Mach Learn (2016) 105:127–153
DOI 10.1007/s10994-016-5584-6

Sequential anomalies: a study in the Railway Industry

Rita P. Ribeiro[1,2] · **Pedro Pereira**[1,3] · **João Gama**[1,3]

Received: 14 April 2015 / Accepted: 5 July 2016 / Published online: 22 July 2016

Abstract Concerned with predicting equipment failures, predictive maintenance has a high impact both at a technical and at a financial level. Most modern equipments have logging systems that allow us to collect a diversity of data regarding their operation and health. Using data mining models for anomaly and novelty detection enables us to explore those datasets, building predictive systems that can detect and issue an alert when a failure starts evolving, avoiding the unknown development up to breakdown. In the present case, we use a failure detection system to predict train door breakdowns before they happen using data from their logging system. We use sensor data from pneumatic valves that control the open and close cycles of a door. Still, the failure of a cycle does not necessarily indicates a breakdown. A cycle might fail due to user interaction. The goal of this study is to detect structural failures in the automatic train door system, not when there is a cycle failure, but when there are sequences of cycle failures. We study three methods for such structural failure detection: outlier detection, anomaly detection and novelty detection, using different windowing strategies. We propose a two-stage approach, where the output of a point-anomaly algorithm is post-processed by a low-pass filter to obtain a subsequence-anomaly detection. The main result of the two-level architecture is a strong impact in the false alarm rate.

Editors: Saso Dzeroski, Dragi Kocev, and Pance Panov.

✉ Rita P. Ribeiro
rpribeiro@dcc.fc.up.pt

Pedro Pereira
pm.pereira.mail@gmail.com

João Gama
jgama@fep.up.pt

1 LIAAD-INESC TEC, University of Porto, Porto, Portugal

2 Faculty of Sciences, University of Porto, Porto, Portugal

3 Faculty of Economics, University of Porto, Porto, Portugal

Keywords Predicting failures · Subsequence anomaly detection · Subsequence novelty detection

1 Introduction

Predicting the future is an activity that has always captured the interest of humanity. As the Greek poet C. P. Cavalfy said: *"Ordinary mortals know what is happening now, the gods know what the future holds because They alone are totally enlightened. Wise men are aware of the future things just about to happen"*. The ability to predict and anticipate what is about to happen can make significant changes in how a business is run. This paper presents a data driven study on the train door early failure detection problem. The goal is to detect, in earlier stages, problems in the train doors automatic opening systems.

A rail vehicle is a highly complex piece of equipment, consisting of a variety of integrated subsystems, assembled to provide public or freight transport. Train passenger doors have a key role in such a transport system, allowing entering or exiting the vehicle at the right moment and ensuring, for the remainder of the trip, the maximum tightness, thermal and acoustic isolation. In addition, modern train doors have safety features, preventing customers from leaving the train while in motion or not stopped at a suitable location for passengers exit. Historically, doors were local and manually operated. The challenges posed by the need to reduce on board human resources, the growth in safety requirements and the advantages associated to a faster operation led to the sophistication of this equipment. Indeed, nowadays doors are a highly complex system, comprising electronic control circuits and pneumatic or electric drive systems, which in many cases reach opening and closing times of less than 2 s, and security mechanisms such as anti-pinch or force limiters. The growth in complexity of these functionalities poses additional problems in terms of reliability and maintenance. In fact, if in the past it was enough to lubricate hinges and adjust door alignments, today each door consists of many subsystems such as pneumatic valves, sensors, micro switches, call buttons and others, which greatly contribute to a huge growth in failure opportunity. In the case of train doors, its failure often causes relevant damages to the operation, not only at the service level, but also on the costs of operating the system, such as delays, trip cancellation or operational inefficiencies.

Given the significant impacts of door failures, much has been done to decrease their occurrence. Attention has been paid to areas spreading from the project phase, concerning design simplification or critical device redundancy, through reinforced preventive maintenance, including, for example, increased equipment replacement rate, to the introduction of new maintenance management methodologies, where Conditional Maintenance (Nowlan and Heap 1978) and Predictive Maintenance (Duyar 2011) stand out as the most usual trends.

Taking into account all this background, the emergence of data mining techniques seems to represent a line of action with great potential to minimize some of the problems the rail industry is facing. Indeed, the prospect that the intensive use of technology can make a more secure, coordinated and efficient transport system led the European Union itself to issue a policy, the European Directive 2010/40/EU (European Parliament and Council of the European Union of 7 July 2010 2010), on Intelligent Transport Systems (ITS). Bearing in mind all that is stated above, there is no doubt that the ability to have a train that could warn us in advance whenever a door failure is about to happen, would be an advantage that clearly contributes to a customer service level improvement, as well as to a more efficient operation and maintenance.

The goal of this paper is to develop a system that timely signals an alarm when a sequence of door operations indicates a deterioration of the system. We must point out that we are not interested in signalling alarms when a single operation is abnormal. This is not an indication of a problem in the train opening system but, most probably, the interference of a passenger.

Failure detection is a relevant machine learning task. In this work, we present a real case study where a failure is signalled in a context of consecutive anomalies. For the task of detecting anomalies, we use the most common algorithms: outlier detectors, anomaly detectors, and novelty detectors. Still, the use of such techniques alone is prone to false alarms, mainly because they assume *i.i.d.* observations. They do not deal with sequential nor temporal information. In this study, we propose a two-step approach for earlier failure detection. The first step consists of an abnormal point detector, whose output is filtered by a low-pass filter. The role of the low-pass filter is to detect *sequences* of point anomalies. We observe that the use of a low-pass filter to process the output of the abnormal point detectors leads to a strong reduction in the false alarm rate. This is the main contribution of this work: taking into consideration the temporal dimension for the outlier occurrence and, thus, providing an early failure detection. From that analysis, we can detect and issue an alert when a failure starts evolving, avoiding the unknown development up to breakdown.

The paper is organized as follows. In this section, we have explained the problem and the motivation. In Sect. 2, we discuss the anomaly detection and novelty detection techniques related to our target problem. In Sect. 3, we present the application problem and, in Sect. 4, we show the results obtained using different techniques. We present the lessons learned in Sect. 5.

2 Related work

The goal of predictive maintenance is to prevent unexpected equipment failures by convenient scheduling of corrective maintenance. Two relevant tasks for predictive maintenance are failure detection and failure prediction. For the purpose of early failure detection, techniques from novelty, anomaly and outlier detection have been used (Basseville 1988; Basseville and Nikiforov 1993; Patton et al. 2010; Papadimitropoulos et al. 2007; Katipamula and Brambley 2005; Yilboga et al. 2010).

Novelty, anomaly and outlier detection are loosely related terms. While the terms anomaly and outlier give the idea of an undesired pattern, the term novelty indicates an emergent or a new concept that corresponds to a possible state of the process under observation. According to Chandola et al. (2009), anomaly detection refers to the task of finding patterns in data that do not correspond to normal system behavior. According to the same author, novelty detection also aims to detect unobserved patterns (emergent, novel) in data. However, this term is distinguished from anomaly detection, since the novel patterns are typically assumed as an admissible state. In one-class-classification (Tax 2001), a special case of novelty detection, we train a decision model using only examples from the normal behavior of a system. The trained model is able to identify examples that do not correspond to normal system behavior.

Chandola et al. (2009) focus specifically on failure detection and provide a framework for the characterization of these problems, as well as an identification of the best methods for different applications, from intrusion detection to text mining problems. The authors also refer to the case of anomaly detection in engineering applications, presenting some specific fit-to-purpose techniques and applications. In engineering applications, one can consider that an anomaly is an outlier. In fact, one of the most globally accepted outlier definitions (Hawkins 1980) states that an outlier is a data object that deviates significantly from the rest of the objects, as if it were generated by a different mechanism. Thus, the assumption that an anomaly must be an outlier seems reasonable. In this context, one way to tackle anomaly detection problems can be by using outlier detection techniques. A recent and unified view of outlier detection methods appears in Schubert et al. (2014). Here, the authors present a comparative analysis of how the different methods model and find the outliers and, the assumptions they, implicitly or explicitly, rely on.

Along another dimension, data mining techniques can be divided into three main groups, depending on the existence of labelled instances: (1) unsupervised; (2) semi-supervised; (3) supervised. In the literature, all three approaches are used in data mining techniques relevant to our problem: outlier detection, anomaly detection, and novelty detection (Zhang et al. 2006; Han et al. 2011).

While supervised techniques usually exhibit better performance, unsupervised or semi-supervised techniques are often preferred due to the costs of labelling examples. This is an important aspect for our application where the costs of labelling examples are high, and expert advice is required.

In the following subsections, we briefly describe these three different approaches and their application to our target problem.

2.1 Unsupervised outlier detection

In the machine learning field, several techniques for outlier detection have appeared in the area of unsupervised learning. In effect, due to the lack of labelled and known instances of outliers, this constitutes a wide area of research concerning outlier detection. These techniques do not rely on previous information, they only assume that, for some similarity measure, outliers will appear isolated or in very small groups.

According to Han et al. (2011), unsupervised outlier detection methods can be grouped into statistical methods, clustering methods, distance-based and density-based methods. The choice of the appropriate method relies on several factors, such as the number of dimensions of the data, data type, sample size, algorithm efficiency, and, ultimately on the user understanding of the problem.

Whenever the goal is to identify univariate outliers, the statistical methods are among the simplest ones. Assuming a Gaussian distribution and learning the parameters from the data, parametric methods identify the points with low probability as outliers. One of the methods used to spot such outliers is the boxplot method, introduced by Tukey (1977). Based on the first quartile ($Q1$), the third quartile ($Q3$) and the interquartile range ($IQR = Q3 - Q1$) of the data, it determines that the interval $[Q1 - 1.5 * IQR, Q3 + 1.5 * IQR]$ contains 99.3% of data. Therefore, points outside that interval are considered as mild outliers, and points outside the interval $[Q1 - 3 * IQR, Q3 + 3 * IQR]$ are considered extreme outliers.

