A Machine Learning-based Approach for Advanced Monitoring of Automated Equipment for the Entertainment Industry

Michele Berno*, Marco Canil*, Nicola Chiarello*, Luca Piazzon*, Fabio Berti†, Francesca Ferrari†, Alessandro Zaupa†, Nicola Ferro*, Michele Rossi*, Gian Antonio Susto*
*Department of Information Engineering, 35131 Padova, Italy,
†Antonio Zamperla S.p.A., 36077 Altavilla Vicentina (VI), Italy.

2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT) | 978-1-6654-1980-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/METROIND4.0IOT51437.2021.9488481

***Abstract*—Machine Learning-based approaches are revolutionizing the way in which complex systems and machines are monitored and controlled. In this work, we present a smart monitoring system that combines a big data architecture with an unsupervised anomaly detection technique, targeting the automated equipment in the entertainment industry. Anomaly detection uses state-of-the-art univariate and multivariate algorithms, as well as recently proposed techniques in the field of explainable artificial intelligence, to achieve enhanced monitoring capabilities and optimize service operations. The monitoring system is here presented and tested on a real world case study, i.e., an amusement park ride.**

***Index Terms*—anomaly detection, big data, entertainment industry, explainable artificial intelligence, industry 4.0, predictive maintenance, unsupervised learning.**

I. Introduction

The industry 4.0 paradigm has paved the way for the extensive adoption of data-driven approaches to optimize products [1], production [2] and services [3]. The availability of historical data and big data infrastructures has enabled new equipment functionalities and their effective implementation, both technically and from a business perspective [4].

In the context of advanced monitoring of complex systems, two main Machine Learning (ML)-based technologies have emerged in recent years: unsupervised anomaly detection (AD) [5], that aims at providing enhanced diagnostic capabilities, and predictive maintenance [6], with the purpose of predicting failures/degradation to enable the early intervention of maintenance operators. In the literature, unsupervised AD tools adopt (i) multivariate approaches based on tabular data and (ii) univariate approaches working with time-series [7]: (i) multivariate approaches [8] have the advantage of capturing multivariate anomalous behaviour that typically goes undetected by classic chart-based monitoring tools, but, when applied to time-series data, they entail the use of feature extraction procedures that are typically time-consuming for the developers and may lead to loss of information; (ii) univariate approaches typically work by *predicting residuals*, i.e., comparing measured and forecast time-series data, and raising an alarm as their difference exceeds a threshold. While Deep learning techniques are available for (i) and (ii), they typically need to be adapted to cope with discrete production data, where time-series are usually split into batches representing the machine cycles [9]. Another relevant issue in the monitoring field is the so-called *concept drift*, which means that the statistical proprieties of the target variables change over time in an unforeseen way [10], posing the additional challenge of tracking or estimating the drift.

In this work, we present a new big data architecture that combines state-of-the-art univariate and multivariate approaches for AD and recently proposed techniques in the field of eXplainable Artificial Intelligence (XAI) that are proposed to reach enhanced monitoring capabilities and optimize service operation; the design of such system was motivated by a use case coming from the industry of automation for entertainment, i.e., the industry that produces automated rides, machines and attractions for entertainment parks. The rest of the paper is organized as follows: in Section II the considered use case and the challenges of its industry are detailed; in Section III the proposed approach for AD is presented and the methodological tools adopted are detailed. In Section IV the database infrastructure for hosting the proposed approach is illustrated, while in Section V the experimental results are reported. Finally, concluding remarks and future works are reported in Section VI.

II. Automation for the Entertainment Industry

Companies like Antonio Zamperla S.p.A., involved in the development and production of entertainment rides, are continuously striving for their rides to become safer, greener and more efficient. Smart monitoring systems are expected to enrich the entertainment industry with a number of key features. For example, faults could be predicted in advance, or data analysis during the testing of the ride's prototypes could allow getting deeper insights onto various design choices. Furthermore, maintenance operations are nowadays performed manually, following ride's manuals or government directives that are usually scheduled on a periodic basis, regardless of the actual conditions of the machines. This implies that,

Corresponding author: gianantonio.susto@unipd.it
M. Berno, M. Canil, N. Chiarello, L. Piazzon equally contributed.
This work has been supported by the Regione Veneto POR FESR 2014-2020. Asse 1. Azione 1.1.4 (*Bando per il sostegno a progetti sviluppati da Aggregazioni di imprese*) initiative for the project "Trasformazione digitale innovativa nell'industria dell'entertainment" and by MIUR (Italian Minister for Education) under the initiative "Departments of Excellence" (Law 232/2016).

