Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 11 (2017) 1153 – 1161

Procedia MANUFACTURING

27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy

Data mining and machine learning for condition-based maintenance

Riccardo Accorsi[a], Riccardo Manzini[a]*, Pietro Pascarella[b], Marco Patella[b], Simone Sassi[a]

[a]*Department of Industrial Engineering, Alma Mater Studiorum Bologna, viale del Risorgimento 2 – 40136 Bologna (Italy)*
[b]*Department of Computer Science and Engineering, Alma Mater Studiorum Bologna, viale del Risorgimento 2 – 40136 Bologna (Italy)*

Abstract

Complex production systems may count thousands of parts and components, subjected to multiple physical and logical connections and interdependencies. This level of complexity inhibits the traditional and statistically-based approach to reliability engineering, failure prediction and maintenance planning. The existing ICT solutions simplify the on-field collection of large amount of data, but require models and tools able to create knowledge from these data. Key questions on how to predict in advance the performance of the production system and the associated failure events could be finally addressed. This paper introduces a set of data analytics models and methods that can be profitably used for decision making in general, and, specifically, in maintenance engineering. These classification models, specifically decision trees, random forests, and neural networks, are applied to a real-world case study, and the resulting accuracy on predicting faults is quantified and compared. We used the historical profiles of the energy variables of an high-speed packaging machine to find out some strategies for the prediction of a given failure. The conducted experiments demonstrate that the accuracy of the random forest is slightly better than the other methods, but even increases the probability of false alarm, which would result in unwanted production break-down. Even though the obtained results are promising, they leave room for further experiments based on the application of other classifiers, rather than the definition of customized methods able to embrace such complexity.

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing

Keywords: condition-based maintenance; data analytics; data mining; machine learning; failure event

* Corresponding author. Tel.: +39 051 2093406; fax: +39 051 2093411.
E-mail address: riccardo.manzini@unibo.it

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing
doi:10.1016/j.promfg.2017.07.239

1. Introduction

Condition based monitoring has become an important technique to ensure the machine/system availability by planning maintenance actions and reducing the need for the corrective maintenance **Errore. L'origine riferimento non è stata trovata.**. This paper presents an original conceptual framework for the implementation of condition-based maintenance in complex production systems. This framework integrates existing modelling approaches, methods, and algorithms frequently adopted by the decision making process in different fields and industrial sectors. This paper comes at the time when emerging technologies (e.g., Radio Frequency Identification (*RFID*), Micro-Electro-Mechanical Systems (*MEMS*), Supervisory Control and Data Acquisition (*SCADA*) systems, and Product Embedded Information Devices (*PEID*)) provide increasingly effective solutions to collect and monitor the real-time operating conditions of the components and functional groups of such production systems, drawing the new era of Industry 4.0. The existing ICT solutions simplify the on-field collection of large amount of data, but require models and tools able to create knowledge from these data. Key questions on how to predict in advanced the performance of the production system and the associated failure events could be addressed. This paper investigates such opportunities and challenges for a complex automatic production system, that involves thousands of mechanical and electric components. In addition, this paper illustrates a real-world application of existing data mining (*DM*) and machine learning (*ML*) models and methods for condition-based maintenance to a complex high-speed packaging machine of a leading Italian company.

The paper is organized as follows: Section 2 presents the condition-based maintenance strategy and introduces the selected case study. Section 3 presents the proposed conceptual framework, illustrates the adopted DM and ML techniques and illustrates the obtained results. Finally, Section 4 presents conclusions and final remarks.

2. Condition-Based Maintenance

A quite comprehensive overview of the different maintenance strategies for a production system is given in Figure 1. The combination of these strategies results into maintenance actions. According to the Standard EN 13306:2001 two main types of maintenance strategies can be identified: preventive and corrective maintenance. The former deals with the planning and scheduling of preventive maintenance or replacing tasks that prevent from potential failures events and resulting downtime (i.e., unavailability) of the production system. The latter encompasses all those replacing tasks called by an unpredicted failure of a component or a functional group. Since these events are (by the name) “unpredicted” they can significantly compromise the whole production system and propagate throughout the entire kinematic chain, thus increasing costs and time for replacement. For these reasons, corrective maintenance is far to be considered a fair strategy in practice in such automatic intensive and high-volume production systems.