Regarding multivariate outliers, one of the most popular methods for unsupervised outlier detection is the Local Outlier Factor (LOF) density-based approach (Breunig et al. 2000). According to Aggarwal (2013), the classification of LOF as a density-based approach is a relaxation of the definition of density. In fact, even though the same author includes LOF in the density-based category, he considers it a relative distance-based approach with smoothing. LOF is a quantification of the *outlierness* of the data points, which is able to adjust for the variations in different densities. The local outlier factor is based on the concept of local density, where locality is given by k nearest neighbours, whose distance is used to estimate the density. By comparing the local density of an object to the local densities of its neighbours, one can identify regions of similar density. Therefore, outliers are points that have a substantially lower density than their neighbours.

2.2 Semi-supervised methods

In many practical anomaly detection applications it is only possible to have training sets consisting of elements from a single class, the normal one, as examples from the counter-class class are unavailable. In fact, in day-to-day problems in the maintenance field, frequently there are many examples belonging to the class Normal, e.g., corresponding to the normal system behavior. Examples corresponding to abnormal or anomalous system operation are difficult and costly to obtain. Moreover, it is very difficult to obtain labelled examples representing all the failure modes or types. As stated by Japkowicz et al. (1995), this is also the case in engineering anomaly detection problems. Anomaly detection in such a scenario, in which learning is made by only using samples from class Normal, are usually given the name of one-class classification or learning from positive-only examples (Tax 2001). There are several algorithms to address one-class classification, such as auto-associative neural networks, also known as Autoencoders (Japkowicz et al. 1995), One-Class SVMs (Han et al. 2011), and OCC (Hempstalk et al. 2008).

2.2.1 Autoencoders

An autoencoder or self-associator network (Japkowicz et al. 1995), is a neural network where the output layer reproduces the input layer. The simplest architecture of an autoencoder is a feedforward multilayer perceptron (MLP), with an input layer, an output layer and one hidden layer connecting them. The difference with the MLP is that in an autoencoder, the output layer has the same number of nodes as the input layer, and the hidden layer has fewer nodes. The autoencoder is trained to reconstruct at the output layer its own inputs $\mathbf{x}$. In the training phase, we use a backpropagation algorithm for the network to learn a function of the type $\hat{f}(\mathbf{x}) = \mathbf{x}$ or in other words, a function whose output reproduces the input. Assuming that training is done only with examples of the Normal class, the positive examples will be reconstructed with high precision, while for the Abnormal class examples there will be a noticeable deviation between the input and the output. Determining a boundary that discriminates between the reconstruction errors of positive and negative data is known as the *threshold determination* problem. The simplest approach, used for example in Japkowicz et al. (1995), consists of computing the largest reconstruction error of all the Normal training instances and then relaxing this bound by a certain percentage. New instances are subsequently classified by checking whether the reconstruction error of the new instance is higher than that of the relaxed boundary.

2.2.2 OCSVM

Tax and Duin (2004) proposed the method of Support Vector Data Description (SVDD) to implement One-Class SVM for novelty detection. This algorithm obtains a spherical boundary, in the feature space, around the data. The volume of this hypersphere is minimized, to minimize the effect of incorporating outliers in the solution. The resulting hypersphere is characterized by a centre c and a radius R. The radius is computed as the distance from the centre to any support vector on the boundary. The optimization problem consists of minimizing the volume of the hypersphere by minimizing R^2 and demanding that the sphere contains all training objects x_i.

2.2.3 OCC

The OCC algorithm (Hempstalk et al. 2008) combines density and class probability estimation for solving one-class classification problems. In this algorithm, only the examples from the Normal class are used for training. Firstly, OCC applies a density estimator to build a reference density for the target class, e.g. the Normal class. Then, this reference density is used to generate artificial data representing a second class, e.g., anomalies or outliers. Finally, a classifier is built with examples from both Normal and Abnormal classes. In Hempstalk et al. (2008), the authors suggest that the goal of the classifier should be accurate class probability estimation rather than minimization of classification error. Thus, they propose bagging of unpruned decision trees, an example of a suitable inductive approach, which has been shown to yield good class probability estimators (Provost and Domingos 2003).

2.3 Supervised methods

The impact of component failures on railway systems can significantly affect technical and operational reliability. Many advanced railway systems are equipped with monitoring and diagnostic tools to improve reliability and reduce maintenance expenditures. Data driven approaches for predictive maintenance is an active area of increasing interest (Angeli and Chatzinikolaou 2004; Papadimitropoulos et al. 2007; Yilboga et al. 2010). A main component of any predictive maintenance is time-to-failure prediction.

Several works address the problem of fault detection using supervised training of neural networks. In chemical engineering, Watanabe et al. (1989); Venkatasubramanian and Chan (1989); Ungar et al. (1990) were among the first to demonstrate the usefulness of neural networks for the problem of fault diagnosis. Later, Venkatasubramanian et al. (1990) presented a more detailed and thorough analysis of the learning, recall and generalization characteristics of neural networks for detecting and diagnosing process failures in steady-state processes. This work was later extended by Vaidyanathan and Venkatasubramanian (1992) to utilize dynamic process data. Watanabe et al. (1994) proposed a hierarchical neural network architecture (HANN) for the detection of multiple faults. According to the author, one of its advantages is that multiple faults could be detected in new data even if the network was trained with data representing single faults.

In a recent work, Sipos et al. (2014) use equipment logs from medical scanners to predict failures using multi-instance learning (Dietterich et al. 1997). In multi-instance learning the learner receives a set of bags which are labelled positive or negative. Each bag may contain multiple instances. Given bags obtained from different equipment and at different dates, the goal is to build a classifier that will label each unseen bag correctly, and the accuracy of models is measured at the bag level.

While multi-instance learning is interesting, the assumption that all instances in a bag are positive or negative cannot be applied in our case study.

2.4 Discussion

A recent survey on outlier detection for temporal data appears in Gupta et al. (2014). The authors refer to advances in sensor technology that lead to research on temporal outlier detection. Temporal outlier detection is defined as *analysis of anomalies in the behavior of* *data over time*. A relevant aspect is the distinction between *point outliers* and *subsequence outliers*. While the former refers to a single unusual data point in a temporal series, the latter refers to a set of consecutive unusual data points. The latter scenario is usually far more challenging than the former (Gupta et al. 2014). Most of the work in outlier detection focuses on point forecasts.

Our goal is to detect structural faults of an opening system from sensor readings without external feedback. Our hypothesis is that a structural fault corresponds to a consecutive set of outliers, e.g., *subsequence outliers*. As we will show in the experimental section, the use of a low-pass filter as a post-processor of the output of an unsupervised *point outlier* detector can identify such *subsequence outliers*.

3 The case study

The purpose of this paper is to develop a data mining system that issues an alert whenever an automatic door is predicted to suffer a failure. In this case study, we focus our attention on the behaviour of the pneumatic doors from one specific train. We used a database composed of data collected from September to December 2012 by the Northern Rail, UK company. The data is from sensors installed in the doors of a Class 156 train. A linear pneumatic actuator activates each door. The pneumatic actuators are equipped with one pressure transductor on both the inlet and outlet chamber, providing a pressure reading every 1/10th of a second whenever the door is commanded to move. The available data, representing operations from September to December 2012, consists of almost 500 thousand readings, corresponding to 4500 opening and closing cycles. We must note that current opening systems are equipped with sensors that react (invert the operation) when a passenger interferes. This fact, a feature of the system, triggers false alarms in fault detection systems that we need to avoid.

To accomplish this task, we have come up with a two-stage classification process. First, each cycle is classified as Normal or Abnormal, and afterwards we use a low-pass filter on the output to decide if there is evidence that a door breakdown is about to happen. For the cycle classification problem, we have experimented with two different learning approaches: unsupervised and semi-supervised.

3.1 Data transformation

Considering that our plan involved working with classification algorithms, we defined an attribute-value matrix, with each tuple representing a cycle, as our input data. For that purpose, we have transformed our time series dataset and created a new set of five variables, as described in detail below.

We started by calculating the total duration of each cycle using its maximum and minimum timestamps. Therefore we could determine the span of each of the five equal length time periods, which we call bins. With this information we could assign each pair of pressure readings, from inlet and outlet chambers, to a specific bin. Then, for each cycle and at each timestamp, we could compute the pressure difference between the inlet chamber and outlet chamber. The discretization process was finished by calculating the average pressure difference for each bin. At last, bearing in mind that the duration of each bin, and therefore the total cycle length, was vital information, we generated the five final variables, multiplying the bin average pressure by the cycle duration.

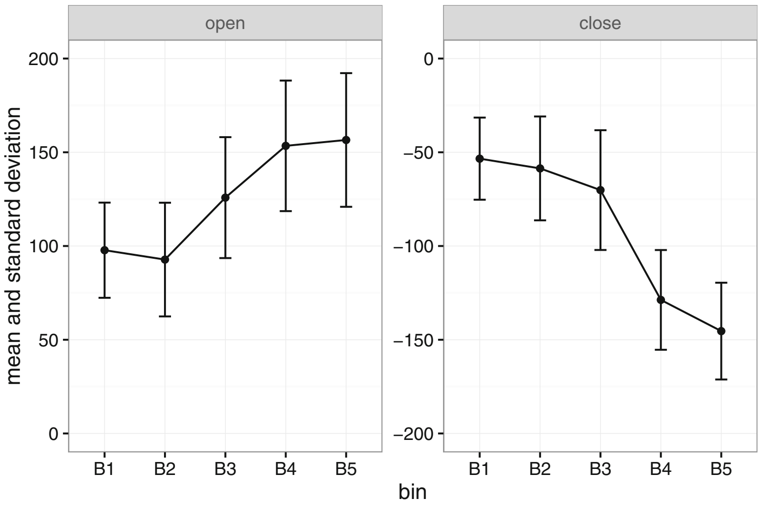

Fig. 1 Evolution of the average difference between the pressures in the inlet and outlet chambers for all opening and closing cycle door movements from September to December 2012

The data transformation process was only finished with the introduction of a new categorical variable, describing the type of movement. Considering that when doors are opening the pressure in the inlet chamber must be higher than in the outlet chamber (otherwise the door would not start its movement), we defined that for all cycles where the value of the first bin was greater than 0, door movement was set to Open, otherwise it was labelled as Close.

