to monitor the health state of traction and braking subsystems of the Greek Railway. Adopting tree algorithms helps the maintenance teams understand the reason for the malfunction.

Kang et al. [13] described a system that uses a Bayesian statistical learning model to represent the expected behaviour of the train in terms of speed. The study's main objective was to capture changes and anomalies in the trains' speed to detect some malfunctions as early as possible.

Barros et al. [16] proposed adopting a rule-based system to detect anomalies on a train compressor unit. This system monitored several analogical and digital variables and then used a low pass filter to smooth the analogical signals and count the number of peaks in a time window. The rules were designed based on the maintenance teams' expertise to define the compressor units' normal state.

Unsupervised Learning

Salierno et al. [17] proposed architecture for predictive maintenance on the railway domain. The proposed architecture is to predict failures in the interlocking railway system of the Italian Railway. The authors adopted a Long Short Term Memory model to capture abnormal patterns of the interlocking system.

Davari et al. [23] describe a sparse autoencoder (SAE) network for predictive maintenance on a metro railway domain. The proposed autoencoder is designed to predict failures on the air compressor subsystem to remove the train from circulation safely.

Chen et al. [14] presented a predictive system for the compressor air unit. The authors used a recurrent neural network using Long Short-Term Memory architecture for failure prediction. The authors compared their method with the random forest method, and the results showed that the neural network proposal was more stable when compared with the Random Forest.

All the described related works (summarized in table 1) rely on identifying the normal state of the system/component, considering as possible anomalies the observations that do not have the same familiar patterns. Different machine learning models or techniques were applied depending on the context and characteristics of the equipment.

Our approach differs from the state of the art because it relies on machine learning techniques to identify abnormal patterns correctly. The supervised approaches presented in this section do not work in real-time because we do not know the ground truth. When we compare with unsupervised learning approaches, where some authors look to the autoencoders' higher values of reconstruction error to signal an anomaly, we suffer from a false positive alarms problem. Our method relies on a semi-supervised learning algorithm, HS-Tree, which learns a single class and classifies all the other classes as an anomaly.

If the output of one observation is positive for an anomaly, we use a *kNN* algorithm to see if the observation is distant from known normal observations of the air compressor unit. The ablation study in this manuscript shows a significant improvement in the evaluation metrics.

Table 1
Related Work Comparison

	Target System	Model	Explainable Model	Evaluation Metric
[7]	Train	Sequence Patterns	No	Recall & Precision
[11]	Doors	Outlier	No	AUC
[9]	Doors	LPF	No	False Alarm Rate & Impostor Pass Rate
[10]	Doors	LPF	No	Reduced False Alarm Rate & Reduced Impostor Pass Rate & Detection Error
[12]	Air Compressor	Linear Regression	No	RMSE & Confusion Matrix
[13]	Train	Bayesian Model	No	Error
[14]	Air Compressor	RNN / LSTM	No	F-Measure & AUC & Accuracy & Recall & Precision
[16]	Air Compressor	Rule-based	No	F-measure
[15]	Railway switches	XGB / RF / DT	Yes	Accuracy & F-Measure & Kappa
[18]	Railway switches	J48 / M5P	Yes	Recall & Precision & Matthews Correlation Coefficient
[8]	Bearings	SVM	No	Accuracy & Recall & Precision
[22]	Axle bearings	SVR	No	Mean Absolute Percentage Error
[17]	Interlocking	LSTM	No	Error
[23]	Air Compressor	SAE	No	F-measure

3. Background

This section presents the algorithms employed to build the proposed model and detect the train system's catastrophic failures.

3.1. Half Space Trees

Half-Space-Trees (HS-Trees) is a one-class anomaly detector algorithm built to deal with data streaming environments [3]. *HS-trees* is an ensemble that learns incrementally as the data arrives, capable of performing unsupervised learning in evolving data. The basic concept of this algorithm is to create binary trees by partitioning data into sub-spaces or regions. When generating a tree, the algorithm selects a random dimension and splits it into disjoint, equal-volume halves, thus creating the left and right side child nodes. This process is repeated until the tree reaches a maximum depth (user-specified parameter). Therefore, any data point in the domain travels a single path from the tree's root to the leaves going through the different sub-spaces. Although the data is splitted into density sub-spaces, this algorithm differs from clustering algorithms such as SOM (self-organizing map) [24] since *HS-trees* output are scores of anomalous values estimated by the density regions, SOM computes clusters regions through distance based-techniques. The *HS-trees* captures the mass in each node, representing the number of data points, and uses it to profile each data point's anomaly estimation.

The data-stream is partitioned into windows of equal sizes, named *reference window* and *latest window*. In *reference window* the algorithm learns the mass profile (r value stored in each node) to infer the anomaly scores of new data points arriving in the *latest window* [3]. According to the mass profile values, a data point is considered normal for high mass regions, while anomalous data points fall into low mass regions. Also, the *latest window* (l value stored in each node) records the mass profile, and once this window is full, that is, after a set of data points are recorded in the new mass profile, the *latest window*, overrides the old mass profile and becomes the *reference window*. Thus *reference window* always keeps the latest mass profile, used to evaluate the new data that arrives in the stream. This process repeats until the end of the stream.

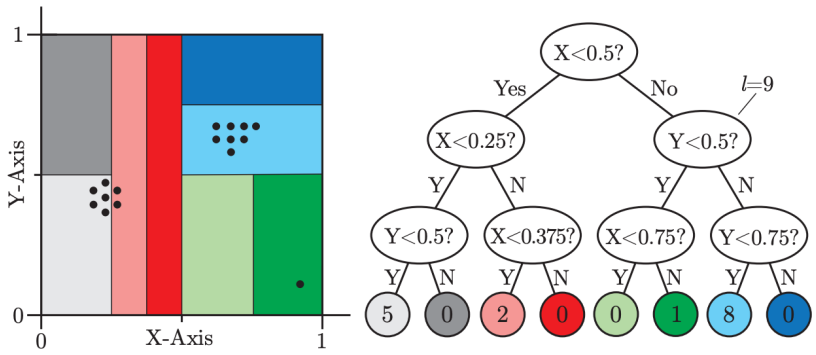


Fig. 1. HS-trees example by [25] and a recorded latest mass profile. The left image represents the data partitioned, and the right image the HS-tree generated.

