Feature Selection for Multi-Label Classification in Predictive Maintenance

Preprint · April 2024

DOI: 10.13140/RG.2.2.32904.23049

4 authors:

Antoine Hubermont
University of Namur
4 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Aymeric Vellinger
University of Namur
5 PUBLICATIONS **1** CITATION
SEE PROFILE

Nemanja Antonic
University of Namur
6 PUBLICATIONS **1** CITATION
SEE PROFILE

Elio Tuci
Middlesex University, UK
139 PUBLICATIONS **2,639** CITATIONS
SEE PROFILE

Feature Selection for Multi-Label Classification in Predictive Maintenance

Antoine Hubermont[1][2], Aymeric Vellinger[1], Nemanja Antonic[1], and Elio Tuci[1]

[1] University of Namur,
antoine.hubermont@unamur.be,
[2] Telespazio Belgium, B-6890 Transinne, Belgium

Abstract. In this study, we propose a novel feature selection technique to improve the classification of simultaneous anomalies in the context of predictive maintenance. We leverage three publicly available datasets collected from measurements and characterised by high dimensions. In this context, the use of machine learning algorithms with 5 multi-label classifiers has proven effective in anomaly classification tasks. Therefore, we implement a self-adaptive evolutionary strategy algorithm tasked with selecting the most relevant features. We assess the cost and the effectiveness of the classification task with and without feature selection. Overall, our results show that the self-adaptive evolutionary strategy is able to drastically reduce the number of features required for training, while improving or maintaining the performance of the classification task. Our study underscores the potential of feature selection for predictive maintenance multi-label classification tasks, contributing to more efficient and effective predictive maintenance strategies in industrial settings.

Keywords: Evolutionary strategy, Multi-label classification, Feature selection, Predictive maintenance, Machine learning

1 Introduction

Maintenance is a crucial topic in industry and has led to the extensive deployment of sensors for monitoring a range of processes, especially those related to production lines [10]. Advances in science and technology ease the use of algorithms based on artificial intelligence to support the monitoring and enable the development of adaptive and optimized maintenance strategies [8]. Predictive maintenance (hereafter PdM) optimizes maintenance strategies to reduce costs and enhance reliability by preventing both the unnecessary replacement of functioning components and the occurrence of service disruptions through anomaly classification. Two approaches emerge from [22]: the prognostic approach, which focuses on the estimation of the remaining useful life of components, and the diagnostic approach, which focuses on the detection and forecasting of anomalies. The diagnosis approach typically involves single-label classification with anomaly detection, which inherently assumes that the detected anomalies are mutually exclusive. However, in PdM, machines or components can exhibit multiple issues concurrently, naturally shifting the classification problem to a multi-label classification problem (hereafter MLC) with dedicated machine-learning classifiers. An example can be found in [7] where the authors compared single-label classification with multi-label classification on the Microsoft *Azure* Predictive Maintenance dataset [1].

A further challenge in implementing PdM systems is managing the data, typically gathered from a diverse network of sensors recording at cyclic intervals. This data, defined as multivariate time-series, includes various features to train classifiers. These features includes sensor types, monitored components, the timing of data capture and the previous data in the sequences. The implementation of PdM systems produces large and complex datasets. It is crucial to simplify this complexity by selecting the most relevant features through feature selection techniques. Feature selection effectively reduces the dimension of the input space, aiming to retain only the most relevant features, simplifying the overall complexity and reducing the computational resources needed to train classifiers [11]. To implement feature selection, two main methods are suggested [9]: the filter method and the wrapper method. Wrapper methods aim at optimizing a learning algorithm during the selection process, whereas filter methods rely on the intrinsic characteristics of the training data to select features, without any specific learning algorithm. In the field of feature selection on time-series, evolutionary strategies (hereafter ES) are used for forecasting tasks on uni-variate time-series [16, 21]. In the following, we use a self-adaptive evolutionary strategy ($SA-ES$) using a classifier as learning algorithm, suited for multi-label feature selection that aims at reducing the computational cost of the wrapper methodology by dynamically adapting the number of candidate solutions generated in each generation of the evolutionary strategy.