This process allowed us to rearrange the dataset, transforming 500 thousand pressure readings into 4630 door cycles, described by five variables. The new set of attributes was named B1 to B5, with B1 being the first bin, when the door has just started to move, and B5 representing the last bin, when the door movement cycle has finished.

Figure 1 shows the evolution of a cycle under the new set of variables for both opening and closing door movements, using the mean and standard deviation of each of these five bins.

Bearing in mind that the temporal information was an important aspect of the dataset, as showed in Fig. 2, daily averages were also calculated for each attribute across the entire period from September to December 2012. From the analysis of this figure, one can observe that, especially in closing movements, the attribute average suffered an important shift during September, in what could be concept evolution or the development of a failure.

3.2 Labelling

In order to label the dataset, two new attributes were introduced, one about the normality of the cycle itself, the cycle class considered as Normal or Abnormal, and another, the sequence class, with information on whether a particular sequence of cycles should be considered as Normal or Failure. As usual for these tasks, a domain expert was called in to classify each tuple in the data matrix. It is important to notice that a cycle can be labelled as Abnormal even though there is no associated failure. Such a case may arise from a door being blocked by a passenger, which must not be considered as a door failure. As for the sequence class label, door failure moments were identified from the Maintenance Reports. Failure time-windows

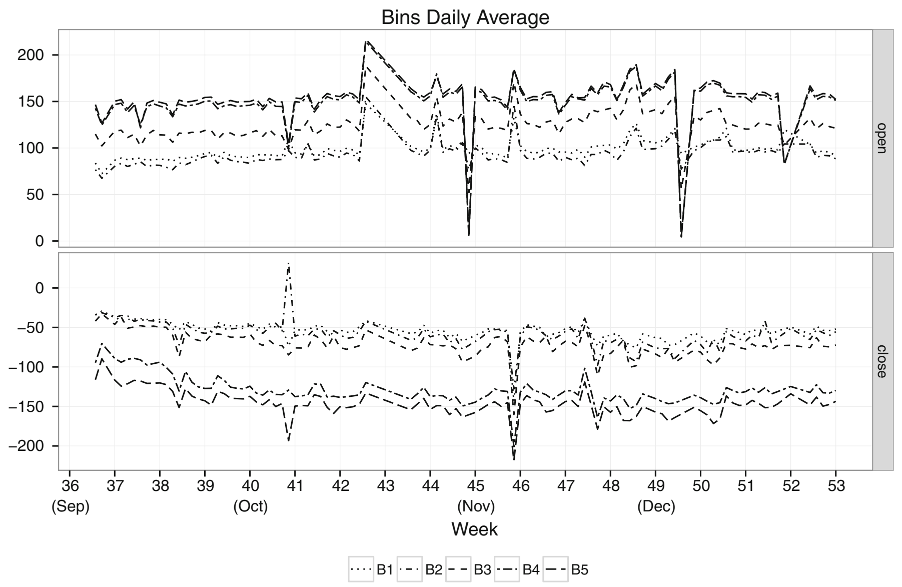

Fig. 2 Daily average of the difference between the pressures in the inlet and outlet chambers within each of the *five bins* (B1–B5) for all opening and closing cycle door movements from September to December 2012

Table 1 Identification of door Abnormal cycles by the domain expert

Month	Total cycles		Abnormal cycles		% Abnormal cycles	
	Open	Close	Open	Close	Open (%)	Close (%)
Sept.	578	568	19	46	3	8
Oct.	628	612	20	9	3	1
Nov.	643	634	54	24	8	4
Dec.	507	489	18	7	4	1
Total	2356	2303	111	86	5	4

Table 2 Identification of door failures by the domain expert

Door failure	Week	Date	Nr abnormal cycles	
			Open	Close
Failure 1	36	09/07/2012	7	2
Failure 2	48	11/28/2012	3	0
		11/29/2012	31	33
		11/30/2012	24	24
Failure 3	49	12/06/2012	33	0

were determined by encompassing the cycles that occurred before, and related to one specific failure.

In the end, after expert classification, there were 197 door cycles labeled as Abnormal, 4% of the total, and three failure events, two of them occurring in both the opening and closing door movements (cf. Tables 1 and 2).

3.3 Abnormal cycle detection

To address the abnormal cycle detection problem, we divided it into two smaller problems: one for the opening door movements; and the other for the closing door movements. Each dataset is composed of five attributes representing the average difference between pressures in inlet and outlet chambers per cycle within each of its five bins. For each problem, we have tested Abnormal cycle identification under two different approaches: (1) unsupervised learning; (2) semi-supervised learning.

In order to maintain the work as close as possible to a real scenario, we considered that we had expert classification only for the first 4 weeks, from which we trained our first model. The decision model is then used to predict the following week. From that point on, for all the subsequent weeks, the models are self-feedback trained, e.g., they learn from their own predictions.

Regarding the unsupervised learning approach, we have used a boxplot-based method (boxplotEns) as follows: we generate five boxplots, one for each attribute, on the training data. For each new observation, we check if the value of each attribute is considered to be an outlier by the respective boxplot. The observation is considered to be an Abnormal cycle if at least one of the five attributes values is signalled as an outlier. This approach corresponds to an ensemble of boxplots to deal with multivariate data.

Regarding the semi-supervised learning approach, we have used three different algorithms: Autoencoders (Japkowicz et al. 1995), OCSVM (Han et al. 2011) and OCC (Hempstalk et al. 2008). Each of these algorithms is trained with data from Normal cycles only. The Abnormal cycle detection made by the first two algorithms was solely based on their standard parameters. For the Autoencoder algorithm we have additionally defined the threshold for the rejection based on the reconstruction error obtained on the training dataset. For each observation of the training dataset, we calculate the mean squared error obtained by the reconstruction of all the five inputs. We then use the boxplot to obtain the error distribution of all the observations and set as threshold for rejection the value above which the error is considered to be an extreme outlier. This means that if the reconstruction error of a new observation is above this threshold then it is considered to be an Abnormal cycle, otherwise it is considered to be a Normal cycle.

Finally, to assess the performance of the abnormal cycle detection task, we used the approach suggested by Hempstalk et al. (2008). For each cycle identification we calculated two ratios: false alarm rate (FAR) and the impostor pass rate (IPR). The false alarm rate gives us the ratio of identified Abnormal cycles that are, in fact, Normal cycles. The impostor pass rate gives us the ratio of Abnormal cycles that are wrongly identified as Normal. These metrics are often used in outlier detection domains. Usually, a higher FAR results in a lower IPR and vice versa. Still, to better adapt these two metrics to our sequential data scenario and benefit early failure detection, we have used the reduced false alarm rate (rFAR) and the reduced impostor pass rate (rIPR), as follows. The rFAR reduces the number of false alarms with the cycles that should not be signaled as so, for appearing just before the correct identification of a failure. The rIPR reduces the number of impostor pass with the cycles that should not be signaled as so, for appearing just after the correct identification of a failure. According to this, the best models are the ones with minimal detection error, given by $Err = \sqrt{rFAR^2 + rIPR^2}$.

3.4 From point-anomaly detection to failure sequence detection

Cycle prediction uses point detection as follows: for each single opening or closing cycle predicts if it is anomaly or not. In fact, the utility of this work hinges on the ability to issue an alarm whenever a door is about to have a breakdown, not to distinguish between Normal and Abnormal cycles. To achieve this part of the process, we use a low-pass filter, as described below. The input for the filter is the output of the anomaly detection algorithm. It can be binary, 1 meaning Normal, 0 meaning Anomaly or numeric, representing the probability of normality.

3.4.1 Low-pass filter

A filter is a device that removes from a signal some unwanted component or feature (Shenoi 2005). Filtering is a known technique, commonly used as a pre-processing step, for smoothing data or removing noisy observations (Sluban et al. 2010).

The defining feature of filters is the complete or partial suppression of some aspect of the signal. Often, this means removing some frequencies and not others in order to suppress interfering signals and reduce background noise. Several filters can be designed to achieve specific goals considering the application. A low-pass filter is a filter that smooths abrupt changes in the signal, attenuating (reducing the amplitude of) signals with high variations. The simplest low-pass filter algorithm is detailed by the recursive equation: $y_i = y_{i-1} + \alpha * (x_i - y_{i-1})$, where y_i is the filter output for the original signal x_i for instant i and α is the smoothing parameter. The initial value of the filter is $y_0 = x_0$. The change from one filter output to the next is proportional to the difference between the previous output and the next input. This exponential smoothing property matches the exponential decay seen in continuous-time systems. As expected, as α decreases, the output samples respond more slowly to a change in the input samples: the system has more inertia.

For each anomaly detection algorithm and each problem, we have tuned the smoothing factor α of the low-pass filter with a specific parameterization. The threshold level was fixed to 0.5, without any tuning.

4 Experimental evaluation

In this section, we experimentally study the influence, in terms of anomaly detection, of the different components of our system. We try to answer the following research questions:

- What is the benefit of using a low-pass filter in sequential anomaly detection?
- What is the impact of system feedback versus expert feedback?
- What is the impact of past information usage in the detection system?

Given these research questions, we follow three evaluation scenarios. In the first scenario, we test the impact of the low-pass filter in the reduction of false alarms using four different anomaly detectors. The second scenario, evaluates two different forms of feedback: supervised feedback that requires expert information, versus unsupervised feedback based on the predictions of anomaly detectors. The third scenario studies performance measures obtained using different training windows: static model versus sliding and growing windows.

