Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 291–296
DOI: 10.15439/2023F677
ISSN 2300-5963 ACSIS, Vol. 35

Towards Industry 4.0: Machine malfunction prediction based on IIoT streaming data

Dragana Nikolova*, Petre Lameski*, Ivan Miguel Pires†, Eftim Zdravevski*
*Faculty of Computer Science and Engineering,Ss Cyril and Methodius University, Skopje, Macedonia
Email: dragana.nikolova.1@students.finki.ukim.mk, petre.lameski@finki.ukim.mk, eftim.zdravevski@finki.ukim.mk
†Instituto de Telecomunicações, Universidade da Beira Interior, Covilhã, Portugal
Email: impires@it.ubi.pt

***Abstract*—The manufacturing industry relies on continuous optimization to meet quality and safety standards, which is part of the Industry 4.0 concept. Predicting when a specific part of a product will fail to meet these standards is of utmost importance and requires vast amounts of data, which often are collected from variety of sensors, often reffered to as Industrial Internet of Things (IIoT). Using a published dataset from Bosch, that describes the process at every step of production, we aim to train a machine learning model that can accurately predict faults in the manufacturing process. The dataset provides two years of production data across four production lines and 52 stations. Considering that the data generated from each production part includes more than four thousand features, we investigate various feature selection and data preprocessing methods. The obtained results exhibit Area Under the Receiver Operating Characteristic Curve (AUC ROC) of up to 0.997, which is remarkable and promising even for real-life production use.**

***Index Terms*—Industrial Internet of Things, Industry 4.0, Machine malfunction prediction, Machine failure prediction**

I. Introduction

The ongoing progression of the fourth industrial revolution, accompanied by a fundamental shift towards digitization, referred to as Industry 4.0, is advancing at an exponential rate [1].

In Industry 4.0 systems, the goal is to utilize different temperature sensors, pressure sensors, audio sensors, camera devices, etc., as Industrial Internet of Things (IIoT) devices for machine monitoring and operation control in industrial environments [2]. However, performing machine fault diagnosis and failure prediction is challenging, especially considering the explainability and interoperability requirements.

Introducing predictive maintenance to production environments can provide many benefits, albeit with a few challenges. Some benefits include heightened productivity, decreased system faults, minimized unplanned downtimes, optimized utilization of financial and human resources, and improved planning of maintenance interventions as stated in [3] [4]. In addition, employing machine learning is an effective means of accomplishing prognostics and predicting failures [5]. Predicting when a part of a product will fail is paramount in identifying and preventing defects, thereby improving product quality and safety [6]. By leveraging data generated from each production part, manufacturers can determine whether a part has a weakness and take appropriate action.

Bosch, a leading manufacturer, recognizes this need and has started recording data at every step of the production process. In 2016, they published an anonymized dataset on Kaggle that provides valuable insights into two years of production data across four production lines and 52 stations [1]. Each workstation in the production process performs a variable number of tests and measurements on each part, generating 4,264 features. This experiment aims to train a machine learning model that can accurately predict faults in the manufacturing process using this dataset.

Given the vast number of features in the dataset, data preprocessing and feature selection is a critical step in the model development process [7], [8]. The enormous data growth requires using big data architectures for efficient, robust, and timely processing of it [9]. In turn, it requires the use of efficient algorithms for optimizing hardware resources and minimizing computational cost [10].

In this paper, we perform analyses and feature extraction techniques for numerical, date, and categorical data types to ensure only the most relevant data are used to train the model. With this approach, we aim to create a highly accurate machine-learning model that can aid the manufacturing industry in predicting faults in the manufacturing process using a training dataset where 6,879 parts out of 1,183,747 were labeled as failed, which is a 0.58 error rate. This relatively low error rate presents a significant challenge in creating an accurate predictive model. Figure 1 shows the number of failed parts per line on the left and the number of failed parts per station, indicating that station S32 has a significantly higher number of failures than the rest.

We aim to solve a classification problem to predict whether a product part will fail to meet quality and safety standards during manufacturing. To achieve this, we trained separate machine learning models such as Random Forest, Decision Tree, GradientBoostingClassifier, AdaBoostClassifier, and XGBoost. After evaluating the performance of each model, we found that the XGBoost model had the highest accuracy in predicting faults. Therefore, it was chosen as the final model for the task.

This paper is organized as follows. Section II extensively reviews the machine-learning approaches used for machine

[1] https://www.kaggle.com/c/bosch-production-line-performance

malfunction prediction. In Section III, we introduce our proposed method and provide a detailed explanation of the approach that we are using. Section IV describes the dataset used in our study and presents the results obtained from our experimental analysis. Finally, in Section V, we summarize our findings and conclusions from our research, discussing the potential impact of our work on the manufacturing industry.