To show the effectiveness of our algorithm on MLC problems, we compare 5 multi-label classifiers trained on 3 datasets belonging to the PdM field with and without feature selection. We finally demonstrate an improvement in anomaly classification with the classifiers trained on a reduced number of features. The paper is organised as follows. Section 2 presents our self-adaptive evolutionary strategy. Section 3 describes the classifiers and MLC chosen approaches, the evaluation objectives and finally the datasets considered for this study. Section 4 details the evaluation results of classifiers trained with and without features selected by $SA-ES$. Finally, in Section 5 we present the conclusion of the study.

2 Self-Adaptive Evolutionary Strategy

In this study, we define a Self-Adaptive Evolutionary Strategy ($SA-ES$), which is part of the family of $(\mu + \lambda) - ES$ algorithms.

Each individual corresponds to a dataset features represented by an array of binary numbers. The length of each array matches the total number of features, where a '1' indicates that a feature is selected and a '0' indicates it is not. At each generation, corresponding to an iteration in the evolutionary process, we apply a bit-mask crossover, which involves selectively combining bits from two parent solutions based on a randomly generated binary mask. We then apply a bit flip mutation, which select bits within the individual and flipping their values. The mutation probability for each individual at each generation is updated considering the fitness function performance at the previous generation. Distinctive parameters are introduced to dynamically adjust the offspring size λ_g), representing the initial number of candidates solutions per generation, and the mutation rate (α_g) at each generation (g). This dynamic adjustment seeks to find a trade-off between exploration and exploitation during the evolutionary process, hence reducing the computational cost induced by the wrapper approach in multi-label feature selection.

During selection, a proportion of the offspring, referred to as the elite, is unvaried and transferred to the next generation. The size of the elite is determined by the parameter $\mu \in [2, \lambda]$. The $SA - ES$ operates at every generation g according to a logic based on the adaptation of the number of offspring λ_g and the step size α_g. This adaptation takes into account the fitness of the best solutions in the previous m generations, according to the following equations:

$$\lambda_g = \frac{t_g + \lambda_r^2}{t_g + \lambda_r}\lambda_{\max}; \quad \lambda_r = \frac{\lambda_{\min}}{\lambda_{\max}}; \quad \alpha_g = \begin{cases} c\alpha_0 t_{g-1} & \text{if } \gamma_g = 0 \\ \alpha_0 t_g & \text{otherwise} \end{cases} \tag{1}$$

where

$$\forall x \in \text{population}_g : \quad \tilde{f}_g = \min_x f(\mathbf{x}); \qquad \text{Given that } \forall m \in \mathbb{N}, \; g \geq m + 2$$

$$t_g = \begin{cases} t_{g-1} & \text{if } \gamma_g = 0 \\ \frac{m}{\gamma_e}\gamma_g & \text{otherwise} \end{cases}; \quad \gamma_e = \sum_{i=g-m+2}^{e} \gamma_i; \quad \gamma_i = \left|\tilde{f}_{i-1} - \tilde{f}_{i-2}\right| \forall i \in [2, g] \tag{2}$$

where $\alpha_0 \in [0, 1]$ is the initial value for the mutation rate, $c \geq 1$ represents the mutation boost factor, γ_i denotes the fitness improvement at generation i and the parameters $\lambda_{\min}$ and $\lambda_{\max}$ refer respectively to the minimum and the maximum allowed number of offspring. Intuitively, we can note that t_g tends to t_{g-1} as $g \to +\infty$, since γ_i tends to 0, as a consequence of elitism, which ensures that:

$$\left|\tilde{f}_i\right| \leq \left|\tilde{f}_{i+1}\right| \forall i \in [0, g - 1] \tag{3}$$

Thus, the number of offspring converges to a constant value bounded by $\lambda_{\min}$ and $\lambda_{\max}$ for $e \to +\infty$ since $t_g \to t_{g-1}$.