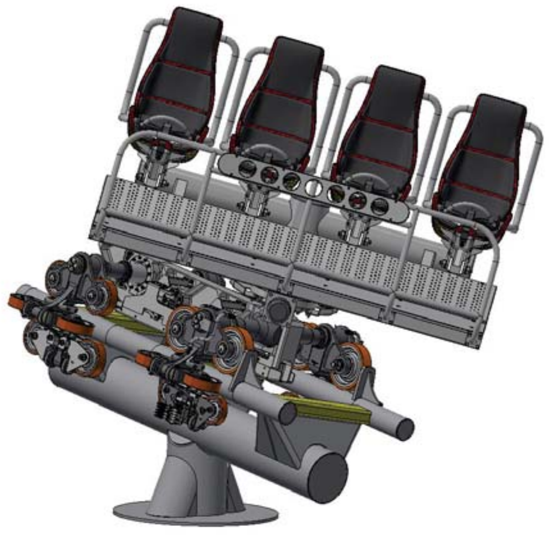


Fig. 1. Case study: schema of the Zamperla DangleZ ride.

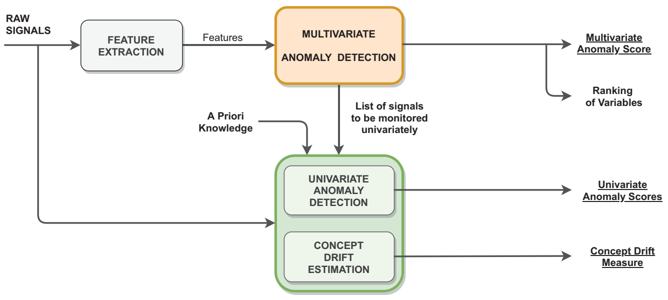
oftentimes, maintenance is performed without a real need, entailing a waste of time, human resources and material. This practice, although being cautious and robust, is quite inefficient. Smart monitoring systems would allow a change of paradigm from the current conservative approach to a greener and more efficient one. Lastly, automated supervision techniques would enforce the safeness of the rides by detecting subtle anomalies, which would be hardly identified by a human supervisor.

In this work, the case study is represented by a coaster (Zamperla's *DangleZ*) whose seats can freely move during the ride, independently of the underlying chassis, see Fig. 1. In this case study, the coaster track is 133 meters long.

The data acquisition process provides, by means of an on-board *PLC*, a set of $q = 55$ different time series acquired with irregular sampling rate. Each time series represents a signal which can be analyzed to monitor the behavior of the machine:

- nine signals are generated by the equipment sensors that report the transit of the coaster over different locations of the rail;
- fifteen signals describe voltages, currents, frequencies and other physical quantities denoting the consumption and the movement of the machine;
- five signals are acquired by a weather station that detects the condition of the surrounding environment;
- the remaining signals are needed to check the correct working conditions and the security of the machine (for instance the state of the safety button).

Data are divided into sessions (or tests) and each session is composed of a number of cycles. Each cycle collects the data generated during one run of the coaster, from the moment it leaves the station to its return to the station. Each session contains data coming from a single day of measurements.


RAW SIGNALS
FEATURE EXTRACTION
Features
MULTIVARIATE ANOMALY DETECTION
Multivariate Anomaly Score
Ranking of Variables
List of signals to be monitored univariately
A Priori Knowledge
UNIVARIATE ANOMALY DETECTION
Univariate Anomaly Scores
CONCEPT DRIFT ESTIMATION
Concept Drift Measure
Fig. 2. Functional schema of the proposed Anomaly Detection approach.