Figure 1 shows a representation of a *HS-tree*. The HS-tree partitions the domain dimensions with range values [0,1] on the left side, where dots represent data points. The right side of Figure 1 shows

an HS-tree with a three-depth level. The inner nodes contain information regarding the splitting values and which dimension was partitioned, leaves contain mass profile values of the latest window. Also, it is possible to observe the mass profile value stored in the root's right child ($l = 9$). This value is incremented by one when a newly arrived data point traverses it on its path to a leaf.

To make the algorithm more robust, $t \in \mathbb{N} > 0$ HS-trees are created and combined to form an ensemble algorithm, reducing the spread or dispersion of predictions, thus, improving performance. Also, a random perturbation is assigned to each feature space when split in the tree's root, providing diversity to the algorithm.

Each tree in the ensemble computes a score for each data point independently regarding the anomaly score. The computed score allows the new arrival data point to traverse through the tree's nodes until it reaches the leaves or a node with a mass profile equal to or less than a user-determined value s denoted by the size limit (minimum mass required in a node). The final anomaly computed score is the sum of all anomaly scores of the HS-trees.

3.2. One-Class K Nearest Neighbour

The One-Class K Nearest Neighbour (OCKNN) [5] is an adaptation of the original K-Nearest Neighbour algorithm [4] for supervised learning using distances between neighbours to classify the data. The K in OCKNN is the number of nearest neighbours, the core deciding factor. This algorithm is a lazy learning algorithm since it does not learn in the training phase, where all data points are used at the time of prediction. This algorithm only stores or memorises the data points in the training phase and waits until classification is performed. It is named One-Class since it requires only one class in the training phase. After memorising all data points, in the prediction phase, the algorithm uses a user-specified distance metric such as euclidean, Manhattan, Chebyshev, Minkowski, Minkowski or Mahalanobis to compute distances between points in the domain space. The mean distance between the new data point and its k nearest neighbours from the training set is calculated.

The mean distance value of each data point to its k neighbours represents the anomaly score. An anomalous classification is made if a distance value for a specific data point is more significant than a $d \in \mathbb{R} > 0$ distance threshold (user-specified parameter).

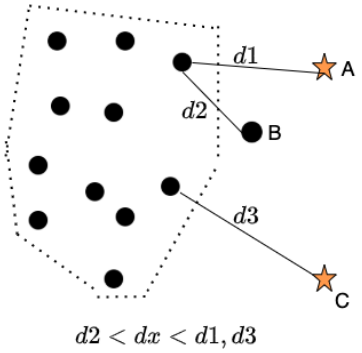


Fig. 2. OCKNN illustration example in two dimensional space, where $k = 1$; $d1, d2, d3$ are distances of points A, B and C respectively, to their nearest neighbour; dx is the distance threshold to consider a given data point to be anomalous.

It can be observed in Figure 2 how the OCKNN algorithm works for one neighbour ($K = 1$). The black dots within the drawn area represent the normal data points. The stars (A and C) are data points

classified as anomalous since distance $d1$, and $d3$ are more significant than the distance threshold dx, while data point B is classified as normal as distance $d2$ is minor than dx.

4. Methodology

4.1. Problem Definition

The Air Production Unit (APU) is part of a compressed air system, which produces pressurizing air from an electric motor. The electrical current consumed by the motor is converted into kinetic energy. The compressed air system is a crucial component of the train and delivers essential pressurized air to several clients like pneumatic suspension, oil injection on the rail to reduce the friction and noise on the curves, and injection of sand to gain traction rails, and finally, connect other trains. Applying predictive maintenance here is essential to predict the equipment failure before it happens, decreasing costs and optimizing the service.

4.2. Trains Data

The data acquisition system collects information from several analogical sensors and digital signals generated by the APU control system. Based on the failure history of the train fleet, it is possible to identify the critical components of the system that generate the majority of the failures. These critical components are: (i) electrical valve; (ii) pressure valve; (iii) oil leaks; (iv) electrical motor; (v) pressure switches; and (vi) drying towers. The sensors and places to install them were strategically defined, considering the output of the failure history study. Figure 3 shows an overview of the train system.

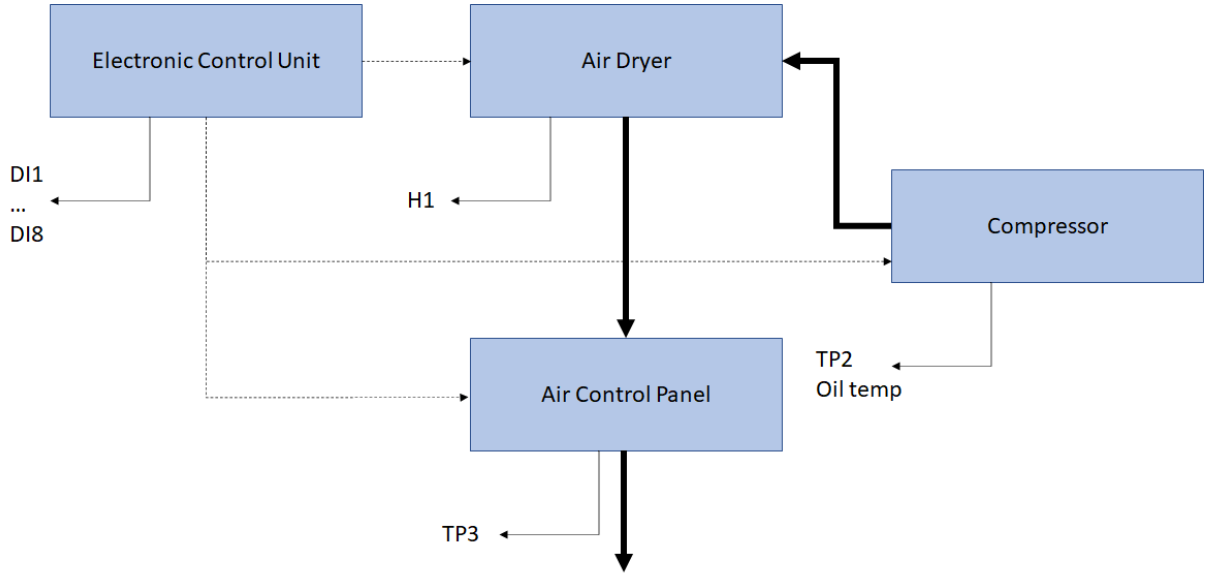


Fig. 3. Train System: dark arrows represent the pneumatic system, dashed arrows the control system and the thin black arrows the sensors