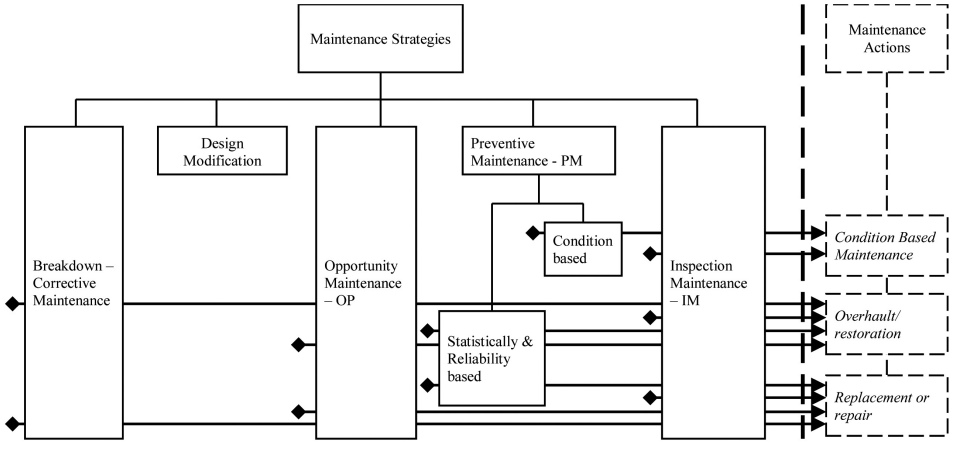

Figure 1: overview of maintenance strategies

Preventive maintenance is renowned as a best practice in the majority of the manufacturing systems. It can be further classified in two sub-strategies, as illustrated in

: reliability-based and condition-based maintenance. Reliability-based maintenance uses the historical failure profiles of a given machine to estimate the reliability curves of the single components and approaching the assessment of the whole system's reliability via Failure Mode and Effect Analysis (*FMEA*) and Failure Mode, Effects, and Criticality Analysis (*FMECA*). This strategy requires robust data gathering, a process of noise filtering, as well as the implementation of statistic reliability models, usually hard to be handled and customized by practitioners.

Condition-Based Maintenance (*CBM*) is built on monitoring (a set of) relevant variable(s) that is/are closely related to failures events **Errore. L'origine riferimento non è stata trovata.**. This strategy is also known as predictive maintenance (PdM) and prognostic maintenance and aims to predict the failure events of a system from the analysis of its *nominal vs. real* working conditions and parameters **Errore. L'origine riferimento non è stata trovata.**. CBM aims to prevent from system's breakdown by advising when and how to perform a replacing task in accordance with the monitored working profile and parameters of the machine. Nevertheless, in analyzing such complex manufacturing systems, where each machine counts thousands of components affected by several failure modes and behaviors, the first challenging step is to determine of the set of parameters to be controlled and monitored. This set needs an architecture of sensors generating a huge amount of records to be processed, stored, and analyzed. This level of complexity compels the adoption of data analytics and tailored methodologies that allows the process of learning from the machine. In order to illustrate these methodologies, the following section introduces the testbed for the application of the proposed conceptual framework toward the implementation of CBM strategies in complex manufacturing systems.

The case study presented in this article refers to a company producing automatic machines for the manufacturing of packaged items, starting from intermediate products. One of the automatic machines in the manufacturing line (called "maker") is the target of the illustrated CBM framework and its application in practice. The following failure prediction analyses were conducted for a selected failure type, named "rod break". This is one of the most crucial failure event affecting the machine. While the testbed for the application of our framework is a given machine, the designed methodology could be quickly extended to other machines and other failure events. The following section is to present some DM and ML techniques applied to predict the failures. These techniques have been applied to the illustrated case study and their accuracy and effectiveness to support CBM investigated and discussed.