II. Related Work

The foundation of this study lies in the preprocessing and feature extraction of numerical and time series data, as well as in classification for malfunction prediction. In the following, we delve into related research on these topics.

A paper focuses on methods for fault prediction [11] and using raw sensor data elaborates on the differences between the Support Vector Machine (SVM) and the Multilayer Perceptron (MLP) for fault prediction. This paper presented an initial development of a supervised machine learning algorithm for diagnosing faults in rotating machinery in the oil and gas industry. They aim to create a simple, easily implementable model that enables quick, informed decision-making. Some preprocessing steps explained in the study are filling in the missing values using linear interpolation, feature engineering performed to introduce the correlation between a data sample and preceding samples in chronological order, and data related to downtime and start-up periods filtered out. As a result, the SVM algorithm demonstrated higher precision than MLP but lower recall for the positive class.

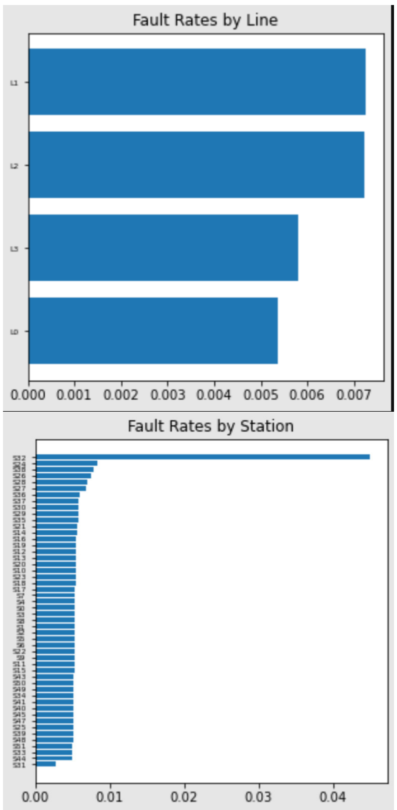

Fig. 1. (above) Number of failed parts by a production line in the training dataset (below) Number of failed parts per production station in the training dataset

[12] discusses the problem of hardware failures in circuits due to aging or variations in circumstances. While self-healing and fault tolerance techniques can recover circuitry from a fault, fault prediction can be used as a pre-stage to these techniques. The proposed method for early fault prediction of circuits uses Fast Fourier Transform, Principal Component Analysis, and Convolutional Neural Network to learn and classify faults. The approach was validated by testing it on two different circuits (comparator and amplifier) using 45 nm technology, providing a fault prediction accuracy of 98.93% and 98.91%, respectively.

Not only hardware failures but, in an article from [13], software failures were also analyzed. The quality of software depends on its bug-free operation, and identifying bugs in the early stages of development can reduce the cost of testing and maintenance. Software defect prediction models can identify bugs before release using historical data from software projects for training. The study used software change metrics for defect prediction, and the performances of machine learning and hybrid algorithms were compared. This study uses different machine learning techniques to create defect prediction models, including Random Forest, Multilayer Perceptron, Fuzzy-AdaBost, and Logitboost. With Logitboost, was reached the best accuracy.

Focusing on the feature extraction part, [14] proposes new damage classifiers for locating and quantifying damage based on a supervised learning problem. A new feature extraction approach using time series analysis is introduced to extract damage-sensitive features from auto-regressive models. The coefficients and residuals of the AR model obtained from this approach are used as the main features in the proposed supervised learning classifiers, which are categorized as coefficient-based and residual-based classifiers. These classifiers are validated using experimental data for a laboratory frame and a four-story steel structure. They are shown to be able to locate and quantify damage, with the residual-based classifiers yielding better results than the coefficient-based classifiers. Furthermore, comparative analyses show that these methods are superior to some classical techniques.

The following related work uses the same dataset in our paper and solves a classification problem for faulted parts. The authors of [15] used the Bosch dataset uploaded on Kaggle. First, they trained a model that predicts which parts are most likely to fail. Then, to manage many categorical features, they employed the FTRL(Follow The Regularized Leader) algorithm to train a model using solely categorical features. Afterward, they stacked the probability predictions with numerical and date features as a new column. This technique serves as a means of reducing features, in which all the categorical features are condensed into one feature column. The top 200 features were used to train an XGBoost model on the entire training dataset, which consists of approximately 1 million samples. The training data were randomly divided into three subsets, each containing 33% of the data. Three separate XGBoost models with the same hyperparameters were trained on 67% of the data and evaluated on the remaining 33% of the data.