III. Proposed Approach for Anomaly Detection

The proposed approach integrates univariate and multivariate methods, as illustrated in Fig. 2:

- at a given machine cycle, a set of raw signals $\mathcal{X} = \{X_1, \ldots, X_q\}$ coming from the equipment sensors (and additional context information) are fed to a *feature extraction* block that computes features $x_1, \ldots, x_p$, where it typically holds $p \neq q$. While signals X_j usually are time-series, features x_i are scalars. Here, we consider the case where each feature x_i is computed from a single raw signal X_j and we denote the set of features computed from X_j by $\mathcal{S}_{X_j}$;
- such features are used within a multivariate AD block having two objectives: (i) obtaining a so-called *health factor* $s(\cdot)$ of the system status, a quantitative indicator that summarizes the degree of "outlierness" of the machine cycle $\mathcal{X}$ under exam; (ii) using an XAI approach to obtain a *feature importance score* $f_i(\mathcal{X})$, $\forall i \in \{1, \ldots, p\}$. $f_i(\mathcal{X})$ is a quantitative index that summarizes the impact of feature x_i in identifying a machine cycle as anomalous. In Fig. 3, we show the feature importance score for an anomalous encoder cycle: in this case, feature no. 22 (x_{22}) is the one having the highest importance in the detection of the event;
- raw signals that deserve to be monitored separately with a time-series based (univariate) approach are detected based on (i) expert knowledge and on (ii) the feature importance $f_i(\mathcal{X})$ coming from the AD module. The rationale is that some anomaly types are better identified by analyzing a specific time-series (see the example discussed in Fig. 9 of Section V). To be reliably detected, the respective signal X_i should be independently assessed, thus avoiding the loss of information descending from a multi-variate feature extraction procedure. To identify such signals, all data points $\mathcal{X}$ are considered. Raw signals to be processed by a univariate approach are identified by applying the following conditions:
  1) a cycle $\mathcal{X}$ is tagged as "anomalous" if it holds
  $$s(\mathcal{X}) > \tau, \tag{1}$$
  where τ is a pre-defined threshold on the anomaly score (under the assumption that "high" values

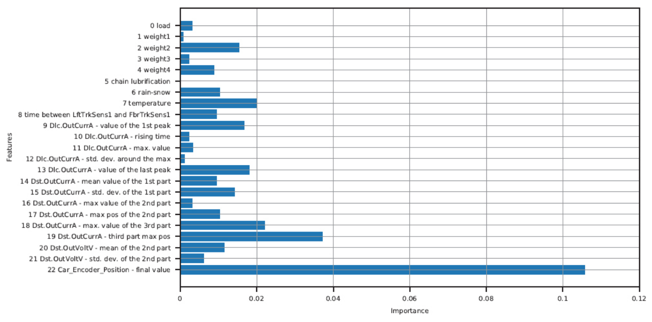


Fig. 3. Feature importance: anomalous encoder cycle.

of the anomaly score indicate a high degree of outlierness). For the problem at hand, we used $\tau = 0.55$ w.r.t. the anomaly score generated by an Isolation Forest (that will be introduced shortly in Section III-A);

2) if, for an anomalous cycle, $\mathcal{X}_a$, we detect that the features obtained from a single time-series X_i are more important than all the others in explaining the anomaly, then the corresponding time-series X_i is sent to the univariate monitoring block. Formally X_i is selected

$$\begin{aligned} \text{if there exists} \quad & f_j(\mathcal{X}) > \delta f_k(\mathcal{X}), \\ \text{where} \quad & j, k \in \{1, \ldots, p\}, \\ \text{and} \quad & x_j \in \mathcal{S}_{X_i}, x_k \notin \mathcal{S}_{X_i}, \end{aligned} \tag{2}$$

where $\delta > 1$ is a pre-defined quantity (for our results, we have set $\delta = 2$);

3) if, besides the two previous steps, some expert knowledge indicates that some raw signals are particularly relevant by themselves, then, they should be monitored univariately as well.

The raw signals identified by the above procedure are then monitored separately as described in Section III-B. The user is finally provided with complementary information coming from the two module types, with anomaly scores, system monitoring indications, ranking of variables and concept drift estimations, allowing for a guided Root Cause Analysis.

A. *Multivariate Anomaly Detection: Isolation Forest*

Isolation Forest [11] is a popular unsupervised algorithm for AD that exploits an isolation procedure to infer a measure of outlierness for each data point in a dataset. It relies on the assumption that anomalies are the minority of instances and they have attribute values that sharply differ from those of normal points.