The data acquisition system communicates with a cloud server that receives the data from the sensors with 1 Hz of sampling frequency. The system stores the data collected from the sensors and respective

timestamps to a data logger file, and every five minutes, the file is sent to the server using the TCP/IP protocol application.

The considered analogical sensors were the following.

- TP2 - Measures the pressure on the compressor.
- TP3 - Measures the pressure generated at the pneumatic panel.
- H1 - This valve is activated when the pressure read by the pressure switch of the command is above the operating pressure of 10.2 bar.
- DV pressure - Measures the pressure exerted due to pressure drop generated air dryers towers, and when it is equal to zero, the compressor is working under load.
- Motor Current - Measures the current of one phase of the three-phase motor, which should present values close to 0 A when the compressor turns off, close to 4 A when the compressor is working offloaded and close to 7 A when the compressor is working under load. When the compressor starts to work, the motor current presents values close to 9 A.
- Oil Temperature - Measures the temperature of the oil present on the compressor
- Flowmeter - Measures the airflow that leaves the APU for Reservoirs

The considered digital sensors were the following.

- COMP - The electrical signal of the air intake valve on the compressor. It is active when there is no admission of air on the compressor, meaning that the compressor turns off or working offloaded.
- DV electric - the electrical signal that commands the compressor outlet valve. When it is active, it means that the compressor is working under load; when it is not active, it means that the compressor is off or offloaded.
- TOWERS - Defines which tower is drying the air and which tower is draining the humidity removed from the air. When it is not active, it means that tower one is working; when it is active, it means that tower two is working.
- MPG - Is responsible for activating the intake valve to start the compressor under load when the pressure in the APU is below 8.2 bar. Consequently, it will activate the sensor COMP, which assumes the same behaviour as MPG sensor.
- LPS - Is activated when the pressure is lower than 7 bars.
- Oil Level - Detects the oil level on the compressor and is active (equal to one) when the oil is below the expected values.

4.3. Proposed model

For our proposal, we only considered the analogical sensors data arriving in the stream recorded at each second. Figure 4 illustrates our anomaly detection model for predicting catastrophic failures.

Before feeding *HS-Trees* algorithm, we aggregated the data in minutes through the timestamp feature. This operation extracted each sensor's mean, median, standard deviation, and variance. Our experiences found that the information extracted by each minute was sufficient to prevent the *HS-Trees* algorithm from losing performance, thus optimizing the data processing time as it computes fewer records.

After running several experiments with *HS-Trees*, we noticed that this method was generating a large number of false positives since only 4 % of data was reported as a failure, while *HS-Trees* was detecting around 25 % of failures. To tackle this problem, we adopted the *OCKNN* algorithm to deal with data arriving continuously. The idea is that the *OCKNN* evaluates each anomaly detected observation from *HS-Trees* to check if it was detected correctly.

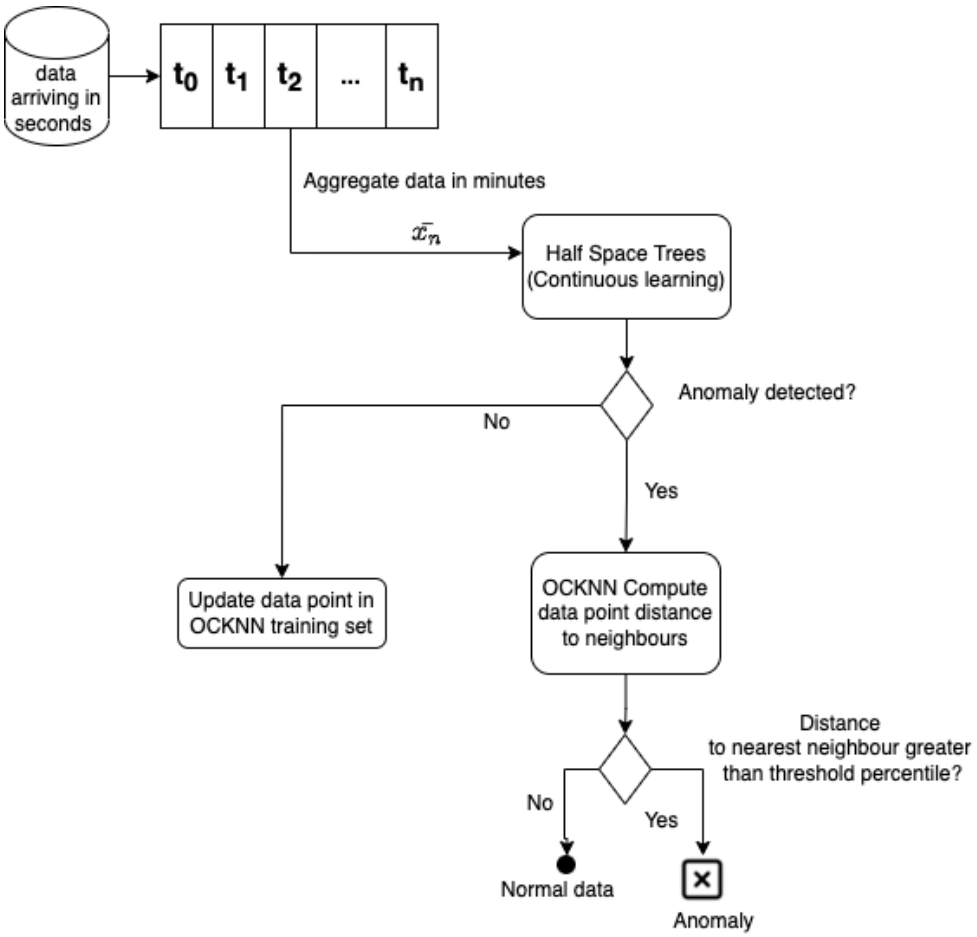


Fig. 4. Proposed methodology

Data points are updated in the *OCKNN* training set if *HS-Trees* inferred these points as normal data. The update process considers the maximum and minimum distances to neighbour's values captured during the stream. Distant normal points to its neighbours are added to the training set while neighbour points with the lowest distance are removed. This update mechanism showed high-performance results as stacking points with high distances present high sensitivity when detecting anomalous data.