3. Using Data Mining and Machine Learning

The goal of the proposed framework is the realization of a model to diagnose abnormal operational conditions of the automatic maker machine before they lead to actual malfunctions and/or stop the manufacturing process. At the same time, we would like to identify the areas that, most likely, caused the abnormal condition (in order to help troubleshooting). To achieve this goal, we used data analysis (DM and ML) techniques for the stages of organization, structuring, and analysis of data coming from the machines and for the processing of the training database (which is the output of the first phase) in order to create the diagnostic model. The conceptual framework of the proposed approach is depicted in Figure 2. In particular, the left portion of the figure shows DM techniques organized as "methods" and "models and algorithms"; the main right part, on the other hand, includes the five main components of the framework, showing activities and relative dependencies for the creation of the model:

- The problem setting component consists in the preliminary activities of selection of the particular machine and of the most significant faults/failures; then, with the help of experts, data related to such faults have been identified among those produced by the machine.
- The data selection and transformation component is in charge of the selection of available data for the creation of the training dataset, according to the knowledge provided by machine experts and by the cause-effect analysis of faults identified in the previous stage.
- The source system and staging area includes the retrieval and organization of raw data coming from heterogeneous data sources, like parameter logs (including values of user-settable machine quantities), message logs (including messages shown on displays), and energy data sampling (coming from sensors included), into a uniform structure that can be automatically processed by following stages.
- The dataset creation component is in charge of creating the dataset to be used by the diagnostic algorithms.

- Finally, the data modeling component includes the data analysis models and algorithms which are actually used to predict the machine behavior in real time.