The isolation procedure is based on an ensemble of tree-like structures, termed *Isolation Trees*: each of them is built, starting from the root, by recursively splitting a sub-sample of data points between the left child and the right child of a node. The splitting test is based on a randomly chosen splitting variable and a randomly chosen splitting threshold: for each internal node, the instances whose corresponding variable is smaller than the threshold are sent to the left child, while the others are sent to the right one. The process is iterated until all the data points are isolated to a leaf of the tree or until a predefined depth is reached. Many Isolation Trees are computed to create an Isolation Forest, randomly selecting the starting sub-sample of data points. Anomalies, due to their nature, are more likely to be isolated close to the root of a tree, therefore, the measure of outlierness associated with an observation, called *anomaly score*, is computed by evaluating the mean path length of such an observation on the various Isolation Trees. The closer the score gets to one, the more likely the observation is anomalous. A thresholding operation can be performed on the anomaly score associated with all observations to obtain binary labels, i.e., "0" for inliers and "1" for outliers.

Recently, an approach for providing a ranked list of variables has been presented [12]: the Depth-based Feature Importance for the Isolation Forest (DIFFI) provides both global and local feature importance in the form of ranked variables. We refer the interested readers to the original paper [12] for a detailed description of the interpretability approach.

B. *Univariate analysis: Grow When Required network*

The univariate analysis is based on a customized implementation of the Growing When Required (GWR) neural gas network [13].

1) GWR algorithm: GWR is an enhanced version of the Growing Neural Gas (GNG) algorithm [14]. Like GNGs, it is an unsupervised learning method that (a) allows obtaining a perfectly topology-preserving representation of the input data and (b) is able to adapt to time-varying distributions. The major improvement brought by GWR is the ability to automatically tune the number of nodes needed for the GWR representation to reach a desired approximation accuracy. For further details and for the definitions of the quantities involved in the algorithm we refer the reader to [13].

Our implementation differs from the original in the update rule for the firing counter of the winning neuron s and its neighbours i, that is, using the notation of [13],

$$h_s(t) = \alpha_b \times h_s(t-1), \tag{3}$$

$$h_i(t) = \alpha_i \times h_i(t-1), \tag{4}$$

where $0 \leq \alpha_b \leq \alpha_i \leq 1$ are training parameters. This slight modification makes the algorithm much easier to tune, rendering it easier to use by non-experts in GWR neural network design. GWR takes as inputs n-dimensional vectors. The key point for using it with time series is to consider each time instant as a different dimension, i.e., each point of the input time series corresponds to a new component in the input vector and represents a specific dimension in the feature space.

So, given a set of input signals (cycles) $\mathcal{X}$ with cardinality $|\mathcal{X}| = k$, GWR outputs a set of new signals $\mathcal{D}$, usually with $|\mathcal{D}| = m < k$, that can represent every element in $\mathcal{X}$ within a predefined error bound, e.g., considering a *norm-2* distance. We call $\mathcal{D}$ a *dictionary* and the elements of $\mathcal{D}$, $\mathbf{p}$, *prototypes*.

A dictionary can be either *static* or *dynamic*. It is *static* if, after being created, it cannot be modified. It is *dynamic* if, after its creation, it can be (dynamically) updated using new input vectors. Dictionaries are used to represent the state of the machine in a particular moment.

2) Concept drift estimation: to estimate the concept drift of the machine, a *static* dictionary $\mathcal{D}_s$ is created using data collected when the machine is in a reference condition, at time $t = t_{\text{ref}}$. For example, one may create a dictionary using data gathered for the machine cycles immediately after some particular maintenance operation (e.g., lubricating the gears) has just been carried out. Then, calling $\mathcal{Z}$ the set of data obtained after the creation of $\mathcal{D}_s$, that is, for $t > t_{\text{ref}}$, each cycle $\boldsymbol{z}_t \in \mathcal{Z}$ can be evaluated against $\mathcal{D}_s$ through the following procedure

1) using some distance metric m_1, find the element $\boldsymbol{p}^* \in \mathcal{D}_s$ that is closest to $\boldsymbol{z}_t$;
2) using another distance metric m_2, compute the distance d_t between $\boldsymbol{z}_t$ and $\boldsymbol{p}^*$.

d_t is the *concept drift* of the considered signal at time t with respect to the moment in which dictionary $\mathcal{D}_s$ was created. In our experiments, we used $m_1 = m_2 = ||\cdot||_2$, but other distance measures can be used ($||\cdot||_1$, $||\cdot||_\infty, \dots$).