4.1 Experimental setup

We started by using the data supplied by the expert for the first 4 weeks, i.e., for the month of September, for training our decision models. In this period, we have registered the occurrence of Failure 1 (cf. Table 2) for both, open and close, movements. For tuning our decision models, we replicated these first 4 weeks of data, two more times, to achieve a 3 months length data set. Our goals regarding the off-line tuning process of each decision model are: (i) define the best learning strategy concerning the usage of past data; (ii) adjust some of the parameters used by the different learning algorithms; and, (iii) find a suitable smoothing factor α for the low-pass filter.

Regarding the first tuning goal, we have followed two main learning approaches to predict each week: *static* and *evolving* models. With the first approach, we learn a decision model once, and then use it to make predictions for all the subsequent weeks. With the second approach, we update the first learnt decision model as time evolves. This update is performed every new week, incorporating the information of that week from two different origins: expert feedback and model feedback. Thus, each decision model is trained according to one of the following four strategies.

– *swexpert*: an evolving approach that uses a sliding window, which starts with the expert information on the first four weeks and then, iteratively, slides by dropping the first week of the window and including the expert feedback on the next week.
– *swpred*: an evolving approach that uses a sliding window, which starts with the expert information on the first four weeks and then, iteratively, slides by dropping the first week of the window and including self-feedback, i.e., the model prediction on the next week.
– *gwpred*: an evolving approach that uses a growing window that starts with the expert information on the first four weeks and then, iteratively, grows by including self-feedback, i.e., the model prediction on the next week.
– *static*: a static approach that uses a window that includes the expert feedback on the first four weeks.

For the second tuning goal, we have experimented with the following four learning algorithms available in R (Core 2014):

– **boxplotEns** through an ensemble of boxplots, each one provided by the function `boxplot` varying the `coef` parameters, to determine which type of outliers (*mild* or *extreme*) should be identified as anomalies;
– **autoencoders** (Japkowicz et al. 1995) implemented with the R package `autoencoder` (Eugene 2015) with its default setting, but specifying `N.input=5`, `lambda=0.0002`, `beta=6`, `rho=0.01`, `epsilon=0.001` and varying the number of nodes in the hidden layer (`N.hidden`);
– **one-class svm (ocsvm)** (Han et al. 2011) available through the function `svm` of the R package `e1071` (Meyer et al. 2014) with its default setting, but specifying `type='one-classification'` and varying the `nu` parameter;
– **one-class classification (occ)** (Hempstalk et al. 2008) available through the R package `RWeka` (Hornik et al. 2009), which interfaces with Weka (Hall et al. 2009), with its default setting, but varying the target rejection rate (`trr`) parameter.

A summary of the parameters of each learning algorithm subject to tuning is presented in Table 6 in “Appendix”.

Finally, our purpose with the third step of the tuning process was to evaluate the impact of the low-pass filter on the reduction of false alarms triggered by the predictions made by the different learning strategies. Thus, for each learning algorithm, we set the initial value of the filter to a Normal cycle, i.e., $y_0 = 1$, and then adjust the α parameter for the open and close movements, separately. We varied the α parameter with values between 0.05 and 0.1, with step of 0.0005. We set the cut-off threshold for the filter at 0.5, meaning that if the filter value for cycle i is below 0.5, then it is considered to be an Abnormal cycle, otherwise it is considered to be a Normal cycle.

Table 7 in "Appendix" shows the best parameterization achieved in the tuning phase, for each training strategy and learning algorithm, by minimizing the detection error (Err). Overall, regarding the low-pass filter, it is possible to observe that close cycle movements, in comparison to open cycle movements, are more prone to false alarms and thus, need more inertia, i.e., lower values of α. This is in accordance with the scenario of passenger interference to stop the closing door movement so that they can enter the train.

The detailed results of applying these decision models to all the data are presented in the following sections.

4.2 Experimental results

4.2.1 Using a low-pass filter in sequential anomaly detection

In this section, we argue that the information about the open-close cycles is not enough to detect structural failures in automatic train door systems. As we already indicated, the ability of a passenger to interrupt a cycle is a feature of the system and not a failure. Therefore, we consider that analysing individual cycles is not enough to infer a persistent failure. Persistent

Table 3 The reduction of the number of false alarms, for each door movement, using a sliding window with expert feedback and different models, due to the use of the low-pass filter

Train window	Learning algorithm	Number of false alarms					
		Open			Close		
		Before filter	After filter	Reduction	Before filter	After filter	Reduction
swexpert	boxplotEns	146	36	75 %	115	10	91 %
	autoenc	191	70	63 %	98	34	65 %
	ocsvm	207	61	71 %	377	198	47 %
	occ	101	16	84 %	957	963	−1 %
swpred	boxplotEns	146	36	75 %	115	10	91%
	autoenc	34	6	82 %	30	0	100 %
	ocsvm	292	99	66 %	325	216	34 %
	occ	220	39	82 %	239	16	93 %
gwpred	boxplotEns	185	55	70 %	97	27	72%
	autoenc	36	4	89 %	28	0	100 %
	ocsvm	200	75	62 %	387	276	29 %
	occ	147	23	84 %	1246	1240	0 %
static	boxplotEns	632	362	43 %	280	217	22%
	autoenc	572	291	49 %	558	397	29 %
	ocsvm	530	353	33 %	1081	1058	2 %
	occ	494	315	36 %	1215	1251	−3 %

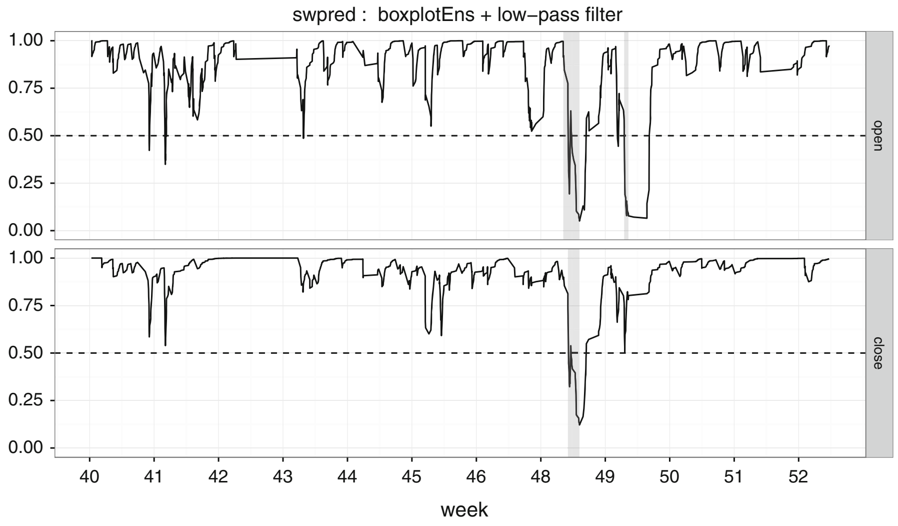

Fig. 3 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using sliding window with self-feedback and boxplotEns for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

failure detection requires the analysis of sequences of cycles. We argue that sequences of anomalous cycles are a strong indication of a structural failure.

Table 3 reports, for the four anomaly detection algorithms, the number of false alarms considering the maintenance reports provided by the train company. We observe a strong reduction of the number of false alarms *after* the low-pass filter. The low-pass filter is quite effective with all models used. For the evolving models, the average reduction is around 75% for open cycles, and around 60% for close cycles.

4.2.2 Using expert information versus models predictions

Training a failure detector model requires information about Normal and Abnormal cycles. This *supervision* can be obtained either by consulting an expert, or by using predictions from the detector models. For real-time prediction, supervised models using expert feedback are not appropriate, in the sense that have high costs. In this section, we compare the performance of the detector models using expert feedback and self-feedback, i.e., feedback obtained from model predictions.

Figures 3, 4, 5, 6 plot, over time, the effect of the low pass filter with a sliding window that uses previous predictions of the models instead of expert information. In these figures, the **x** axis represents time, and the **y** axis represents the probability of normal status, varying from 1 (normal state) to 0 (failure). For each plot, there are two panels: the top panel refers to open-cycles, and the bottom panel to close-cycles. The gray bars represent the anomaly state of doors, as reported by the maintenance office.

From the analysis of the results, we observe that overall, the performance of these self-feedback models improves in comparison with expert-feedback on anomalous cycles (see the plots in “Appendix”). Still, we must point-out that the boxplotEns algorithm is fully unsupervised, as it does not require any information about the *normality* state of the cycles.

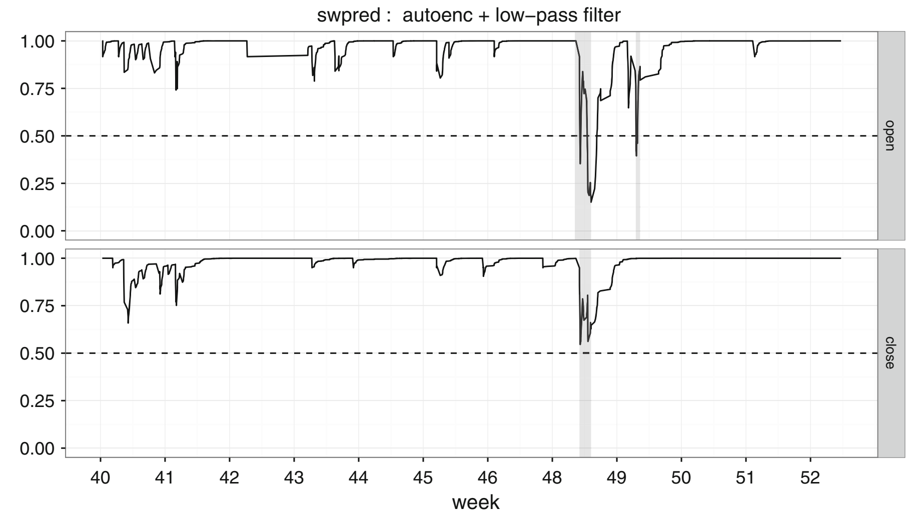

Fig. 4 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using sliding window with self-feedback and autoencoder for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