3 Experiments

In the following section, we outline the experimental setup designed to classify anomalies on PdM datasets with dedicated MLC classifiers. Numerous multi-label classifiers are discussed in the literature, and their effectiveness varies depending on the task [3].

3.1 Classifiers

We select the classifiers following the systematic review of PdM machine-learning classifiers in [5] and multi-label classifiers in [3] to cover the two MLC approaches defined in [18]: problem transformation and algorithm adaptation. Problem transformation shifts the multi-label classification problem into a set of single-label classification problems. Algorithm adaptation allows to modify the structure of the classification algorithms to support the detection of multiple labels. We choose four Ensemble of Problem Transformation classifiers as defined in [3] which combine techniques from problem transformation and aggregate their results via embedding, bagging and voting systems. These classifiers are Ensemble of Binary Relevance [14, 18], Classifier Chain [14], Pruned Set [13] and Random k Labelsets [20] (respectively hereafter BR, CC, PR and $RAkEL$). These aim to minimise the drawbacks of the original problem transformation methods such as information loss by ignoring label dependency or the lack of generalisation.

First, BR treats each class individually with a binary classifier and issues final predictions with bagging. CC structures binary classifiers in an ordered chain where each classifier benefits from predictions of previous classifiers in the chain. PR selects and removes infrequent classes from the training set and reintroduces them with more frequent classes. Finally, $RAkEL$ is an ensemble of LP which treats each unique label combination as a new label. Then, it is trained on the original dataset randomly divided on k sub-sets.

As algorithm adaptation method, we choose Multi Label K Nearest Neighbour [23] (hereafter $MLkNN$). Following the same logic as Single-label kNN, $MLnNN$ assigns labels by looking at the most represented labels amongst the nearest neighbours of an instance.

As recommended in [3], we select $J48$ decision tree as a base classifier for its general better performance without the need of tuning parameters. Regarding the implementations and parameters for the classifiers, we choose to keep default parameters available on their respective implementations accessed with the library *scikit multilearn* [17] and its $MEKA$ wrapper [15].

3.2 Evaluation

To evaluate the classifiers, we choose the Hamming Loss defined in Equation 4. Hamming Loss a widely used metric in MLC tasks to quantify the classification results and give the proportion of misclassified labels amongst all the possible labels where 0 is equivalent to a perfect score. We aim to improve the classification results by minimizing the Hamming Loss and diminishing the information required to classify anomalies by selecting the most interesting features.

$$\text{HammingLoss}(H, N) = \frac{1}{N}\sum_{i=1}^{|N|}\frac{|Y_i|\Delta|Z_i|}{|C|}; \tag{4}$$

where H is a multi-label classifier, N is the test set (i.e. set of instances), Y represents the set of class labels in N, Y represents the predictions computed by classifier H, C is the number of available classes and Δ is the symmetric difference between the test labels and predictions (xor operation). The experiments are carried with a $5 - fold$ validation to minimize the impact of the instance splits inside the train and test sets except for $Azure$ where we choose a $3 - fold$ validation due to computational time limits.

3.3 Datasets

We use three different datasets related to predictive maintenance and suitable for multi label classification for our comparative study: $AI4I2020$, Microsoft Azure Predictive Maintenance (hereafter $Azure$) and Condition monitoring of hydraulic systems (hereafter $Hydraulic$). Table 1 references the common metrics used in MLC tasks to reflect the different meta characteristics of the datasets. The first dataset is the $AI4I2020$ dataset developed by [12]. This set of data has been specifically created to evaluate algorithms for predictive maintenance problems. The dataset contains 10.000 instances referring to 7 distinctive features from milling machines.