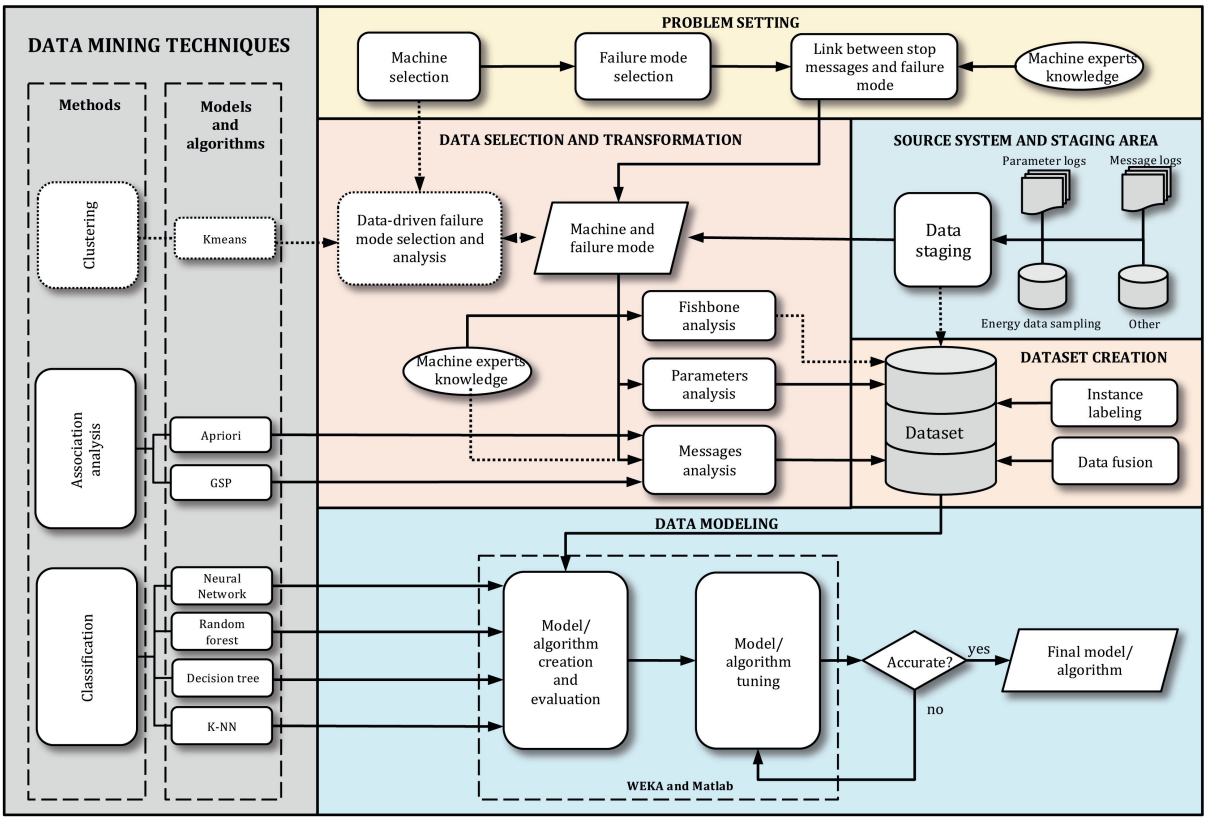

Figure 2: conceptual framework

Some of the basic assumptions of the model creation required the development of an a-priori knowledge about the different types of available data in order to select the set of significant variables suitable for the construction of the training database (see Section 2). Such stages included several meetings with staff members experienced on the analyzed machines, such as technicians, maintenance personnel, analysts, and designers, providing fundamental information on the machine behavior, the nature of the data, methods for their organization, and critical evaluation of analytical results. Such activities are indicated in Figure 2 as “machine experts knowledge” nodes. The models, therefore, would leverage on human knowledge, combining it with the inexpensiveness of automated techniques for data analysis, taking “the best of both worlds”.

The use of DM and ML techniques has proved useful in three different stages of the whole process:

1. Clustering techniques have been used to group machine data for detecting similar behaviors in an unsupervised way, in order to integrate experts’ knowledge to detect anomalies in the machine activity.
2. Association rules (Section 3.1) have been exploited to detect recurring machine behaviors following or preceding the failure events, in order to define additional variables complementing the set of existing data to help troubleshooting.
3. Finally, classification models (Section 3.2) were exploited to build the actual behavior model that is able to predict faults and (possibly) identifying the causes that probably caused the malfunction.

3.1. Association Rules

Association rules [4] are used to discover interesting relations between several occurring events. Such "exploration" techniques are typically used in the market basket and clickstream analysis to measure the affinity of products purchased by a particular consumer or that of visited web pages. The objective of this analysis is to highlight groups of events that occur together, e.g., to infer a cause-effect relationship. By analyzing the resulting rules (in the form "if *condition* then *result*"), we can select only those that express high significance and interest. The constraints mostly used to this end are minimum thresholds on support and confidence.

More precisely, given a set of items I and a set of transactions T, each transaction containing a subset of items in I, an association rule is an implication of the type $X \Rightarrow Y$ (X implies Y), with $X, Y \subset I$ and $X \cap Y = \emptyset$. According to the previous definition, X is called the rule *antecedent* or left-hand-side (LHS) while Y is the *consequent* or right-hand-side (RHS). The support s of rule $X \Rightarrow Y$ is then defined as the frequency of involved items in the dataset (i.e., in T), while its confidence c measures how often the rule has been found to be true:

$$s(X \Rightarrow Y) = \frac{|\{t \in T; X \cup Y \subseteq t\}|}{|T|} \qquad c(X \Rightarrow Y) = \frac{|\{t \in T; X \cup Y \subseteq t\}|}{|\{t \in T; X \subseteq t\}|} \quad (1)$$

In our model, algorithms for discovering association rules (namely, Apriori **Errore. L'origine riferimento non è stata trovata.** and GSP [6]) were used to analyze the machine behavior before a particular fault occurs. In particular, we investigated sequences of messages appearing on the Human-Machine Interface, selecting the most frequent ones preceding a fault: the goal is clearly to help troubleshooting, integrating additional binary variables to those already provided by the machine. Examples of rules discovered on the maker machine, similar to the one illustrated in Figure 2, are included in Table 1, where we are clearly only interested in those rules having "rod break" (the fault selected for that machine) as the consequent (thus confidence always equals 1, since we removed from T all item sets not including the consequent).

Table 1: association rules for the maker machine

Rule	Support
{EXCESSIVE REJECT – DEV} ⇒ {ROD BREAK}	0.89
{LOW PRODUCTION SPEED, EXCESSIVE REJECT – DEV} ⇒ {ROD BREAK}	0.87
{NOMINAL SPEED} ⇒ {ROD BREAK}	0.58
{EXCESSIVE REJECT – DEV, NOMINAL SPEED} ⇒ {ROD BREAK}	0.57
{LOWER REEL RELEASED} ⇒ {ROD BREAK}	0.45

Analysis of the above rules (with the help of machine experts) led us to discover the following interesting facts:

1. The first rule {excessive reject – DEV} ⇒ {rod break} expresses that, in 89% of cases, following the beginning of production and before the breaking of the rod, the "DEV" (i.e., a device controlling product quality parameters) communicated to the machine to discard an excessive amount of products. This behavior suggests that the cause of the fault is the same one triggering a large number of rejects.
2. The rule {lower reel released} ⇒ {rod break} states that, in 45% of cases of rod breaking, a change in the upload of raw materials has preceded the stop. According to experts, if a "rod break" fault occurs within 60,000 items manufactured following the change, then the cause of the stop can be attributed to quality problems of the newly inserted raw material.
3. Finally, rules containing "nominal speed" and "low production speed" messages are irrelevant and obvious, since they indicate that, following the beginning of production and before the fault occurred, machine was working at regular or low speed.