3) Anomaly detection: when dealing with the AD task, a *dynamic* dictionary $\mathcal{D}_d$ should be created and continuously updated using the most recent data. In this way, it always provides a representation of the current state of the machine. Assuming that $\mathcal{D}_d$ is updated with all the data up to time (cycle) $t-1$, the evaluation procedure for a signal $\boldsymbol{z}_t$ obtained at time (cycle) t is as follows:

1) using some distance metric m_1, find the element $\boldsymbol{p}^* \in \mathcal{D}_d$ that is closest to $\boldsymbol{z}_t$;
2) using another distance metric m_2, compute the distance a_t between $\boldsymbol{z}_t$ and $\boldsymbol{p}^*$.

If the *anomaly score* a_t is greater than a threshold a_{th}, then, classify $\boldsymbol{z}_t$ as anomalous. In our experiments, we used $m_1 = m_2 = ||\cdot||_2$. The idea behind this approach is that, if the machine operates in normal (non-anomalous) conditions, every new data $\boldsymbol{z}_t$ should not be too far from the prototypes in $\mathcal{D}_d$, trained with all the data up to time $t-1$. Also, even if the machine undergoes some kind of concept drift, this is included and tracked by $\mathcal{D}_d$. Thus, a big value for a_t can only occur when $\boldsymbol{z}_t$ differs greatly from the values it used to have up to time $t-1$.

IV. Database and data management

Working on an ever growing dataset, a structured architecture is needed to maintain data consistency and access flexibility. Moreover, in our application, acquisition data coming from the rides contains a lot of redundancy and requires a long parsing procedure to get a reasonable data representation to be used in our algorithms. For this reasons, we rely on a relational database to organize both the data gathered from the sensors installed on Antonio Zamperla S.p.A. rides and the relevant information coming from the analysis, in the most favourable format for our AD procedures. All the park and ride related data are also modeled in the database to allow its usage in the real case scenario, where a single database instance should manage data coming from multiple parks.

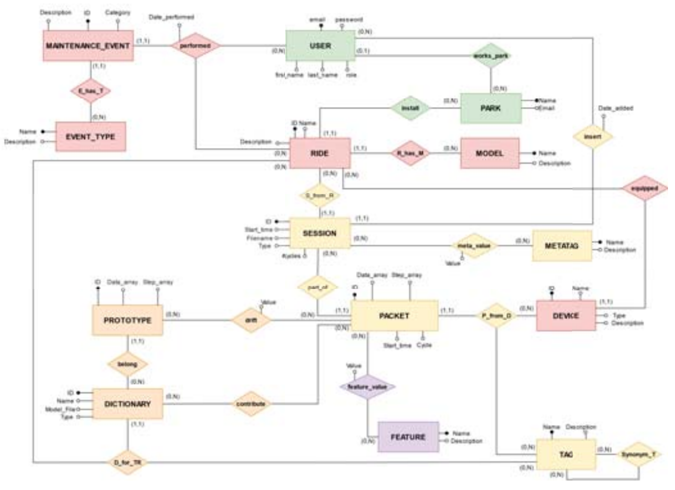


Fig. 4. Entity-Relationship schema of the database.

Fig. 4 shows the Entity-Relationship schema of our database, containing four principal sets of entities:

- *Parks and Users* (Green): Park names with a reference email and Profile information of the users allows to access and insert acquisition data (Park Managers), or analysis data (Analysts).
- *Ride related information* (Red): Each ride has a specific model and is equipped with a set of acquisition devices. All the maintenance events are stored together with their category (ordinary/extraordinary maintenance or hardware upgrade) and their type (e.g., components lubrication, replacement of gaskets). Events data is used in the AD system to identify the different feasible working conditions of the machines.
- *Acquisition Data* (Yellow): Data are organized into sessions, each of which represents a set of ride cycles under the same underlying conditions, i.e., the acquisition data related to a full morning or afternoon. These conditions are stored in the form of metatag values related to the session (e.g., Weather condition, load, notes). The acquisition is organized into cycles, meaning it the equivalent of a ticket paid by a final user. During the acquisition, each device gathers data for a set of variables, each identified by a unique TAG name. Synonymy relations between tags are stored to deal with old tag names in legacy acquisitions. Data are stored in packets, each one containing a vector of values (data_array) for a TAG coming from a device during a specific cycle.
- *Analysis related data*: Dictionaries (the obtained prototypes) for the Univariate AD and the corresponding drift value of each packet are stored (Orange entities). Features for the Multivariate AD and their values for each packet are stored (Purple entities).