The second dataset is the $Azure$ dataset developed for the collection "Predictive Maintenance Modelling Guide" from the Microsoft $Azure$ suite using synthetically-generated data [1]. It is composed of 100 machines each one monitored by 4 sensors (i.e., sensors capturing vibration, rotation, pressure and voltage). Each sensor is recorded every hour during one year. The dataset is furnished with troubleshoot records containing information about errors (recoverable) and failures (not recoverable) that could impact machine components. Hourly monitoring introduces noise in the recorded data. To mitigate the noise, we apply feature engineering techniques. We incorporate past information by shifting features in time in the dataset, resulting in the creation of lagged features as in [7] and [4]. The time used to create the lagged features is a parameter of the model called lagging window, which in this study is set to 24h. Moreover, for each sensor, two lagged features are created using a moving average (avg), and a moving standard deviation (std). We have also incorporated the following information: the time since the last maintenance operation on each component (in hours), the number of errors (and their type) occurred in the last n hours (where n is defined by the lagging window) and the machine service age. The logic behind this choice is to let the $SA - ES$ algorithm select the features that allow the failure classification algorithm to accomplish its task to the best of its possibility. Similarly to what we did with the $AI4I2020$ dataset, labels are tailored for simultaneous failure classification. Labels are vectors encoding the component name to 1 if it is impacted by a failure in the following 24h, or to 0 otherwise.

The third dataset is $Hydraulic$ publicly available on the UC Irvine Machine Learning Repository [6]. The data is a collection of experiments recorded on a hydraulic test rig monitored by 6 sensors for pressure (PS), 4 for temperature (TS), 1 for motor power (EPS) and 1 for vibration (VS). Additional information are provided such as cooling efficiency (CE), cooling power (CP)

and efficiency factor (SE). Each experiment is labeled by the condition states of the rig components: cooler, valve, internal pump, hydraulic accumulator. In the original dataset, each component has multiple condition states depending on the degradation stage. For our methodology we divide the condition state in two classes, class 0 if the condition is optimal and class 1 if the condition is degraded. The goal is to detect potential degradation in each experiment. Sensors have different monitoring time (ranging from $1Hz$ to $100Hz$) and we average the collected data to obtain a constant monitoring frequency of one measure per second.

The three datasets are initially described using typical single-label classification metrics such as the number of instances, features, and labels. However, since instances in these datasets can be assigned to multiple labels, specialized statistics are necessary to adequately describe label frequency and sparsity. Cardinality, as outlined in Equation 5, measures the average number of labels per instance, while density, detailed in Equation 6, represents the average number of labels per instance relative to the total number of available labels [19].

Both cardinality and density significantly influence classifier training [2]. High cardinality in a dataset means that classifiers need to handle many labels simultaneously per instance, indicating a complex labeling environment. Conversely, low density suggests that each instance is labeled with only a few of the possible labels, highlighting the challenge of predicting less frequent anomalies effectively.

$$\text{Cardinality} = \frac{1}{D}\sum_{i=1}^{D} |Y_i|; \tag{5}$$

$$\text{Density} = \frac{1}{D}\sum_{i=1}^{D} \frac{|Y_i|}{|C|}; \tag{6}$$

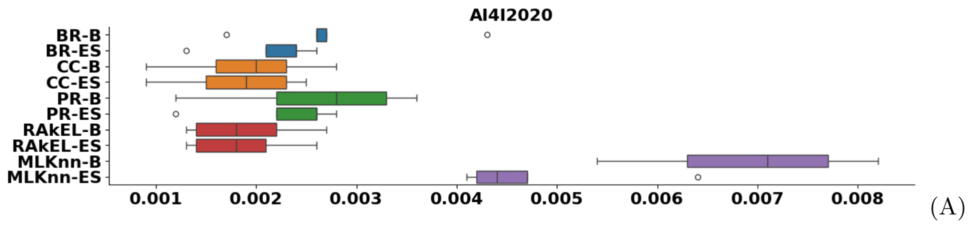
where D is the number of instances in a dataset, Y is the label set and C the set of available classes.