Authors in [16] propose a systematic feature engineering and selection methodology considering data from a variety of sensors. From the originally recorded time series and some newly generated time series, a variety of time and frequency domain features are extracted and then selected. Such approaches can be also used in the machine malfunction problems where there is abundancy of time-series data originating from IoT devices and sensors.

The main focus in many related papers is typically on data preprocessing, and this was also our primary focus. Our work dealt with numerical features and extracted additional features from the date values, emphasizing capturing the time dependence between different parts. For example, in [15], the feature extraction was made directly on the features combined, while we deal with numerical and date features separately with different techniques for each. However, we also devoted significant attention to analyzing the raw data, as the time series data provided by Bosch were anonymized and normalized, which made it necessary to create estimations of the duration in the time series data. We were able to leverage this information to develop additional features based on the date values.

Additionally, we paid careful attention to null values during feature extraction, given the large number of numerical values we were dealing with (970 in total). Finally, we addressed the challenge of imbalanced data by performing downsampling and oversampling. We use a gradient-boosted decision tree as our primary model for malfunction prediction. This robust algorithm can handle complex data structures and identify the most valuable features for predicting the desired outcome.

III. METHODS

The primary objective of this paper is to develop a fault prediction model using the Bosch dataset. The dataset was released by Bosch in 2016 as a challenge to improve its future defect reduction efforts. Furthermore, we aim to solve a classification problem using the dataset. We trained different models separately, Random Forest, Decision Tree, GradientBoostingClassifier, AdaBoostClassifier, and XGBoost. After evaluation, we found that the XGBoost model produced the highest accuracy.

The numerical data in the dataset contains many zero values, with 929,125,166 fields or 80.91% of the total being empty. However, this is not surprising, as each part goes through a specific set of stations and measurements, and empty cells indicate that a part did not undergo a particular measurement at a certain station or line. As such, these zero values are not considered missing data, and filling them using standard methods like mean imputation or forward/backward filling is inappropriate.

The dataset comprises 970 numerical columns, including the Id and Response columns. There are no columns with all null values, so such columns cannot be dropped. However, 227 columns have 99% null values, and their relevance must be determined to decide whether to keep or discard them. Upon closer inspection, we found that 11 of these columns have non-zero values for parts that were not classified as failed, and these can be immediately removed. We calculated Pearson correlation coefficients between the 11 columns and the Response variable to ensure these columns are irrelevant. That indicated correlation values close to zero, meaning they are not significantly correlated with the outlier parameter and can be safely removed.

Our next step is to use the XGBoost model to identify the most and least significant numerical features to reduce the set of features. First, we ran the model on the remaining 227 columns from the previous step and found that 107 features had a significance of 0.005 or less, so we removed them. This process left us with 852 numerical features.

Next, we applied the XGBoost model to the remaining 852 columns and removed those with more than 90% null values and a significance of 0, resulting in the removal of 218 columns. It brought the total number of numerical features down to 634, reducing the percentage of empty fields to 47.58% from the initial 80.91%.

Of the remaining 534 columns, 100 had a significance greater than 0 and more than 90% non-zero values, so we used them directly in the model. For the remaining 434 columns, we found that, on average, 49 columns had over 50% non-zero values. Therefore, for each of these 49 columns, we merged 10 zero columns and used them in the combining process, where the first non-zero value is taken. As a result of these steps, we ended up with 149 numerical features and 13.58% null fields.

Moving next to the date features, we have extracted some features from the existing date features, and then we have added additional date features with a focus on time dependence between parts.

To include time dependence, we added the following time domain features:

1) Number of parts in one takt (6 minutes). To calculate the number of parts that pass through the measuring stations in one takt, we look for the consecutive parts with the same starting takt where the first column is the takt and the second column is the number of parts that pass in the same takt.
2) Number of failures in the next 1, 10, 24 hours
3) Number of failures in the last 1, 10, 24 hours

The date features represent the date and time the measurements were taken. These features can be important because they capture temporal patterns and trends that may be relevant for predicting the target variable.

Since the test and training data are consecutive parts with indices from 1 to 2,367,494, the specified features are appropriate for the training and unlabeled data.

Of the 4,258 categorical features, 1,913 are duplicates and will be removed. Among the remaining ones, 1,549 have a single value, and 428 have multiple values. Any empty feature will also be discarded. Categorical feature values are represented as classes denoted by T followed by a number, which labels different processes. For instance, column L1_S24_F1269 contains four classes: 'T1372', 'T618624', 'T83888', and 'T8389632'. After removing duplicates and empty features, we use one-hot encoding to represent each category with an integer. One-hot encoding transforms a single variable with d distinct values into d binary variables, where each observation indicates a particular binary variable's presence (1) or absence (0). This results in a vector of size 988 for each row, but since most values are zero, we end up with a sparse matrix. To avoid overfitting, we will compare the performance of the model with and without categorical features.