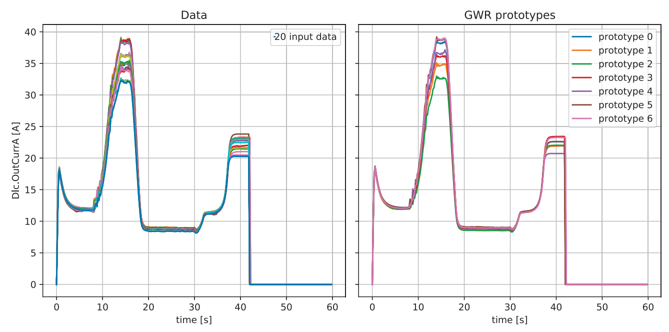


Fig. 5. The left part of the figure shows the values of an engine's current consumption in a particular session. These values have been used to create the reference dictionary depicted in the right part of the figure, using GWR.

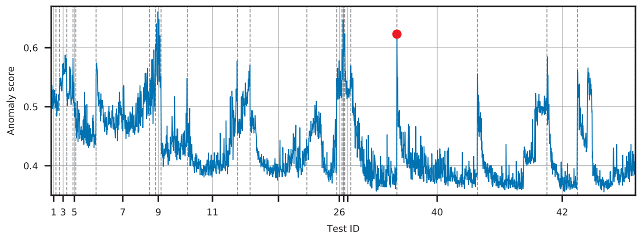


Fig. 6. Evolution of the anomaly score (blue). Vertical, dashed, lines separate different acquisition sessions. The red marker indicates an outlier.

V. Experimental Results

Considering the *multivariate analysis*, the feature extraction procedure analyzes the signals which change the most, to monitor all their possible behaviors and compute suitable features in signal parts which vary across acquisition cycles.

We started analyzing the whole set of 55 different signals acquired from the machine. At first, in light of the previous consideration, features were computed only from the 10 most variable (and most informative) signals. Subsequently, domain knowledge allowed us to identify the most important signals for the purpose of detecting anomalies, so that the number of signals to be used was reduced from 10 to 5. We remark that features are not only related to physical signals acquired from the machine, but also to parameters that characterize its working condition and the surrounding environment, like the machine load (number of customers carried), the lubrication, the ambient temperature and the presence of rain. To exemplify, Fig. 5 shows a particular electric current during a machine cycle: this signal was processed by extracting features related to the value of the first current peak, the rising time (to rise to 90% of the peak's amplitude), the maximum current value, a standard deviation in its neighborhood, and the value of the last peak before the signal drops to zero. Once all the features were computed, a correlation analysis was performed to remove redundancy, obtaining vectors composed of $p = 23$ independent features ready to be inputted into the Isolation Forest.

In Fig. 6, the evolution of the anomaly score obtained from the Isolation Forest is shown, while in Fig. 7 the feature importance is given for the outlier indicated with the red marker in Fig. 6.

For the *concept drift* estimation, we focused on one specific signal, namely, the current consumption of a particular engine, since it highly correlates with the lubrication that is performed periodically on the machine. We used, as a reference, measurements collected in a day when the gears were lubricated and during which the machine was tested in all its possible load configurations (i.e., recalling that, as shown in Fig. 1, four seats are available, the possible configurations correspond to 1, 2, 3 or 4 occupied seats). The left part of Fig. 5 illustrates these data. Some of these configurations generate traces that are clearly identifiable by visual inspection, as a higher number of occupied seats implies more weight and, therefore, a higher current consumption. However, the proposed GWR approach does this automatically, extracting and then storing the typical patterns (the prototypes) into the dictionary depicted in the right part of Fig. 5.