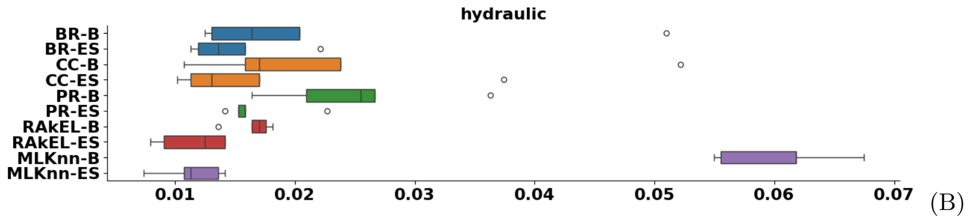
Table 1. Datasets meta characteristics

	AI4I2020	*Hydraulic*	*Azure*
Instances	10000	2205	291300
Features	7	17	27
Labels	5	4	4
Cardinality	0.037	2.328	0.02
Density	0.007	0.582	0.005

4 Results

We train MLC classifiers as described in 3.1 using the three datasets outlined in Section 3.3 to evaluate the effects of $SA - ES$ on both the Hamming Loss





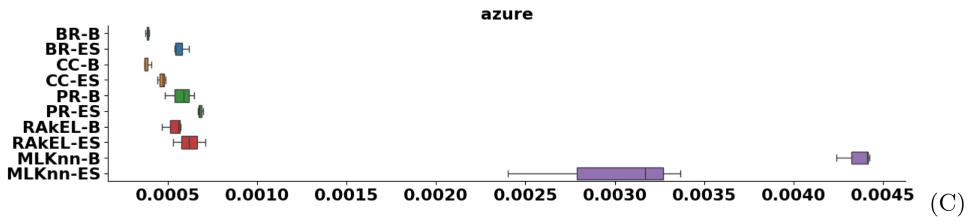


Fig. 1. Comparison of Hamming Loss for classifiers trained on baseline datasets (marked with a B extension) versus those trained using features selected by $SA - ES$ (marked with ES extension), across the following datasets: $AI4I2020$ (A), $Hydraulic$ (B), and $Azure$ (C).

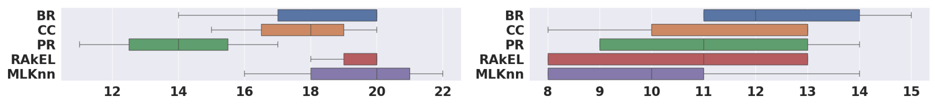


Fig. 2. Comparison of the number of features used by classifiers trained with $SA - ES$ selected features across the $Azure$ (left) and $Hydraulic$ (right) datasets.

and the number of features selected to train the classifiers. Figure 1 shows the Hamming Loss for classifiers trained on the baseline dataset (marked with the B extension) and those trained using features selected by $SA - ES$ (marked with the ES extension). For the $Hydraulic$ and $AI4I2020$ datasets, regardless of the MLC method used, classifiers trained with $SA - ES$ feature selection exhibit a lower Hamming Loss compared to those trained on the baseline dataset, indicating that our approach enhances classification performance. Conversely, on the $Azure$ dataset, $SA - ES$ feature selection does not lead to an improvement in Hamming Loss compared to the baseline, and it slightly increases it. This issue could be explained with a nearly perfect Hamming Loss achieved by the baseline classifiers, limiting the impact of feature selection on the scores.

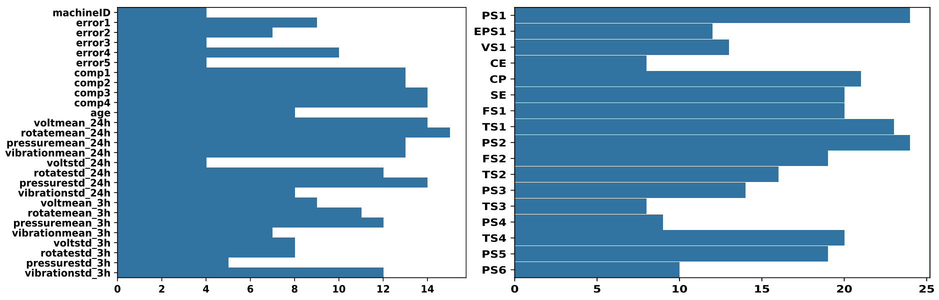


Fig. 3. Comparison of features frequency selection for classifiers utilizing $SA - ES$ selected features across the $Azure$ (left) and $Hydraulic$ (right) datasets.