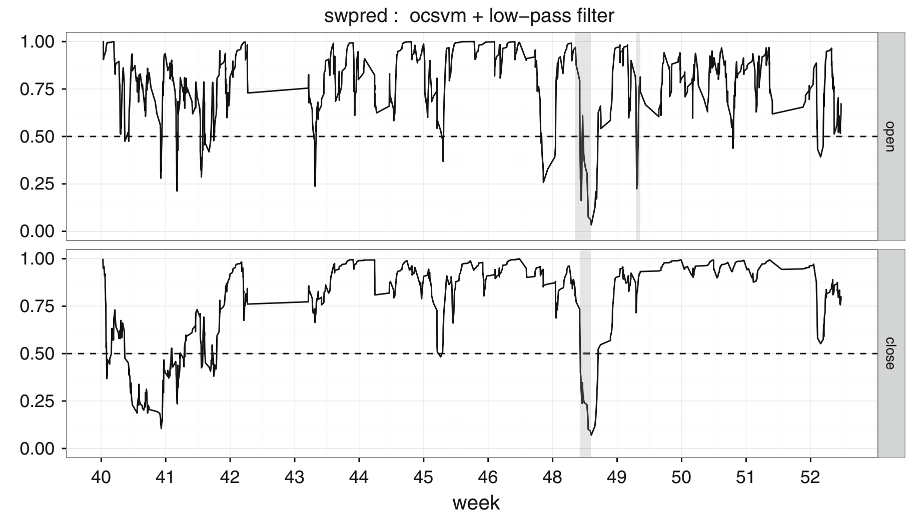

Fig. 5 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using sliding window with self-feedback and ocsvm for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

Thus, regardless of type of feedback, it achieves the same performance. In effect, both boxplotEns and autoencoder with low-pass filter show the best behaviour on failure detection (cf. Figs. 3 and 4). They are able to detect both failures with zero or a reduced number of false alarms. Even though the autoencoder is not able to detect the first failure in the close movement, it is able to detect it through the open movement. The other models, as shown

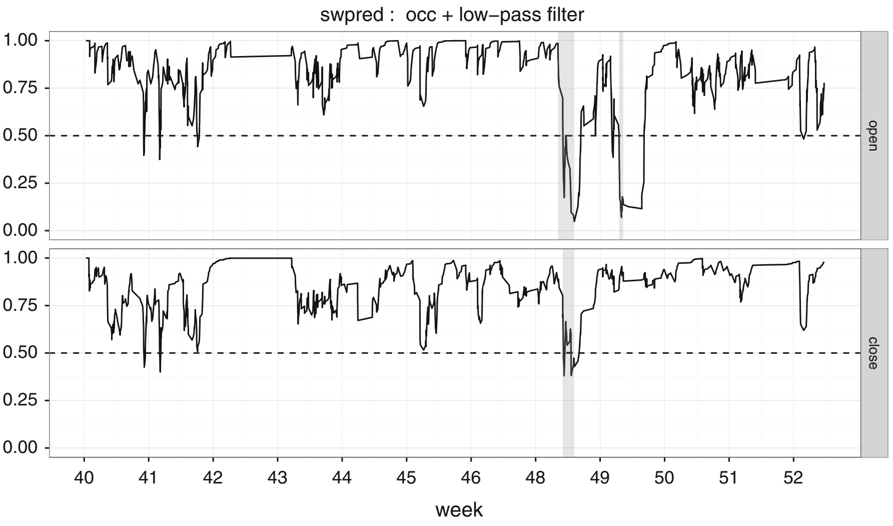

Fig. 6 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using sliding window with self-feedback and occ for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

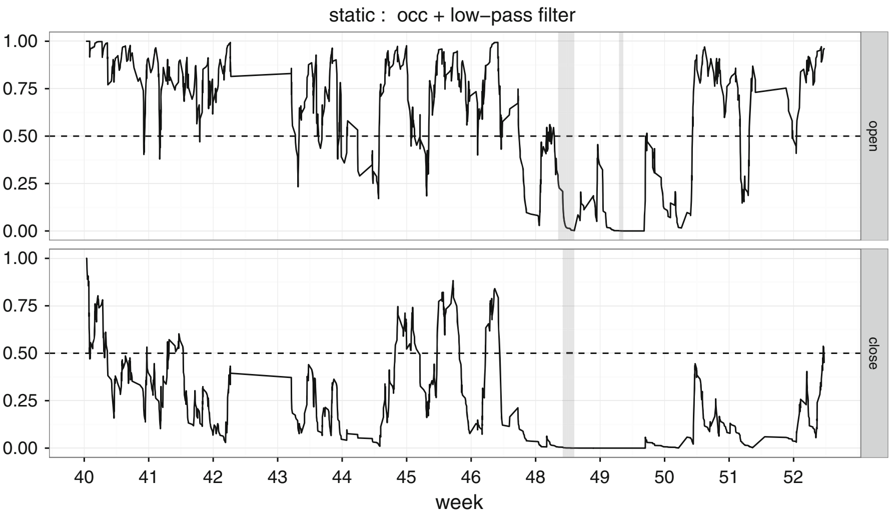

Fig. 7 The performance of a static `occ` model with a low-pass filter degrades over time. The *graphs* depict the probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using a static window with expert feedback and `occ` for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

in Figs. 5 and 6, yield a higher number of false alarms. Still, all these models are able to detect the two failures at their beginning, which is an important aspect for the predictive maintenance.

Table 4 Reduced False Alarm Rate (rFAR), Reduced Impostor Pass Rate (rIPR) and Detection Error (Err) using different training windows with different learning algorithms without the low-pass filter

Train window	Learning algorithm	Open			Close		
		rFAR	rIPR	Err	rFAR	rIPR	Err
swexpert	boxplotEns	0.087	0.000	0.087	0.069	0.000	0.069
	autoenc	0.113	0.000	0.113	0.058	0.000	0.058
	ocsvm	0.123	0.000	0.123	0.225	0.000	0.225
	occ	0.060	0.022	0.064	0.570	0.000	0.570
swpred	boxplotEns	0.087	0.000	0.087	0.069	0.000	0.069
	autoenc	0.020	0.033	0.039	0.018	0.000	0.018
	ocsvm	0.173	0.022	0.174	0.194	0.000	0.194
	occ	0.130	0.000	0.130	0.142	0.000	0.142
gwpred	boxplotEns	0.110	0.000	0.110	0.058	0.000	0.058
	autoenc	0.021	0.033	0.039	0.017	0.000	0.017
	ocsvm	0.119	0.000	0.119	0.231	0.000	0.231
	occ	0.087	0.022	0.090	0.743	0.000	0.743
static	boxplotEns	0.375	0.000	0.375	0.167	0.000	0.167
	autoenc	0.339	0.000	0.339	0.333	0.000	0.333
	ocsvm	0.314	0.000	0.314	0.644	0.000	0.644
	occ	0.293	0.000	0.293	0.724	0.000	0.724

4.2.3 Using different training windows

A commonly used strategy in data mining consists of training a static model from a sample of historical data, using this model to make predictions for future examples. This strategy is problematic in our case. The performance of static models degrades over time. For illustration purposes, we show in Fig. 7 the predictions made by a static occ model over time. As can be observed, the model is unable to re-adapt to the evolution of operating regimes of open and close movements (cf. Fig. 2). Thus, it frequently signals Normal functioning cycles as Abnormal ones. All the other models exhibit similar performance when using a static approach—see "Appendix".

In Tables 4 and 5, we present the estimated performance of the models regarding Abnormal cycle detection, i.e., the rFAR, rIPR and Err obtained by the models before and after applying the low-pass filter, respectively. We experimented with different models and different model updating strategies. The results improved by the application of the low-pass filter are signaled in bold in Table 5 From the analysis of the results, the following conclusions can be drawn: (i) evolving models produce fewer false alarms than static models and thus, are better from our application perspective; (ii) with the exception of the unsupervised multivariate boxplotEns approach, which is trained with all the data, models that are trained with Normal cycle data achieve better results using a sliding window strategy than a growing-window strategy; the reason for this might have to do with the fact that sliding windows capture better events, such as door failures, that have short-term evolution; (iii) the application of low-pass filter leads to a smaller number of false alarms, but also to a higher number of passing impostors, i.e. Abnormal cycles that are considered Normal; however, overall, it leads to a smaller detection error; (iv) using a sliding window with expert feedback gives us the lowest rIPR, still, this is also the most infeasible approach; (v) considering the four learning algorithms, sliding-

Table 5 Reduced False Alarm Rate (rFAR), Reduced Impostor Pass Rate (rIPR) and Detection Error (Err) using different training windows with different learning algorithms with the low-pass filter

Train window	Learn. alg.+ low-pass filter	Open			Close		
		rFAR	rIPR	Err	rFAR	rIPR	Err
swexpert	boxplotEns	**0.021**	0.121	0.123	**0.006**	0.175	0.176
	autoenc	**0.041**	0.088	**0.097**	**0.020**	0.211	0.211
	ocsvm	**0.036**	0.077	**0.085**	**0.118**	0.053	**0.129**
	occ	0.009	0.165	0.165	0.574	0.000	0.574
swpred	boxplotEns	**0.021**	0.121	0.123	**0.006**	0.175	0.176
	autoenc	**0.004**	0.176	0.176	**0.000**	1.000	1.000
	ocsvm	**0.059**	0.132	**0.144**	**0.129**	0.140	**0.190**
	occ	**0.023**	0.088	**0.091**	**0.010**	0.193	0.193
gwpred	boxplotEns	**0.033**	0.110	0.115	**0.016**	0.211	0.211
	autoenc	**0.002**	0.176	0.176	**0.000**	1.000	1.000
	ocsvm	**0.044**	0.099	**0.108**	**0.164**	0.105	**0.195**
	occ	**0.014**	0.462	0.462	**0.739**	0.000	**0.739**
static	boxplotEns	**0.215**	0.000	**0.215**	**0.129**	0.088	**0.156**
	autoenc	**0.172**	0.000	**0.172**	**0.237**	0.000	**0.237**
	ocsvm	**0.209**	0.000	**0.209**	**0.631**	0.000	**0.631**
	occ	**0.187**	0.000	**0.187**	0.746	0.000	0.746

window with self-feedback gives the best performance on rFAR and rIPR; (vi) overall, we can say that the boxplotEns and autoencoders with a sliding window with self-feedback give us the best performance for failure detection with reduced number of false alarms, meeting the goals of our case study.