In the case of *HS-Trees* inferred points as anomalous, the *OCKNN* method calculates the distance from each arriving data point to its closest neighbour to verify whether they are at an abnormal distance. To better understand our model, it is presented in Algorithm 1 the pseudo-code implementation.

Before starting the data stream, initial parameters and data structures were defined: A set of initial training data with 1400 records for the *OCKNN* method, which represents a whole day stack (24h) from a period that we know there was no anomaly in the train system; a K number of neighbours to compute distances to the arriving data points from which we only used 1 neighbour; a $dist_{Max}$ variable to record the maximum distance to its neighbour set as 0 (lowest value to be replaced in the first iteration) and a $dist_{Min}$ variable to record the minimum distance to its neighbour set as 9999 (a high value to be replaced in the first iteration). The output of our model returns a list of TimeStamp values that indicate when an

Algorithm 1: Pseudo-code for HS-trees with OCKNN approach.

Input : $D_i \leftarrow$ OCKNN set of initial training points
$D_s \leftarrow$ Data Stream
$dist_{Max} \leftarrow$ Maximum distance to neighbours
$dist_{Min} \leftarrow$ Minimum distance to neighbours
$K \leftarrow$ Number of OCKNN neighbours
Output: $S \leftarrow$ Anomalies TimeStamp

$S \leftarrow 0$, $D_i \leftarrow 1400$, $dist_{Max} \leftarrow 0$, $dist_{Min} \leftarrow 9999$, $K \leftarrow 1$, $OCKNN.fit(D_i)$;
while D_s *continues* **do**
  Receiving the next streaming point x;
  $dist \leftarrow OCKNN.ComputeDistance(x)$;
  $distPercentile \leftarrow distMean + zScore * distSTD$;
  $predict \leftarrow HSTrees.predict(x)$;
  if $predict == anomaly$ **then**
    if $dist > distPercentile$ **then**
      $S.append(x.Timestamp)$;
    end
    if $dist > dist_{Max}$ **then**
      $dist_{Max} = dist$;
    end
  end
  if $predict == normal$ **then**
    if $dist < dist_{Min}$ **then**
      $dist_{Min} = dist$;
      $dist_{MinIndex} = closestNeighbour.index$;
    end
    if $dist > dist_{Max} * 0.75$ **then**
      $D_i.drop(x[dist_{MinIndex}])$;
      $D_i.append(x)$;
      $OCKNN.fit(D_i)$;
      $dist_{Max} = 0$;
      $dist_{Min} = 9999$;
    end
  end
  $HSTrees.partialFit(x)$;
end
Result: S

anomaly has occurred in the APU train system.

The data stream cycle starts in Line 2, where variable x is assigned to each arrival data point. The algorithm starts by computing the distance to the nearest neighbour, in line 8. Then it computes the distance percentile, which is used as a threshold to identify anomalies (line 5), employing a Zscore table value of 2.326 representing percentile 99%, which means detecting 1% of observations with high

distance to neighbours. Then, the *HS-Trees* method starts by inferring the arrival data point (line 6), checking if it corresponds to an anomaly (line 7). If the data point is considered anomalous, it validates if the distance to its neighbour is greater than the threshold percentile (line 8). If confirmed, the TimeStamp value is stored in the list S (line 9). Also, the algorithm records the $dist_{Max}$ assigning its value to the nearest neighbour distance of the current data point if that distance is greater than the previous $dist_{Max}$ value.

In case *HS-Trees* infers the arrival data point as normal behaviour (line 12) the algorithm assigns $dist_{Min}$ and $dist_{MinIndex}$ as the current data point distance to its neighbour and the neighbour id respectively only if that distance is less than previous $dist_{Min}$ value. The next validation is performed to update the *OCKNN* training set with points identified as a normal activity of the train system by the *HS-Trees* if it matches a certain condition. The data is incremented in the *OCKNN* training set if the current normal data point distance to its neighbour is greater than the third quartile of $dist_{Max}$ value. The algorithm also discards from *OCKNN* training set the nearest neighbour data point with the lowest distance. Then, $dist_{Max}$ and $dist_{Min}$ values are reset. These operations are listed from line 16 to 21. Finally, *HS-Trees* is incrementally fitted for each data point in order to build the mass profile used to estimate anomalies (line 22).

As anomalous events are rare, we define a threshold value representing 1 % of the arriving data with the highest distance values to its nearest neighbour. This threshold parameter value allowed our method to detect most anomalous periods generating few type I and II errors. Figure 5 shows the anomalous data points detected by our method in one of the performed experiments. Distance values equal to zero represent data points classified as normal behaviour, while distance values greater than zero are the anomalies detected by our method. The colours represent the real meaning of the data. In red are the data points that correspond to the real anomalies, and in blue, the data points that correspond to the real normal behaviour of the train system.