Despite not improving Hamming Loss on the $Azure$ dataset, our approach significantly reduces the number of features required to train classifiers as illustrated in Figure 2. In $Azure$, PR is trained using just 14 features, while other classifiers are trained with a number of features between 18 and 20, in contrast to the 27 features used in the original baseline $Azure$ dataset. Similarly, in the $Hydraulic$ dataset, a noticeable feature reduction is observed, with classifiers trained using 10 to 12 features achieving lower Hamming Loss than those trained with 17 baseline features.

For the $AI4I2020$ dataset, the impact of $SA - ES$ feature selection is less pronounced due to the already low number of features in the baseline dataset, with selected features ranging from 4 to 6.

Lastly, it is worth to note the consistency of our approach in the features selected by $SA - ES$. Figure 3 illustrates the frequency at which each feature is chosen by $SA - ES$ across all classifiers for both the $Hydraulic$ and $Azure$ datasets. The sporadic selection of certain features, such as mean vibration over the last 24 hours for $Azure$ or CE for $Hydraulic$, does not imply that these sensors should be removed from the monitoring strategy. Rather, it offers valuable insights into the critical aspects of the system and shows a consensus amongst the MLC approaches on the significance of frequently selected features like PS1 for $Hydraulic$ and mean voltage of the last 24 hours for $Azure$.

5 Conclusions

In this study, we investigated the effectiveness and importance of feature selection in multi-label classification problems for predictive maintenance (PdM). We introduced a self-adaptive evolutionary strategy ($SA - ES$) leveraging a dynamic approach which adjusts the number of offspring and the mutation rate at each generation. This strategy lowers the computational cost induced by the learning approach taken to select features. We implemented 5 different classifiers, namely Ensemble of Binary Relevance, Classifier Chain, Pruned Set, Random k Labelsets and MLkNN, using the Hamming Loss metric as a fitness function. As a result, significant reductions in the number of features were observed across all datasets, indicating the effectiveness of the $SA-ES$ approach in identifying relevant features for PdM tasks. While the improvement in Hamming Loss was consistent for $AI4I2020$ and $Hydraulic$ datasets, the $Azure$ dataset exhibited slight fluctuations, possibly due to the already high performance of classifiers trained on the baseline dataset. Furthermore, the consistency in feature selection regardless of the classifiers suggests a consensus on the relevance of certain features in the PdM context, providing valuable insights into critical aspects of the monitored systems.

Overall, our study underscores the potential of feature selection for PdM multi-label classification tasks, contributing to more efficient and effective predictive maintenance strategies in industrial settings.

6 Acknowledgments

The authors acknowledge the SPW-recherche (Walloon Region) and Telespazio Belgium throughout the research funding WIN4DOC under Grant No. 2110246 and NAXYS. Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region. To Robert, for the bagels and explaining CMYK and color spaces.

References

1. Predictive Maintenance Modelling Guide Data Sets. URL https://gallery.azure.ai/Experiment/Predictive-Maintenance-Modelling-Guide-Data-Sets-1
2. Bernardini, F., Silva, R., Rodovalho, R., Mitacc Meza, E.: Cardinality and density measures and their influence to multi-label learning methods. Learning and Nonlinear Models **12**, 53–71 (2014). DOI 10.21528/LNLM-vol12-no1-art4
3. Bogatinovski, J., Todorovski, L., Džeroski, S., Kocev, D.: Comprehensive comparative study of multi-label classification methods. Expert Systems with Applications **203**, 117,215 (2022). DOI 10.1016/j.eswa.2022.117215
4. Cardoso, D., Ferreira, L.: Application of Predictive Maintenance Concepts Using Artificial Intelligence Tools. Applied Sciences **11**, 18 (2020). DOI 10.3390/app11010018
5. Carvalho, T., Soares, F., Vita, R., Francisco, R., Basto, J.a., G. Soares Alcalá, S.: A systematic literature review of machine learning methods applied to predictive maintenance. Computers & Industrial Engineering **137**, 106,024 (2019). DOI 10.1016/j.cie.2019.106024
6. Helwig, N., Pignanelli, E., Schtze, A.: Condition monitoring of hydraulic systems. UCI Machine Learning Repository (2018). DOI: https://doi.org/10.24432/C5CW21
7. Hubermont, A., Tuci, E., De Quattro, N.: Simultaneous failures classification in a predictive maintenance case. In: ESANN 2023 proceesdings, pp. 537–542. Ciaco - i6doc.com (2023). DOI 10.14428/esann/2023.ES2023-129