The first two rules allowed us to create two new Boolean variables which are added to the ones provided by the data staging phase and that are useful to troubleshoot possible machine stops:

- The variable "Excessive rejection – DEV" is true only when the cause of a fault may be due to the DEV (in particular, this happens after the "excessive reject – DEV" message until 15,000 items have been manufactured).
- On the other hand, the variable "Lower reel released" is true as long as the cause for a fault might be due to a material quality problem (in particular until 60,000 items have been manufactured following a material change).

3.2. Classification Models

Classification is a very popular ML problem, which requires identifying, in a set of existing categories (*classes*), to which one a new observation belongs. This is performed by exploiting an available knowledge, consisting in a *training set* of data containing observations whose category membership is known. The task of classification can be viewed as a three step process:

1. Training: using the training set a model is created; the training set consists of instances with several features including the one describing the instance class.
2. Evaluation: using a set of metrics the created model is evaluated; in particular, the model accuracy is estimated by exploiting data excluded from the training set (*test set*) and measuring the error rate of predicted classes on instances of the test set.
3. Use: the model is finally used on real data with unknown class; in this case, clearly, the class of new instances is unknown but all other attributes are known.

We exploited three different types of classifier to evaluate their performance on predicting faults: decision trees **Errore. L'origine riferimento non è stata trovata.**, random forests **Errore. L'origine riferimento non è stata trovata.**, and neural networks **Errore. L'origine riferimento non è stata trovata.**, all implemented within the Weka environment **Errore. L'origine riferimento non è stata trovata.**. A precise definition of such classifiers is clearly out of the scope of this paper, but we briefly describe them in the following to highlight their main characteristics, that will be exploited in their evaluation in Section 3.3.

3.2.1. Decision Trees

A decision tree is a classifier with a tree structure representing a succession of tests on the available attributes (attribute predictors) with the aim to predict an unknown attribute (the class to be discovered). Internal tree nodes represent a test on a given attribute and each node branch a possible result of the test. Tree leaves output the predicted class (see Figure 3, left).

The algorithm for building a decision tree is divided into two main steps: in the first (*growing*) stage, the tree is actually created, while the second (*pruning*) stage is used to eliminate some tree nodes in order to avoid *overfitting* (noise learning), thus deleting attributes that are deemed not "useful" for classification. The growing stage is clearly the most important one and it is based on choosing the order under which attributes are tested, in each internal tree node, and the attribute value that divides (*split*) instances to create a new tree node. In the algorithm used in the project, the split attribute is chosen as the one with maximum information gain, i.e., the attribute that is able to "better" separate instances belonging to different classes. This is recursively applied until all the elements in a node have the same class (thus a split is no longer possible): at this point, the recursion ends and a leaf node is created, labeled with the aforementioned class (if a node only contains instances belonging to different classes but with equal attribute predictors, a split is not possible and a leaf is created labeled with the most frequent class of that node). This top-down induction of decision trees **Errore. L'origine riferimento non è stata trovata.** is an example of a *greedy algorithm*, and it is by far the most common strategy for learning decision trees from data. Once the tree has been created and pruned, it can be used with a new instance by traversing it, starting from its root, following its branches by choosing at each node according to the values of attribute predictors. The tree visit ends in a leaf, whose label is the class predicted for the new instance.

A major advantage of decision trees is that the model they provide is highly interpretable, since they can also be displayed graphically in a way that is easy for non-experts to interpret.

3.2.2. Random Forests

A random forest is obtained by combining together several decision trees. To classify a new instance, this is given as input to every "forest" tree, obtaining its predicted class. The overall predicted class is obtained through a final voting stage. This usually allows avoiding decision trees' habit of *overfitting* the training set.