As discussed in Section III-B, such dictionary was used to compute the drift of the machine from a reference setup. In the considered example, the drift depends on the service time of the lubricating oil: as time evolves, the oil film gets thinner, the friction gets higher and, consequently, the current consumption increases. We computed the mean drift of every session, that is, averaging the drifts of the cycles belonging to the same session. The results are shown in Fig. 8, where the label *total* identifies when a total lubrication was carried out. A positive correlation between the reduction of the lubricating oil film and the current consumption drift can be observed. The current drift can be used to monitor the conditions of the ride and, setting a threshold on it, it can be used to (automatically) infer when lubrication is required.

Next, we show how the dictionary can be used to also detect anomalies on machine cycles. After creating the reference dictionary, a second *dynamic* dictionary $\mathcal{D}_d$ was updated with data until the 1st of December 2020. Then, this latter dictionary was used to evaluate measures taken on the 3rd of December 2020, which was a snowy day. Note that snow is not common in the place where the machine was deployed and it was the first time that measurements with such weather conditions were

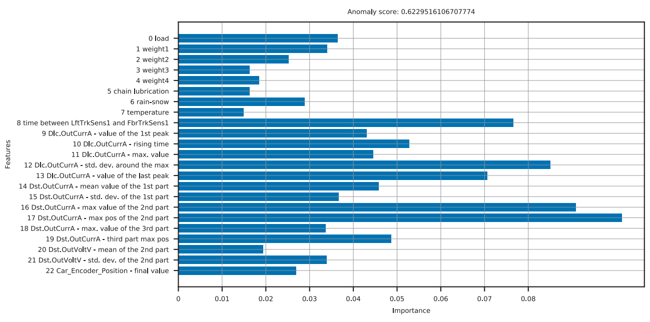


Fig. 7. Feature importance for the anomalous cycle indicated with the red marker in Fig. 6

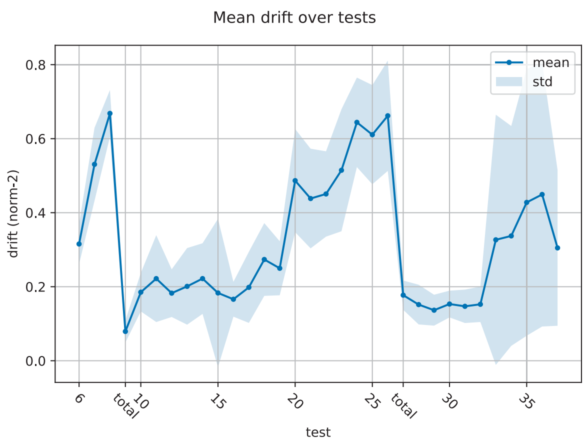


Fig. 8. Mean and standard deviation of the drift of the different sessions. The label *total* identifies the instants when a total lubrication was carried out. A clear relation between the reduction of the lubrication and the current consumption drift can be observed. The graph represents data acquired between the 9th of September and the 30th of December, 2020.

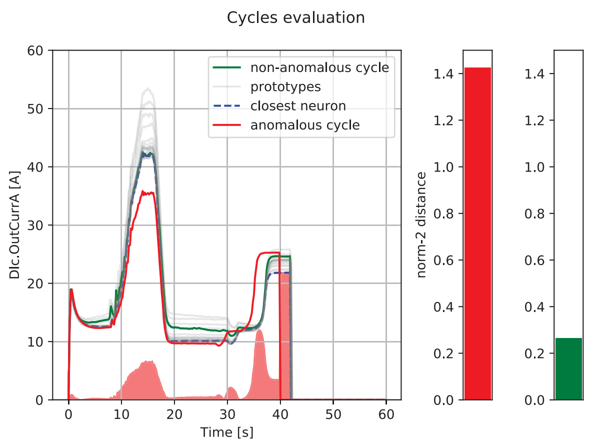


Fig. 9. $\boldsymbol{z}_a$ and $\boldsymbol{z}_n$ represent, respectively, an anomalous cycle (red line) and a non-anomalous one (green line). The dictionary prototypes are plotted in grey. The cycle under evaluation is $\boldsymbol{z}_a$ and the dashed blue line indicates the prototype $\boldsymbol{p}^*$ that is closest to it. The red shaded area shows the absolute difference between $\boldsymbol{z}_a$ and $\boldsymbol{p}^*$. The two bars on the right represent the anomaly score for $\boldsymbol{z}_a$ (red) and $\boldsymbol{z}_n$ (green), respectively.