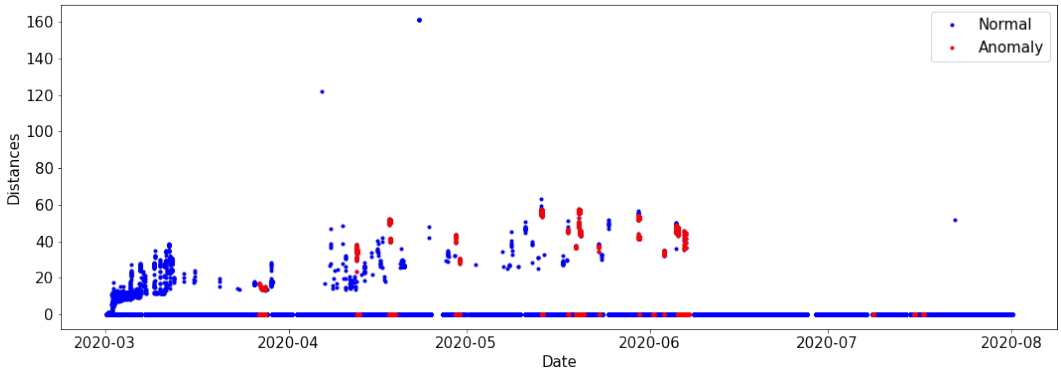


Fig. 5. Anomalies detected by our method

It can be seen in Figure 5 a set of normal values detected as anomalies probably due to the initial fit of the *HS-Trees* model to the data distribution. The model classifies fewer observations as anomalous from mid-March, identifying practically part of all anomalous periods with only a few examples represented by normal activity (false negatives). This model was developed with python, using the implementation of

the *HS-Trees* algorithm from the scikit-multiflow[1] library [26] and the implementation of the *K Nearest-Neighbour* algorithm from the scikit-learn[2] library [27] adapted to work as one class classification with online data.

5. Model Evaluation

In this section, we evaluate our model and report the result of our experiments. We evaluated the model's effectiveness using data from a train in operation in 5 months of 2020, with some catastrophic failures reported during that period. The data contains 21 periods reported as anomalous. Some last a few minutes, others a couple of hours.

5.1. Evaluation Procedure

In order to evaluate the performance of our approach, five experiments were carried out with some state-of-the-art anomaly detection algorithms in the context of data streams, using the data from the analogical sensors present in the APU system. Therefore, the mean, median, standard deviation and variance from the DV_pressure, TP2, TP3, H1, Oil_temperature, Motor_current and mode were used. The last feature concerns the status of the train. This feature has three states: in progress, stopped, and under maintenance. Maintenance status data has been discarded as tests are performed on the trains, causing the APU system to generate anomalous values, misleading the model's predictions. It is also important to mention that all data were normalized using the standard window scaling technique, which standardizes features by removing the mean and scaling to unit variance. The mean and standard deviation are computed on a given window frame.

Regarding the algorithms, we tested our approach (**HSTreeOCKNN**) against anomaly detection methods for data streams such as: **Half-Space-Trees (HSTrees)** [3], **XStream** [28], **Isolation Forest (IForestASD)** [29] and **ExactStorm** [30].

To assess the models, we verified that the detected anomalies were within the reported anomalous period, as shown in Figure 6. If for a given model, there is an overlap in its output to the ground truth (in that anomalous period is detected more than one anomaly), then all observations from that period are counted as anomalous (True Positive) in the model's output. Note that the results of our methodology were validated by experts at *Metro do Porto*.

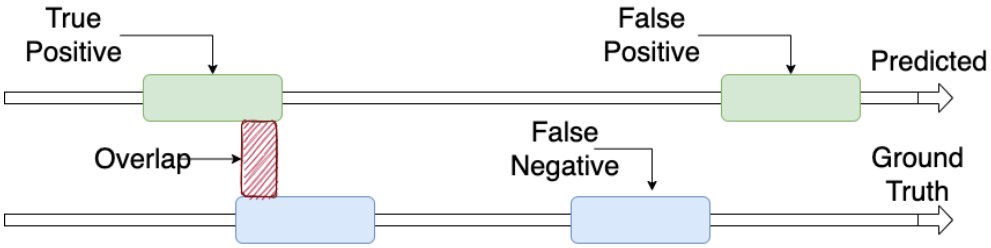


Fig. 6. Models Validation approach.

We performed several experiments to adjust the hyperparameters reaching the settings in Table 2.

[1] https://scikit-multiflow.github.io/
[2] https://scikit-learn.org/stable/

Table 2
Selected hyperparameters

	HSTreeOCKNN	HSTree	IForestASD	XStream	ExactStorm
N Estimators (Chains)	15	15	25	15	-
Window Size	400	400	400	400	400
Depth	10	10	-	7	-
size limit	75	75	-	-	-
Max Radious	-	-	-	-	0.15
N Components	-	-	-	20	-

We used the accuracy, Precision, Recall, and F1 metrics for model evaluation, giving the necessary information to analyze the type I and type II errors. We can know how many observations from both classes (normal and abnormal) were correctly identified with accuracy. Equation 1 shows accuracy formulation, being:

- True Positive (TP) - The number of observations correctly identified as an anomaly;
- False Positive (FP or Type I error) - The number of observations classified as an anomaly but corresponding to normal activity;
- True Negative (TN) - The number of observations correctly identified as normal activity;
- False Negative (FN or Type II error) - The number of observations classified as a normal activity but corresponding to anomalies;