Performance of random forests depends on two primary factors: the correlation between forest trees (a higher correlation value leads to higher error rates) and the forest trees strength (a stronger tree leads to a lower error rate. Increasing the strength of each decision tree decreases the overall forest error rate). Random forests have, in general,

a higher accuracy with respect to decision trees. On the other hand, by using this approach, the possibility of interpreting the model is clearly lost, which is a major drawback with respect to decision trees.

3.2.3. Neural Networks

Research on (artificial) neural networks stems from the observation of the human brain, which is a complex, non-linear, and highly parallel computer, able to organize its elementary units (neurons) in order to perform some computations faster than any modern computer. The idea is to produce a model simulating the behavior of the human brain. A neural network is thus a processor, consisting of the interconnection of elementary computational units (neurons) with two basic characteristics: (a) the *knowledge*, which is acquired through a process of learning (training), and (b) the *weights*, the network parameters associated with the connections, where the network knowledge is stored.

Neural networks typically consist of multiple layers and the signal path traverses from front to back. Their structure is formed by (see Figure 3, right): *input nodes* (representing instance attributes), *hidden nodes* (the inner layers of the network, whose content is invisible, which represent a partial result of the application of the knowledge stored at input nodes), and *output nodes* (each one representing a predicted class). The neurons are thus the nodes of a network with processing capability. They receive in input a combination of signals (input vector), carrying out a transformation by way of a non-linear *activation* function (using the known weights values). Their output is transmitted to other nodes through weighted connections. Overall, a neural network represents an input-output connection that depends on: (a) the type of elements (complexity of the internal structure, different activation functions), (b) the network architecture (the number of nodes, the structure and orientation of connections), and (c) the values of internal parameters associated with elementary units and connections, learned during the training.

The goal of the training of a neural network is to define an optimal set of weights to be assigned to the network neurons. Supposing that weights are initially assigned random values, the following steps are repeated until the network accuracy exceeds a predetermined threshold or a maximum number of iterations has been reached:

1. For each neuron, the result of applying the weights and the activation instance to the input instance is computed.
2. The computations are propagated forwards (through the hidden layers) towards the output layer.
3. The predicted class (calculated output) is compared with the expected one and the error is computed.
4. The error is back-propagated through the hidden layers and weights are recalculated (this back-propagation guarantees a property of global convergence).

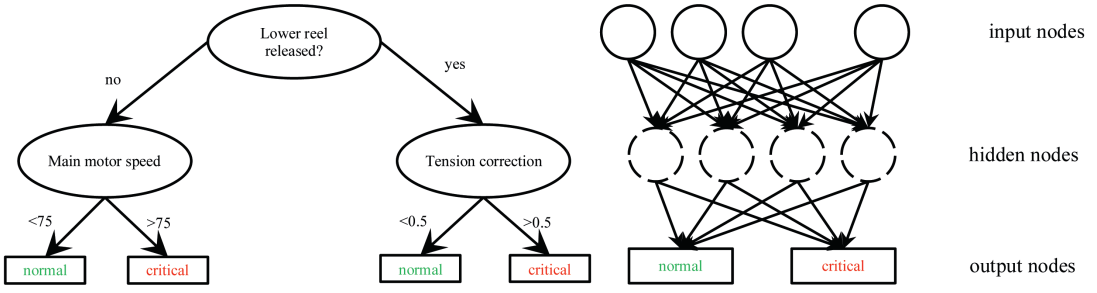

Figure 3: a sample decision tree (left) and neural network (right)

3.3. Experimental Results

This section shows results obtained by testing the classification models on test data. In particular, for each model described in Section 3.2 we evaluated the accuracy in predicting the unknown class. All instances were labeled according to the fact that the machine was working properly (“normal” class) or a fault was present (“critical” class). The dataset used for all models has 5718 total instances (3315 “normal” and 2403 “critical”) and 116 attributes (97 energy variables, 15 from parameters, and 4 resulting from the association rules analysis, see Section 3.1). Since energy variables were sampled every 16 seconds, for each energy variable we considered its average value and variance over three consecutive instances. Moreover, we considered the machine as working correctly only if the

"normal" class is repeated for three consecutive samplings. This reduces the dataset to 1105 instances of "normal" class and 801 instances for the "critical" class.