5 Conclusions

The main goal of predictive maintenance is to prevent unexpected equipment failures by recommending maintenance as early as possible. In this paper, we study a data driven predictive maintenance system that issues an alarm whenever it is predicted that an automatic train door is about to have a failure. According to our study, there are three key aspects for a successful sequencial anomaly detection: (i) two-stage algorithm, (ii) sliding window evolving models and (iii) self-training models.

We have presented a failure detection system based on a two-stage algorithm: (1) event classification based on anomaly detection and (2) sequence analysis applying a low-pass filter. This two-stage approach can be seen as a generic method for subsequence outlier detection: the first stage detects point anomalies that are merged, in the second stage, by the low-pass filter for subsequence outlier detection. Standard approaches in this field do not take into account that we may need to predict a breakdown/failure and not simply to distinguish between Normal and Abnormal cycles.

The two other key characteristics for successful sequential anomaly detection are evolving models using sliding windows, and self train of semi-supervised models. The full system is autonomous (does not required human intervention) and it can work online, which are crucial factors for real-time predictive maintenance.

More specifically, evolving models produce fewer false alarms than static models, with the sliding window strategy working better than the growing-window strategy. While expert feedback leads to the best performance, it is the least feasible in practice: sliding-window with self-feedback gives the best performance, especially in conjunction with boxplots and autoencoders for anomaly detection. The use of a low-pass filter leads to a smaller number of false alarms and overall smaller failure detection error.

Thus, we have shown that, at least for this specific problem, with a small investment in sensors, data logging and data mining techniques, we are able to decrease maintenance costs and increase reliability for the studied system.

A failure detection project can be improved by future developments. Lead time or reliability can be improved with practical implications for those who apply this kind of methodology in maintenance management. Moreover, the approach developed in the paper can be applied in several different environments, from pneumatic or electric doors, no matter if they are in a bus, a train, or even in a shopping mall. It can also be applied to heavy duty industrial equipment, such as a hydraulic press. The workflow we have used can be fully automated. It can work in a stand-alone environment, after initial configuration for the specific problem at hand is performed.

We plan to bring the application presented here to commercial use. It will be embedded in a maintenance management and remote diagnoses software product that is already on the market. At the time of writing this, it is in the final stage of software development and once this milestone is reached a test program will be applied to validate the results.

Acknowledgments We thank the editors and the reviewers for their valuable comments and suggestions, which have helped to improve the quality of the paper. This work was supported by the research project "TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020", financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF) and by European Commission through the project MAESTRA (Grant number ICT-2013-612944).

Appendix: Results

The following tables show the tuned parameters and values for each learning algorithm (cf. Table 6) and the best parameterization obtained for each decision model in the tuning phase (cf. Table 7).

The following figures (cf. Figs. 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18) show over time, the effect of the low pass filter on different learning strategies (sliding window with expert feedback, growing window with self feedback and static window) and different learning algorithms (boxplotEns, ocsvm, occ and autoenc). In these figures, the **x** axis represents time, and the **y** axis represents the probability of normal status, varying from 1 (normal state) to 0 (failure). For each plot, there are two panels: the top panel refers to open-cycles, and the

Table 6 Tuned parameters for each learning algorithm

Learning algorithm	Tuned parameter	Values
boxplotEns	`coef`	`{1.5, 3}`
autoenc	`hidden`	`{1, 2, 3, 4}`
ocsvm	`nu`	`{0.5, 0.1, 0.05, 0.01, 0.005, 0.001}`
occ	`trr`	`{0.5, 0.1, 0.05, 0.01, 0.005, 0.001}`

Table 7 The best parameterization for each training strategy and learning algorithm

Train window	Learning algorithm	Open		Close	
		Algorithm parameter	Low-pass filter parameter (α)	Algorithm parameter	Low-pass filter parameter (α)
swexpert	boxplotEns	`coef = 1.5`	0.0830	`coef = 1.5`	0.0500
	autoenc	`hidden = 1`	0.0830	`hidden = 1`	0.0500
	ocsvm	`nu = 0.010`	0.0945	`nu = 0.1`	0.1000
	occ	`trr = 0.01`	0.0830	`trr = 0.5`	0.0955
swpred	boxplotEns	`coef = 1.5`	0.0830	`coef = 1.5`	0.0500
	autoenc	`hidden = 1`	0.0830	`hidden = 1`	0.0500
	ocsvm	`nu = 0.1`	0.0945	`nu = 0.1`	0.0500
	occ	`trr = 0.1`	0.0830	`trr = 0.1`	0.0500
gwpred	boxplotEns	`coef = 1.5`	0.0830	`coef = 1.5`	0.0500
	autoenc	`hidden = 1`	0.0830	`hidden = 1`	0.0500
	ocsvm	`nu = 0.05`	0.0945	`nu = 0.1`	0.0500
	occ	`trr = 0.05`	0.0830	`trr = 0.5`	0.0965
static	boxplotEns	`coef = 1.5`	0.0830	`coef = 1.5`	0.0500
	autoenc	`hidden = 1`	0.0830	`hidden = 1`	0.0500
	ocsvm	`nu = 0.010`	0.0945	`nu = 0.1`	0.1000
	occ	`trr = 0.050`	0.0830	`trr = 0.5`	0.0570

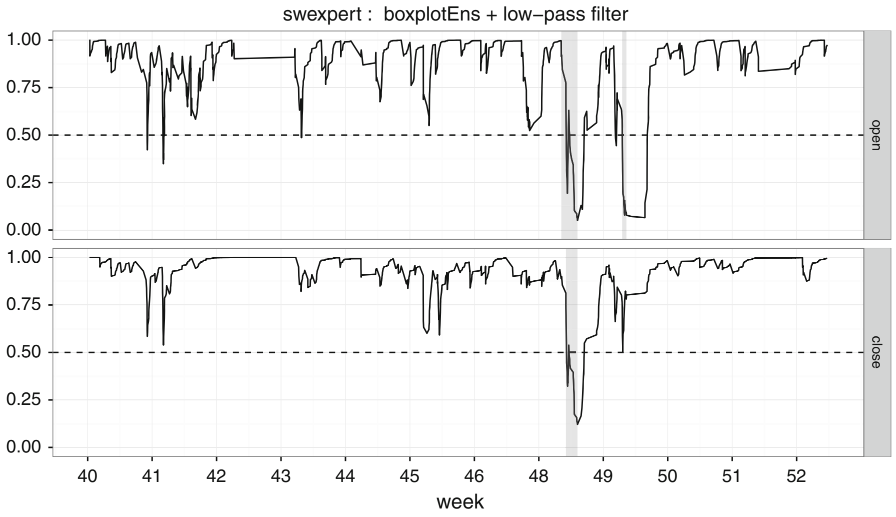

Fig. 8 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using sliding window with expert feedback and boxplotEns for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

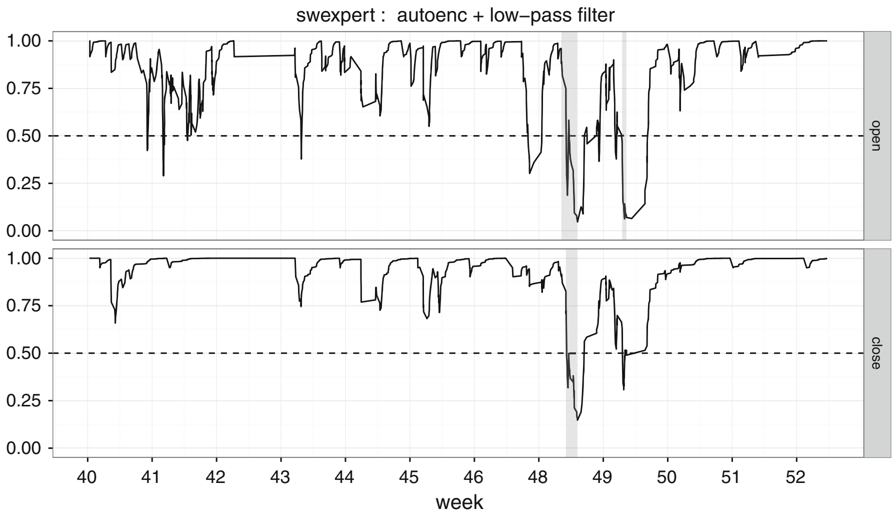

Fig. 9 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using sliding window with expert feedback and autoencoder for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by vertical grey bars

bottom panel to close-cycles. The gray bars represent the anomaly state of doors, as reported by the maintenance office.