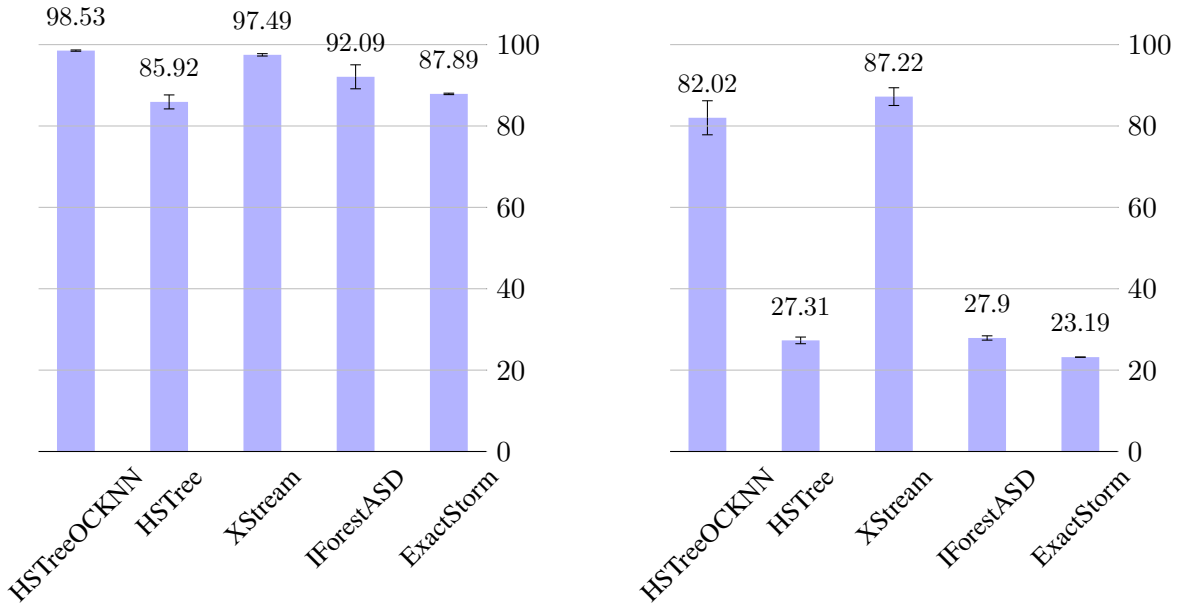
$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

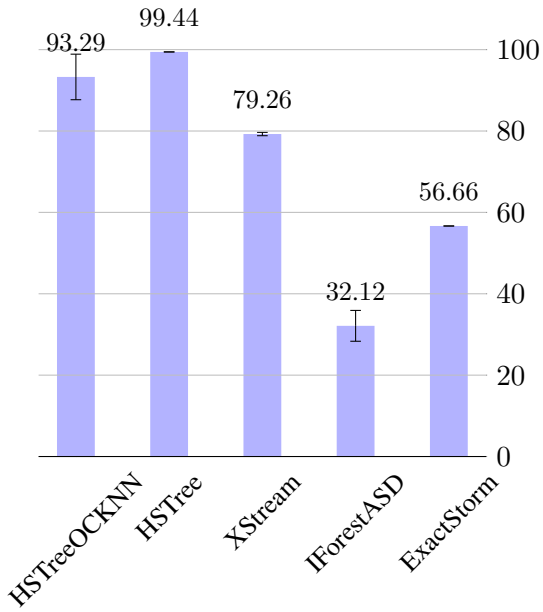
$$F1score = 2 * \frac{recall * precision}{recall + precision} \tag{4}$$

Regarding the precision and recall metrics in equation 2 and 3, both measure the rate of FP and FN, respectively. For instance, a high recall value means low FN, while a small precision indicates high FP values. To analyze the balance of these two metrics, we compute the F1 score as the harmonic mean of Precision and Recall. While it is possible to take a simple average of the two scores, harmonic means are more resistant to outliers. Thus F1 score in equation 4 is a balanced metric that appropriately quantifies the correctness of models.

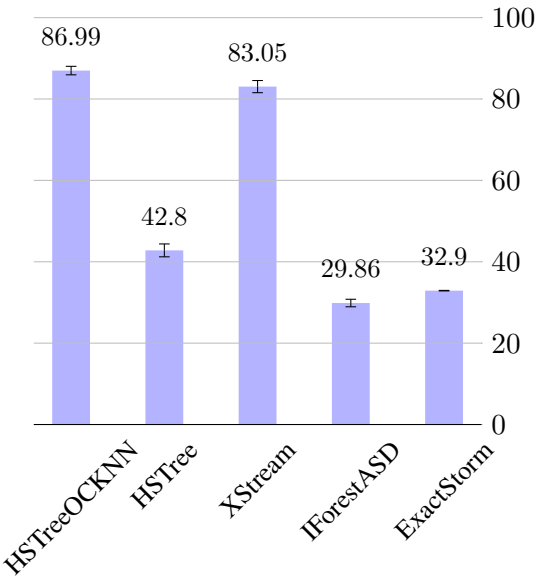


(a) Accuracy%

(b) Precision%



(c) Recall%



(d) F1 Score%

Fig. 7. Performance results of the methods using the metrics Accuracy (a), Precision (b), Recall (c) and F1 Score (d)

5.2. Discussion

First, we start by analyzing the results of the models in Figure 7(a) where metric accuracy was used. We can observe that our model was the best, reaching an accuracy of around 98 %, followed by the **XStream** algorithm that achieved a 1% lower accuracy when compared to our approach. **IForestASD** was ranking third with an accuracy of 92 %, while **ExactStorm** and **HSTree** performed worst with an 87 % and 85 % accuracy respectively. A high accuracy value was expected since failures were rare, representing only 4 % of all data. However, this information is vague and insufficient to analyze the types of errors generated by the models.

Analyzing the remaining metrics, it can be seen in Figure 7(b) the percentage of type I errors generated by the models. In this case, the **XStream** was the best model with 87 % precision, followed by **HSTreeOCKNN** achieving 82 %. As **OCKNN** receives the output of **HS-Trees** to train and validate anomalies, we know a priori that there would be an increase in performance due to the elimination of false positives by the **OCKNN**. Therefore was expected an increase in the precision metric by our method, which is visible in Figure 7(b), representing a 55 % higher value if we only used **HS-Trees** model to tackle this problem. The other models presented a poor performance, with values below 30 % precision, which means that these models misclassified most observations as anomalous.

Analyzing Figure 7(c) concerning recall metric, the results presented by the models are much better (except for the **IForestASD**), meaning all methods generated low type II errors. The models should generate less FN than FP since these errors indicate that an anomaly was mistaken for normal activity in the APU train systems. Failure to classify anomalous activity into normal activity will cause the train to run into a catastrophic failure, leading to high repair costs and the sporadic closure of the railway. As it can be seen, **HSTree** could detect almost every anomaly with a high cost of FP (Figure **??**). **HStreeOCKNN** was placed in second with 93 % of recall, a small cost of generating FN compared to **HStree** in order to get a higher precision value. **XStream** did a decent job with a 79 % recall value. At last, **ExactStorm** with 56 % recall followed by **IForestASD** detecting only a few anomalies.