Performance of each classifier was measured according to the measures described in the following. Since we are using binary classifiers, the evaluation takes into account four basic data: the number *TP* of true positives, *TN* of true negatives (instances that are correctly classified as "normal" or "critical", respectively), *FP* of false positives, and *FN* of false negatives (instances that are incorrectly classified as "normal" or "critical", respectively). From this we compute the measures: accuracy (*a*), precision (*p*), recall (*r*), specificity (*s*), and negative predicting value (*n*) as:

$$a = \frac{TP+TN}{TP+TN+FP+FN}, \qquad p = \frac{TP}{TP+FP}, \qquad r = \frac{TP}{TP+FN}, \qquad s = \frac{TN}{TN+FP}, \qquad n = \frac{TN}{TN+FN} \tag{2}$$

Intuitively, the accuracy measures the overall performance of the classifier, while precision and recall (resp., specificity and negative predicting value) measure the performance on the positive (resp., negative) class only. Results obtained using cross-validation, by using 90% of the data for training and the remaining 10% for testing, are shown in Table 2.

Table 2: performance of used classifiers

Measure\Classifier	Decision tree	Random forest	Neural network
accuracy (*a*)	63%	71%	64%
precision (*p*)	68%	68%	67%
recall (*r*)	68%	95%	74%
specificity (*s*)	56%	86%	58%
negative predicting value (*n*)	56%	38%	50%

Comparing results from the three classifiers we can draw the following conclusions:

- The accuracy of the random forest is slightly better than those obtained for decision trees and neural networks, both exhibiting quite similar results.
- Random forests are able to reach good recall values for both "normal" (*r*=95%) and "critical" (*s*=86%) classes.
- On the other hand, the precision of random forests on the "critical" class is very low (*n*=38%); this means that the probability of false alarm for this classifier is quite high (in 1-0.38=72% of cases, the classifier raises a false alarm).

4. Conclusions

In this paper we have presented an original conceptual framework for the implementation of condition-based maintenance in complex production systems. The framework benefits from the partnership of both automatic data analysis techniques and human knowledge, since machine experts are involved in several steps, for example, to select the most important faults for the machine at hand or the data relevant to the analysis. The use of the framework in a real-world application allowed us to compare the performance of three different data mining techniques for real-time fault prediction. Among the considered classifiers, the decision tree has been selected as the most effective, thanks to its higher precision on predicting stops (44% of false alarm probability) and its interpretability that could help troubleshooting. However, additional work is expected to improve the overall performance of the classifier. In particular, we are working for increasing the frequency of energy data sampling, which has a clear impact on the characteristics required for the on-board machinery and the communication network connecting the machine to the server where data are stored, and are considering alternative classifiers, like a time-series classifier **Errore. L'origine riferimento non è stata trovata.**: Such classifier (see the box named K-NN in Figure 2) would analyze the actual trend of energy variables (instead of their simple average and variance) and predict the current state of the machine as the class of the most similar trend among the ones in the training set. Lastly, the design of a customized real-time process control system that incorporates the most accurate classification rules and enables adjusting the machine's settings on-line, preventing from faults, is one of the expected target of this research project.

References

[1] D. Goyal and B.S. Pabla. Condition Based Maintenance of Machine Tools – A Review. In *CIRP Journal of Manufacturing Science and Technology*, 2015.

[2] R. Manzini, A. Regattieri, H. Pham, and E. Ferrari. Maintenance for Industrial Systems. Springer, 2010.
[3] A. Prajapati, J. Bechtel, and S. Ganesan. Condition Based Maintenance: A Survey. In *Journal of Quality in Maintenance Engineering*, 18(4), 2012.
[4] R. Agrawal, T. Imieliński, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Procs of SIGMOD '93*.
[5] R. Agrawal and R. Srikant Fast Algorithms for Mining Association Rules in Large Databases. In *Procs of VLDB '94*.
[6] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Procs of EDBT '96*.
[7] J.R. Quinlan. Induction of Decision Trees. In "Machine Learning 1", Kluwer Academic Publishers, 1986.
[8] T.K. Ho. The Random Subspace Method for Constructing Decision Forests. In *IEEE TPAMI*, 20(8), 1995
[9] J.A. Anderson. An Introduction To Neural Networks, MIT Press, 1995.
[10] E. Frank, M.A. Hall, and I.H. Witten. The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, 2016.
[11] X. Xi, E. Keogh, C. Shelton, et al. Fast Time Series Classification Using Numerosity Reduction. In *Procs of ICML '06*.