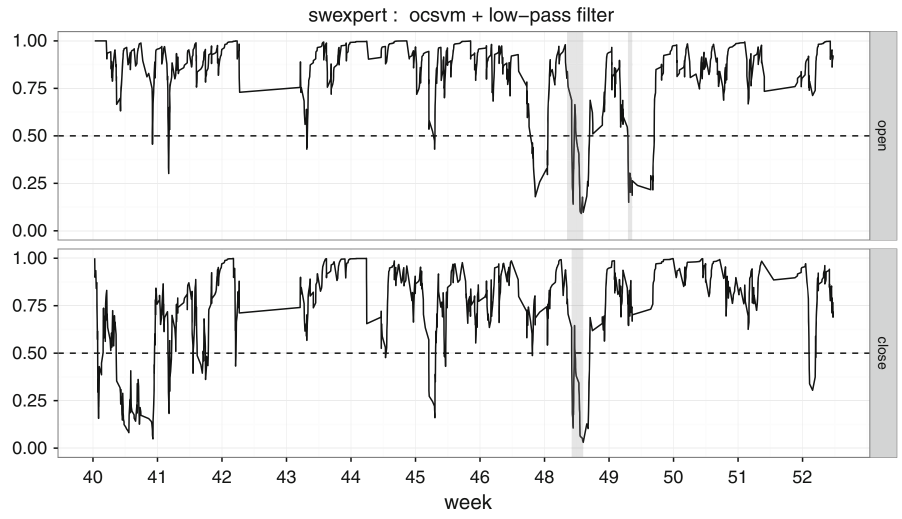

Fig. 10 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using sliding window with expert feedback and ocsvm for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

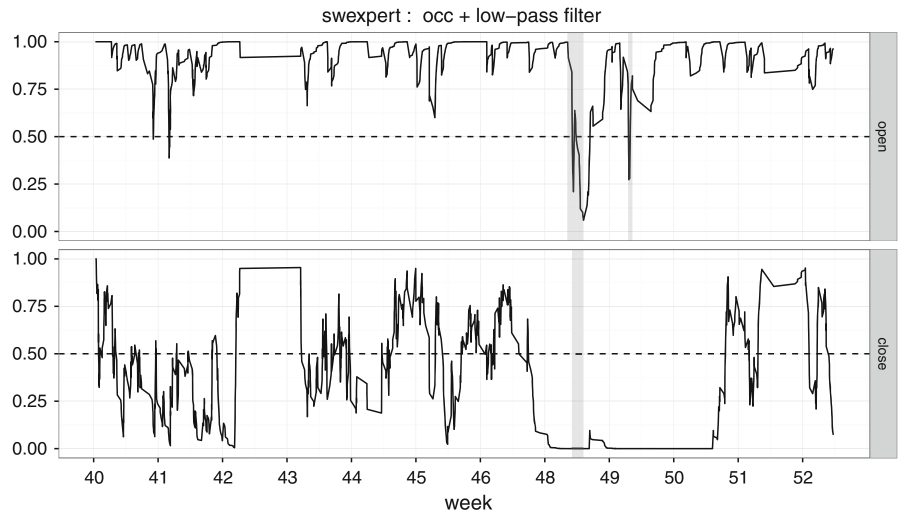

Fig. 11 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using sliding window with expert feedback and occ for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

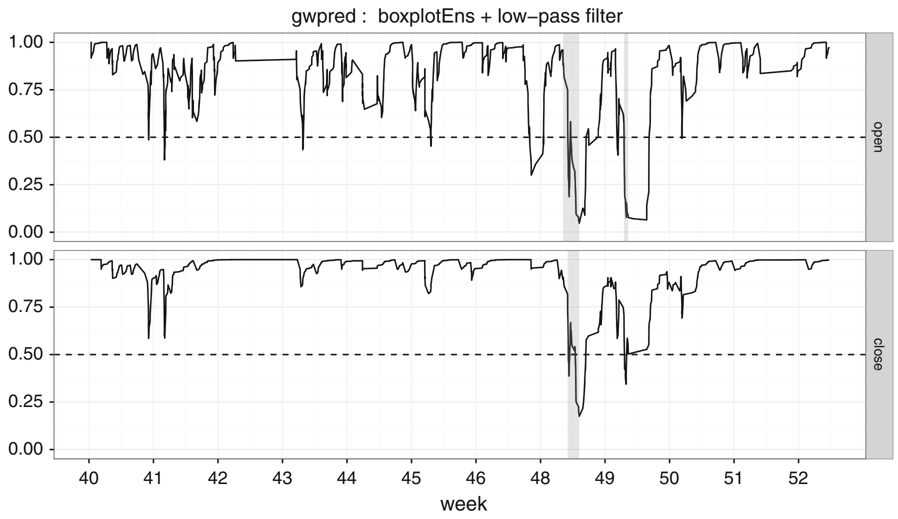

Fig. 12 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using growing window with self-feedback and boxplotEns for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

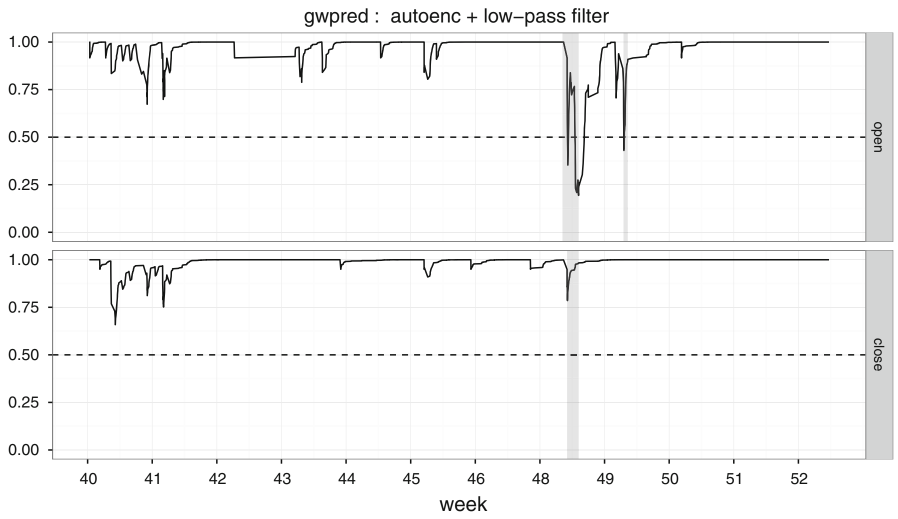

Fig. 13 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using growing window with self-feedback and autoenc for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

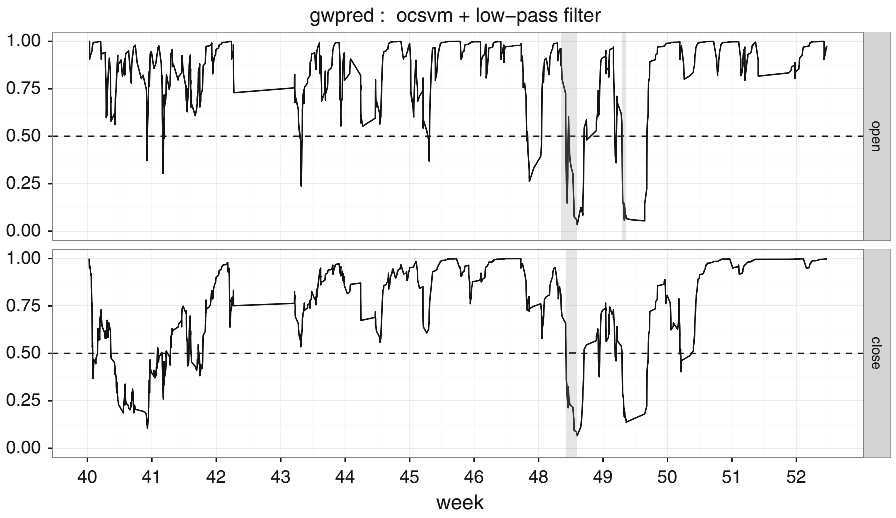

Fig. 14 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using growing window with self-feedback and ocsvm for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

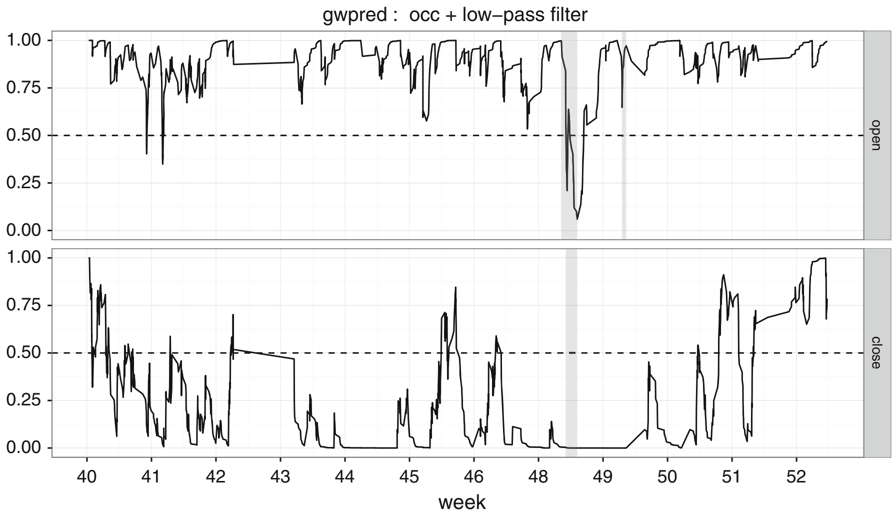

Fig. 15 The probability of normal state (for open cycles— *top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using growing window with self-feedback and occ for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

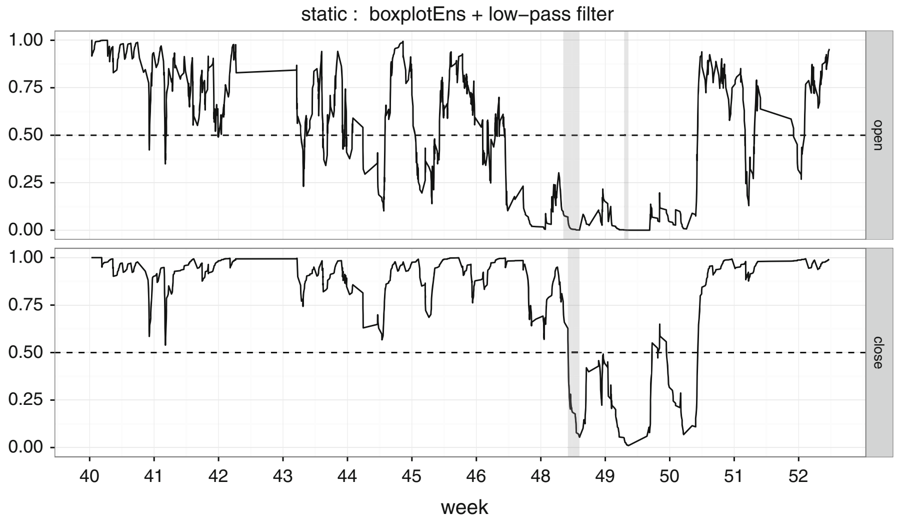

Fig. 16 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using static window and boxplotEns for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