To analyze the balance between precision and recall metrics, we can observe Figure 7(d), which presents the model's performance evaluated by the F1 score metric. Therefore, our approach was the best, achieving 87 % F1 score, followed by **XStream** as it was observed in Figures 7(b) and 7(c), it had a slightly lower type I error rate than the **HSTreeOCKNN** model, but with a significantly higher type II error rate. **HSTree** shows a poor F1 score due to the high type I error rate, while **IForestASD** and **ExactStorm** presented the worst performance in both precision and recall metrics.

6. Conclusions

Predictive Maintenance enables more efficient, longer-term planning for maintenance operations and makes it easier to allocate maintenance resources and define operational maintenance goals. One of the most promising aspects of the railway industry's transformation is Predictive Maintenance through data collected on the equipment during operation to identify failures in real-time. Therefore, repairs can be adequately planned without unexpectedly taking trains out of service for emergencies or unnecessary routine Maintenance.

This paper presents a data-driven predictive maintenance framework for the APU train system of *Metro of Porto*. We used the *HS-Trees* method combined with *OCKNN* to build a predictive model capable of detecting catastrophic anomalies and dealing with streaming data.

Our empirical study shows that the use of *HS-Trees* provided significant performance improvements when used in conjunction with *OCKNN*. The proposed predictive model obtained high anomaly detection performance while maintaining fewer false positives and negatives compared to State-of-the-art methods. Distances from neighbours are a viable solution to reduce false positives for this problem. For future work, we intend to test the robustness of our model when drift occurs in data. This phenomenon is represented by a significant change in the data distribution, i.e., degradation as the train components age. Also, we will evaluate this methodology in other real-case scenarios related to Predictive Maintenance.

Acknowledgements

This research has been financially supported in part by the Spanish Ministerio de Economia y Competitividad (research project PID2019-109238GB-C22), and by the Xunta de Galicia (Grants ED431C 2018/34 and ED431G 2019/01) with the European Union ERDF funds. CITIC, as Research Center accredited by Galician University System, is funded by Consellería de Cultura, Educación e Universidades from Xunta de Galicia, supported in an 80% through ERDF Funds, ERDF Operational Programme Galicia 2014-2020, and the remaining 20% by Secretaría Xeral de Universidades (Grant ED431G 2019/01). This work was also supported by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within the Project UIDB/00760/2020 and UIDP/00760/2020. This work was supported by the CHIST-ERA grant CHIST-ERA-19-XAI-012, and project CHIST-ERA/0004/2019 funded by FCT.

References

[1] J. Yan, Y. Meng, L. Lu and L. Li, Industrial big data in an industry 4.0 environment: Challenges, schemes, and applications for predictive maintenance, *IEEE Access* **5** (2017), 23484–23491.
[2] P. Kamat and R. Sugandhi, Anomaly detection for predictive maintenance in industry 4.0-A survey, in: *E3S Web of Conferences*, Vol. 170, EDP Sciences, 2020, p. 02007.
[3] S.C. Tan, K.M. Ting and T.F. Liu, Fast anomaly detection for streaming data, in: *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.
[4] E. Fix, *Discriminatory analysis: nonparametric discrimination, consistency properties*, Vol. 1, USAF school of Aviation Medicine, 1985.
[5] D.M.J. Tax, One-class classification: Concept learning in the absence of counter-examples. (2002).
[6] A. Koons-Stapf, Condition Based Maintenance: Theory, Methodology, & Application (2015).
[7] J. Rabatel, S. Bringay and P. Poncelet, Anomaly detection in monitoring sensor data for preventive maintenance, *Expert Syst. Appl.* **38**(6) (2011), 7003–7015. doi:10.1016/j.eswa.2010.12.014.
[8] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang and A. Hampapur, Improving rail network velocity: A machine learning approach to predictive maintenance, *Transportation Research Part C: Emerging Technologies* **45** (2014), 17–26.
[9] P. Pereira, R.P. Ribeiro and J. Gama, Failure prediction – an application in the railway industry, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* **8777** (2014), 264–275.
[10] R.P. Ribeiro, P. Pereira and J. Gama, Sequential anomalies: a study in the Railway Industry, *Machine Learning* **105**(1) (2016), 127–153.
[11] G. Manco, E. Ritacco, P. Rullo, L. Gallucci, W. Astill, D. Kimber and M. Antonelli, Fault detection and explanation through big data analysis on sensor streams, *Expert Syst. Appl.* **87** (2017), 141–156. doi:10.1016/J.ESWA.2017.05.079. https://www.sciencedirect.com/science/article/pii/S0957417417304074.
[12] W.-j. Lee, Anomaly detection and severity prediction of air leakage in train braking pipes, *International Journal of Prognostics and Health Management* **21** (2017).
[13] S. Kang, S. Sristi, J. Karachiwala and Y. Hu, Detection of anomaly in train speed for intelligent railway systems, *2018 Int. Conf. Control. Autom. Diagnosis, ICCAD 2018* (2018).
[14] K. Chen, S. Pashami, Y. Fan and S. Nowaczyk, Predicting air compressor failures using long short term memory networks, in: *EPIA Conference on Artificial Intelligence*, Springer, 2019, pp. 596–609.