8. Kanawaday, A., Sane, A.: Machine learning for predictive maintenance of industrial machines using IoT sensor data. In: 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 87–90 (2017). DOI 10.1109/ICSESS.2017.8342870. ISSN: 2327-0594
9. Kashef, S., Nezamabadi-pour, H., Nikpour, B.: Multilabel feature selection: A comprehensive review and guiding experiments. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8**, e1240 (2018). DOI 10.1002/widm.1240
10. King, R., Curran, K.: Predictive Maintenance for Vibration-Related failures in the Semi-Conductor Industry. Journal of Computer Engineering & Information Technology, **8**(1), 1 (2019). DOI 10.4172/2324-9307.1000215
11. Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R.P., Tang, J., Liu, H.: Feature Selection: A Data Perspective. ACM Computing Surveys **50**(6), 1–45 (2018). DOI 10.1145/3136625
12. Matzka, S.: Explainable Artificial Intelligence for Predictive Maintenance Applications. In: 2020 Third Int. Conf. on Artificial Intelligence for Industries (AI4I), pp. 69–74. Institute of Electrical and Electronics Engineers (2020). DOI 10.1109/AI4I49448.2020.00023
13. Read, J., Pfahringer, B., Holmes, G.: Multi-label classification using ensembles of pruned sets. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 995–1000 (2008). DOI 10.1109/ICDM.2008.74
14. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning **85**(3), 333–359 (2011). DOI 10.1007/s10994-011-5256-5
15. Read, J., Reutemann, P., Pfahringer, B., Holmes, G.: MEKA: A multi-label/multi-target extension to Weka. Journal of Machine Learning Research **17**(21), 1–5 (2016). URL http://jmlr.org/papers/v17/12-164.html
16. Siddiqi, U.F., Sait, S.M., Kaynak, O.: Genetic algorithm for the mutual information-based feature selection in univariate time series data. IEEE Access **8**, 9597–9609 (2020). DOI 10.1109/ACCESS.2020.2964803. Conf. Name: IEEE Access
17. Szymański, P., Kajdanowicz, T.: A scikit-based Python environment for performing multi-label classification. ArXiv e-prints (2017)
18. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. IJDWM **3**, 1–13 (2007)
19. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining Multi-label Data. In: O. Maimon, L. Rokach (eds.) Data Mining and Knowledge Discovery Handbook, pp. 667–685. Springer US, Boston, MA (2010). DOI 10.1007/978-0-387-09823-4_34
20. Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-labelsets for multilabel classification. IEEE Transactions on Knowledge and Data Engineering **23**(7), 1079–1089 (2011). DOI 10.1109/TKDE.2010.164
21. Vellinger, A., Torres, J., Divina, F., Vanhoof, W.: Neuroevolutionary transfer learning for time series forecasting. In: Int. Conf. on Soft Computing Models in Industrial and Environmental Applications, pp. 219–228. Springer (2023)
22. Wen, Y., Rahman, M., Xu, H., Tseng, B.: Recent Advances and Trends of Predictive Maintenance from Data-driven Machine Prognostics Perspective. Measurement **187**, 110,276 (2021). DOI 10.1016/j.measurement.2021.110276
23. Zhang, M.L., Zhou, Z.H.: A k-nearest neighbor based algorithm for multi-label classification. In: 2005 IEEE International Conference on Granular Computing, vol. 2, pp. 718–721 Vol. 2 (2005). DOI 10.1109/GRC.2005.1547385