taken. Fig. 9 illustrates the evolution of an anomalous cycle $\boldsymbol{z}_a$ from this snowy day: the dictionary prototypes are plotted in grey, $\boldsymbol{z}_a$ is represented by the red solid line, while the dashed blue line represents the prototype $\boldsymbol{p}^*$ that is closest to the evaluated cycle, and, so, the one that is used to compute the anomaly score. The red shaded area at the bottom of the plot represents the absolute difference between $\boldsymbol{z}_a$ and $\boldsymbol{p}^*$. Remember that the distance is computed by summing the squared differences between the two signals (norm-2) and note that, during the evaluation, these are normalized. The anomaly score for non-anomalous cycles is usually between 0.2 and 0.8 and, consequently, we decided to set the anomaly threshold at $a_{\text{th}} = 0.95$. The cycle from the snowy day, $\boldsymbol{z}_a$, originated an anomaly score of 1.42. The main contributions to this score are (i) the first peak at about 15s, that is much lower than that of the prototypes, and (ii) the final current peak, which is shifted to the left. The latter fact is probably because of the reduction of friction due to the snow and to the low temperature, which allowed the ride to complete the cycles earlier. During the same day, the machine and the rails on which it runs warmed up, and the current consumption returned to normal values as shown by the green line in Fig. 9.

VI. Conclusions and Future Work

In this work, an AD system to monitor amusement rides has been presented. The system combines a database management engine and a multi-faceted AD engine performing univariate and multi-variate analyses. The approach is unsupervised, making it appealing for scenarios where tagged information is unavailable or unreliable. Also, features are ranked according to their importance in explaining an alarm, using an explainable artificial intelligence method. Various extensions of the AD engine are possible, such as using other norms and adopting a semi-supervised technique, i.e., by additionally using tagged data, where tags can either come from manual labeling or from new sensors.

References

[1] G. Zambonin *et al.*, “Machine learning-based soft sensors for the estimation of laundry moisture content in household dryer appliances,” *Energies*, vol. 12, no. 20, p. 3843, 2019.

[2] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, “Intelligent manufacturing in the context of industry 4.0: a review,” *Engineering*, vol. 3, no. 5, pp. 616–630, 2017.

[3] J. Lee, H.-A. Kao, S. Yang *et al.*, “Service innovation and smart analytics for industry 4.0 and big data environment,” *Procedia Cirp*, vol. 16, no. 1, pp. 3–8, 2014.

[4] R. Sahal, J. G. Breslin, and M. I. Ali, “Big data and stream processing platforms for industry 4.0 requirements mapping for a predictive maintenance use case,” *Journal of Manufacturing Systems*, vol. 54, pp. 138–151, 2020.

[5] L. Stojanovic, M. Dinic, N. Stojanovic, and A. Stojadinovic, “Big-data-driven anomaly detection in industry (4.0): An approach and a case study,” in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 1647–1652.

[6] G. A. Susto *et al.*, “An adaptive machine learning decision system for flexible predictive maintenance,” in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, 2014, pp. 806–811.

[7] H. Wang, M. J. Bah, and M. Hammad, “Progress in outlier detection techniques: A survey,” *IEEE Access*, vol. 7, pp. 107 964–108 000, 2019.

[8] Y. Zhao, Z. Nasrullah, and Z. Li, “Pyod: A python toolbox for scalable outlier detection,” *Journal of Machine Learning Research*, vol. 20, pp. 1–7, 2019.

[9] M. Carletti, C. Masiero, A. Beghi, and G. A. Susto, “A deep learning approach for anomaly detection with industrial time series data: a refrigerators manufacturing case study,” *Procedia Manufacturing*, vol. 38, pp. 233–240, 2019.

[10] A. Tsymbal, “The problem of concept drift: definitions and related work,” *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[11] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[12] M. Carletti, M. Terzi, and G. A. Susto, “Interpretable anomaly detection with diffi: Depth-based feature importance for the isolation forest,” *arXiv preprint arXiv:2007.11117*, 2020.

[13] S. Marsland *et al.*, “A self-organising network that grows when required,” *Neural networks*, vol. 15, no. 8-9, pp. 1041–1058, 2002.

[14] B. Fritzke, “A growing neural gas network learns topologies,” in *Advances in neural information processing systems*, 1995, pp. 625–632.