[15] Z.A. Bukhsh, A. Saeed, I. Stipanovic and A.G. Doree, Predictive maintenance using tree-based classification techniques: A case of railway switches, *Transportation Research Part C: Emerging Technologies* **101** (2019), 35–54.
[16] M. Barros, B. Veloso, P.M. Pereira, R.P. Ribeiro and J. Gama, Failure Detection of an Air Production Unit in Operational Context, in: *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning*, J. Gama, S. Pashami, A. Bifet, M. Sayed-Mouchawe, H. Fröning, F. Pernkopf, G. Schiele and M. Blott, eds, Springer International Publishing, Cham, 2020, pp. 61–74. ISBN 978-3-030-66770-2.
[17] G. Salierno, S. Morvillo, L. Leonardi and G. Cabri, An architecture for predictive maintenance of railway points based on big data analytics, in: *International Conference on Advanced Information Systems Engineering*, Springer, 2020, pp. 29–40.
[18] I. Kalathas and M. Papoutsidakis, Predictive Maintenance Using Machine Learning and Data Mining: A Pioneer Method Implemented to Greek Railways, *Designs* **5**(1) (2021), 5.
[19] N. Davari, B. Veloso, G.d.A. Costa, P.M. Pereira, R.P. Ribeiro and J. Gama, A Survey on Data-Driven Predictive Maintenance for the Railway Industry, *Sensors* **21**(17) (2021), 5739.
[20] D. Fernández-Francos, D. Martínez-Rego, O. Fontenla-Romero and A. Alonso-Betanzos, Automatic bearing fault diagnosis based on one-class ν-SVM, *Computers & Industrial Engineering* **64**(1) (2013), 357–365.
[21] D. Martínez-Rego, O. Fontenla-Romero, A. Alonso-Betanzos and J.C. Principe, Fault detection via recurrence time statistics and one-class classification, *Pattern Recognition Letters* **84** (2016), 8–14.
[22] E. Fumeo, L. Oneto and D. Anguita, Condition based maintenance in railway transportation systems based on big data streaming analysis, *Procedia Computer Science* **53** (2015), 437–446.
[23] N. Davari, B. Veloso, R.P. Ribeiro, P.M. Pereira and J. Gama, Predictive maintenance based on anomaly detection using deep learning for air production unit in the railway industry, in: *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, 2021, pp. 1–10.
[24] T. Kohonen, Self-organized formation of topologically correct feature maps, *Biological cybernetics* **43**(1) (1982), 59–69.
[25] R. Wetzig, A. Gulenko and F. Schmidt, Unsupervised Anomaly Alerting for IoT-Gateway Monitoring using Adaptive Thresholds and Half-Space Trees, in: *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, IEEE, 2019, pp. 161–168.
[26] J. Montiel, J. Read, A. Bifet and T. Abdessalem, Scikit-multiflow: A multi-output streaming framework, *The Journal of Machine Learning Research* **19**(1) (2018), 2915–2914.
[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., Scikit-learn: Machine learning in Python, *the Journal of machine Learning research* **12** (2011), 2825–2830.
[28] E. Manzoor, H. Lamba and L. Akoglu, xstream: Outlier detection in feature-evolving data streams, in: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1963–1972.
[29] Z. Ding and M. Fei, An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window, *IFAC Proceedings Volumes* **46**(20) (2013), 12–17.
[30] F. Angiulli and F. Fassetti, Detecting distance-based outliers in streams of data, in: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 811–820.
[31] V. Chandola, Anomaly Detection: A Survey, *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications* **41**(3) (2009), 71–97. ISBN 9780123985378. doi:10.1016/B978-0-12-398537-8.00004-3.
[32] J. Meira, R. Andrade, I. Praça, J. Carneiro, V. Bolón-Canedo, A. Alonso-Betanzos and G. Marreiros, Performance evaluation of unsupervised techniques in cyber-attack anomaly detection, *Journal of Ambient Intelligence and Humanized Computing* **11**(11) (2020), 4477–4489.
[33] S.S. Khan and A. Ahmad, Relationship between variants of one-class nearest neighbors and creating their accurate ensembles, *IEEE Transactions on Knowledge and Data Engineering* **30**(9) (2018), 1796–1809.
[34] J. Moubray, *Reliability-centered Maintenance*, Industrial Press, 2001. ISBN 9780831131463. https://books.google.pt/books?id=bNCVF0B7vpIC.
[35] E. Fumeo, L. Oneto and D. Anguita, Condition Based Maintenance in Railway Transportation Systems Based on Big Data Streaming Analysis, *Procedia Computer Science* **53** (2015), 437–446, INNS Conference on Big Data 2015 Program San Francisco, CA, USA 8-10 August 2015. doi:https://doi.org/10.1016/j.procs.2015.07.321. http://www.sciencedirect.com/science/article/pii/S1877050915018244.
[36] O.O. Aremu, A.S. Palau, A.K. Parlikad, D. Hyland-Wood and P.R. McAree, Structuring Data for Intelligent Predictive Maintenance in Asset Management, *IFAC-PapersOnLine* **51**(11) (2018), 514–519. doi:10.1016/j.ifacol.2018.08.370.
[37] P.C. Lopes Gerum, A. Altay and M. Baykal-Gürsoy, Data-driven predictive maintenance scheduling policies for railways, *Transp. Res. Part C Emerg. Technol.* **107**(October 2018) (2019), 137–154. https://doi.org/10.1016/j.trc.2019.07.020.
[38] Z. Allah Bukhsh, A. Saeed, I. Stipanovic and A.G. Doree, Predictive maintenance using tree-based classification techniques: A case of railway switches, *Transp. Res. Part C Emerg. Technol.* **101**(February) (2019), 35–54. doi:10.1016/j.trc.2019.02.001.

[39] M. De Benedetti, F. Leonardi, F. Messina, C. Santoro and A. Vasilakos, Anomaly detection and predictive maintenance for photovoltaic systems, *Neurocomputing* **310** (2018), 59–68. doi:10.1016/j.neucom.2018.05.017.
[40] Q. Wang, S. Bu and Z. He, Achieving predictive and proactive maintenance for high-speed railway power equipment with LSTM-RNN, *IEEE Transactions on Industrial Informatics* **16**(10) (2020), 6509–6517.
[41] D. Morandi and S. Jüngling, Anomaly Detection in Railway Infrastructure., in: *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 2021.
[42] C. Yang, Y. Sun, C. Ladubec and Y. Liu, Developing Machine Learning-Based Models for Railway Inspection, *Applied Sciences* **11**(1) (2021), 13.