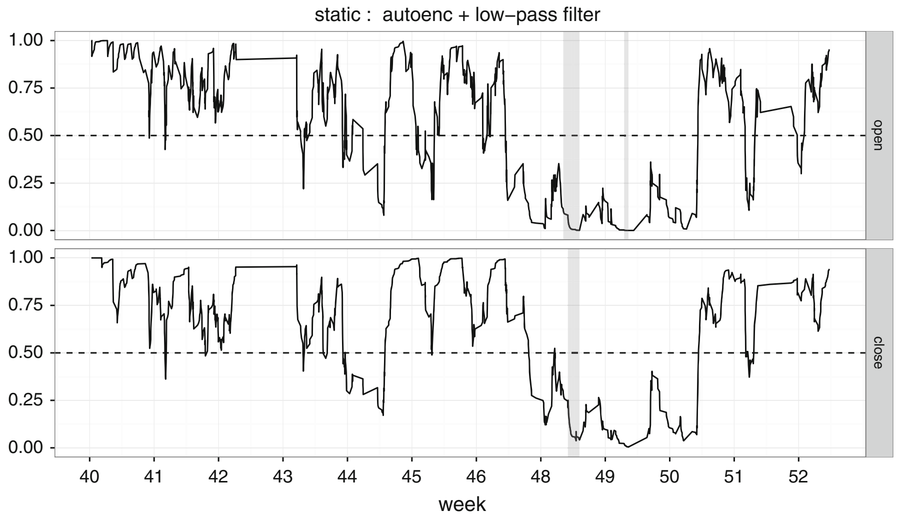

Fig. 17 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using static window and autoencoder for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

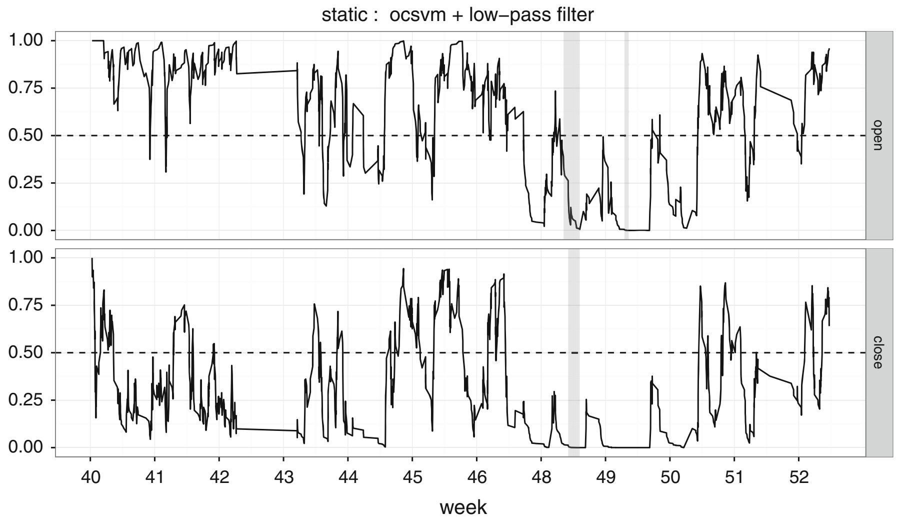

Fig. 18 The probability of normal state (for open cycles—*top panel* and close cycles—*bottom panel*) as estimated by the low-pass filter using static window and ocsvm for failure sequence detection. A failure is predicted when this probability drops below 0.5. The actual failures are indicated by *vertical grey bars*

References

Aggarwal, C. C. (2013). *Outlier analysis*. Berlin: Springer.

Angeli, C., & Chatzinikolaou, A. (2004). On-line fault detection techniques for technical systems: A survey. *International Journal of Computer Science & Applications*, *1*(1), 12–30.

Basseville, M. (1988). Detecting changes in signals and systems a survey. *Automatica*, *24*(3), 309–326.

Basseville, M., & Nikiforov, I. V. (1993). *Detection of abrupt changes: Theory and application* (Vol. 104). Englewood Cliffs: Prentice Hall.

Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). Lof: Identifying density-based local outliers. *ACM Sigmod Record, ACM*, *29*, 93–104.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, *41*(3), 15.

Dietterich, T. G., Lathrop, R. H., & Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artif Intell*, *89*(1–2), 31–71.

Duyar, A. (2011). Simplifying predictive maintenance. *Orbit*, *31*(1), 38–45.

Eugene Dubossarsky, Y.T. (2015). *Autoencoder: Sparse autoencoder for automatic learning of representative features from unlabeled data*. http://CRAN.R-project.org/package=autoencoder, r package version 1.1

European Parliament and Council of the European Union of 7 July 2010 (2010) Directive 2010/40/EU on the framework for the deployment of intelligent transport systems in the field of road transport and for interfaces with other modes of transport. Official Journal of the European Union

Gupta, M., Gao, J., Aggarwal, C. C., & Han, J. (2014). Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and data Engineering*, *26*(9), 2250–2267.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explorations*, *11*(1), 10–18.

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hawkins, D. M. (1980). *Identification of outliers* (Vol. 11). London: Chapman and Hall.

Hempstalk, K., Frank, E., & Witten, I.H. (2008). One-class classification by combining density and class probability estimation. In: Daelemans W, Goethals B, Morik K (eds) *Machine Learning and Knowledge Discovery in Databases, European Conference*, ECML/PKDD 2008, Antwerp, Belgium, September 15–19, 2008, Proceedings, Part I, Springer, Lecture Notes in Computer Science, vol 5211, pp 505–519

Hornik, K., Buchta, C., & Zeileis, A. (2009). Open-source machine learning: R meets Weka. *Computational Statistics*, *24*(2), 225–232.

Japkowicz, N., Myers, C., & Gluck, M.A. (1995). A novelty detection approach to classification. In*Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, IJCAI 95, Mo ntréal Québec, Canada, August 20–25 1995, 2 Volumes, Morgan Kaufmann, pp 518–523

Katipamula, S., & Brambley, M. R. (2005). Methods for fault detection, diagnostics, and prognostics for building systems a review, part i. *International Journal of HVAC Research*, *11*(1), 3–25.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2014). e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. http://CRAN.R-project.org/package=e1071, r package version 1.6-4

Nowlan, F., & Heap, H. (1978). *Reliability-centered maintenance*. Washington, D.C.: Dolby Access Press

Papadimitropoulos, A., Rovithakis, G. A., & Parisini, T. (2007). Fault detection in mechanical systems with friction phenomena: An online neural approximation approach. *IEEE Transactions on Neural Networks*, *18*(4), 1067–1082.

Patton, R. J., Frank, P. M., & Clark, R. N. (2010). *Issues of Fault Diagnosis for Dynamic Systems* (1st ed.). Berlin: Springer Publishing Company Incorporated.

Provost, F., & Domingos, P. (2003). Tree induction for probability-based ranking. *Machine Learning*, *52*(3), 199–215.

R Core Team (2014) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org/

Schubert, E., Zimek, A., & Kriegel, H. P. (2014). Local outlier detection reconsidered: A generalized view on locality with applications to spatial, video, and network outlier detection. *Data Mining and Knowledge Discovery*, *28*(1), 190–237.

Shenoi, B. A. (2005). *Introduction to digital signal processing and filter design*. Hoboken: John Wiley & Sons.

Sipos, R., Fradkin, D., Moerchen, F., & Wang, Z. (2014). Log-based predictive maintenance. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York, NY, USA, KDD '14, pp 1867–1876

Sluban, B., Gamberger, D., & Lavrač, N. (2010). Performance analysis of class noise detection algorithms. *Proceedings of the 2010 Conference on STAIRS 2010: Proceedings of the Fifth Starting AI Researchers' Symposium* (pp. 303–314). Amsterdam, The Netherlands: IOS Press.

Tax, D. (2001). One-class classification: Concept learning in the absence of counter-examples. PhD thesis, Technische Universiteit Delft

Tax, D. M. J., & Duin, R. P. W. (2004). Support vector data description. *Machine Learning*, *54*(1), 45–66.

Tukey, J. W. (1977). *Exploratory Data Analysis*. Boston: Addison-Wesley.

Ungar, L., Powell, B., & Kamens, S. (1990). Adaptive networks for fault diagnosis and process control. *Computers & Chemical Engineering*, *14*(4), 561–572.

Vaidyanathan, R., & Venkatasubramanian, V. (1992). Representing and diagnosing dynamic process data using neural networks. *Engineering Applications of Artificial Intelligence*, *5*(1), 11–21.

Venkatasubramanian, V., & Chan, K. (1989). A neural network methodology for process fault diagnosis. *AIChE Journal*, *35*(12), 1993–2002.

Venkatasubramanian, V., Vaidyanathan, R., & Yamamoto, Y. (1990). Process fault detection and diagnosis using neural networksi. Steady-state processes. *Computers & Chemical Engineering*, *14*(7), 699–712.

Watanabe, K., Matsuura, I., Abe, M., Kubota, M., & Himmelblau, D. (1989). Incipient fault diagnosis of chemical processes via artificial neural networks. *AIChE Journal*, *35*(11), 1803–1812.

Watanabe, K., Hirota, S., Hou, L., & Himmelblau, D. (1994). Diagnosis of multiple simultaneous fault via hierarchical artificial neural networks. *AIChE Journal*, *40*(5), 839–848.

Yilboga, H., Eker, OF., Guculu, A., & Camci, F. (2010). Failure prediction on railway turnouts using time delay neural networks. In: *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pp 134–137

Zhang, J., Yan, Q., Zhang, Y., & Huang, Z. (2006). Novel fault class detection based on novelty detection methods. In D. Huang, K. Li, & G. Irwin (Eds.), *Intelligent computing in signal processing and pattern recognition, lecture notes in control and information sciences* (pp. 082–987). Berlin: Springer.