Moreover, we will reduce the dimensionality of the resulting matrix using a dimensionality reduction algorithm. Sparse PCA is an unsupervised learning method used in statistical analysis to identify sparse features that can reconstruct the data. We will replace the 988 features with 5 features obtained from Sparse PCA.

After processing the features, we have 1,183,747 rows and 173 features from the training set, of which 19 are date, 149 are numeric, and 5 are categorical.

Classification models aim to assign data to different classes, but in an unbalanced dataset, one class may have a much larger number of samples than the other classes. It creates a majority class and a minority class, which can be problematic during model training because the model may not learn enough about the minority class. In our case, the defect class is the minority class, with only 6,879 parts, or 0.58% of the training data being defects.

One effective approach is to reduce the sample size of the majority class and increase the weight of the minority class. It can be achieved by either downsampling the majority class or oversampling the minority class. Downsampling may lead to a loss of information, while oversampling can result in overfitting. [17] elaborate on the sampling approaches, and they suggest that downsampling approaches give a better overall performance on all datasets. Thus, we tested the model with both techniques, and the best results were achieved by oversampling the minority class and downsampling the majority class.

IV. Experiments

In this section, we describe the datasets and the experimental results obtained in our study.

A. Data description

Bosch, a leading manufacturing company, has made a dataset available on Kaggle as part of a research project to assess the quality and safety of its manufacturing recipes. The dataset tracks parts as they move through the production lines, each with a unique identifier that serves as a row in the dataset. The dataset is a time series, with each row representing a specific section and the date attribute providing the time when each measurement was taken.

The dataset contains three types of characteristics: numeric, categorical, and date characteristics. Each feature is named according to a specific convention, including the line, station, and feature number. For example, the feature L1_S25_F2202 was measured at line 1, station 25, and has a sequence number of 2202. Therefore, this feature is measured in column L1_S25_D2203, since each feature Lx_Sy_Fn is calculated at time Lx_Sy_D(n+1).

By processing the column names, we concluded that there are four lines and 52 segment stations. Each line performs a specific production process, and one line can have multiple stations where different operations are performed, such as machining, turning, and welding. Line L0 has stations S0 to S23, line L1 has stations S24, S25, line L2 has stations S26, S27, S28, and line L3 has stations S29 to S51. Each workstation performs various tests and measurements on a given part, resulting in 4,264 features. We have 969 numeric features, 1,156 date features, and the rest are categorical.

The date features are normalized and given as takt time, which is a value for how long it takes for a process to fulfill the demand. To determine the period the data represent, we analyzed the unique values in the date features. There are 105,413 unique values, ranging from 0 to 1,718.48, with a rate of 0.01. Figure 2 shows a graph of the values and their frequencies, indicating space in the middle, meaning no measurements were made during that period.

We conducted an autocorrelation analysis using a lag function to understand the time dependence between the parts further. The results are presented in Figure 3, where the most significant values are at 1,675, with 7 local maxima corresponding to 7 days of the week. Therefore, 16.75 in the normalized data correspond to one week, and the data have a granularity of 6 minutes. It means that 0.01 corresponds to 6 minutes, and 1 corresponds to 600 minutes or 10 hours, indicating that the data correspond to two years.

For each part, we determined the maximum and minimum date, their difference, and the station with the maximum and minimum time. We also calculated the path size for each part by counting the number of non-zero values. Additionally, we found the week for the maximum and minimum dates by calculating a module of 16.75 on such values. Given the week, we labeled each day of the week, where 1 is Monday, 2 is Tuesday, 3 is Wednesday, 4 is Thursday, 5 is Friday, 6 is Saturday, and 7 is Sunday. Therefore, there are a total of 10 date features. In addition, as described previously, we added 9 features based on time dependence. Overall, in addition to focusing on the preprocessing of the data, we also analyzed the raw data to extract useful features for our model.

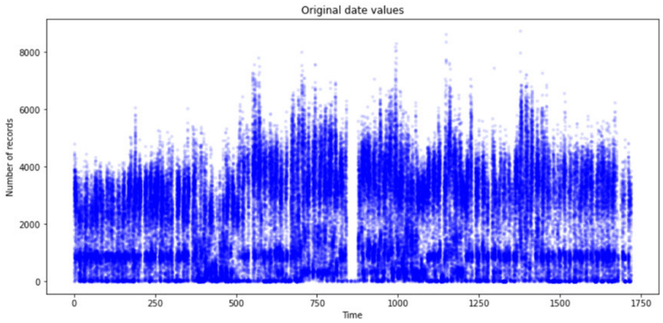

Fig. 2. Date values with frequencies

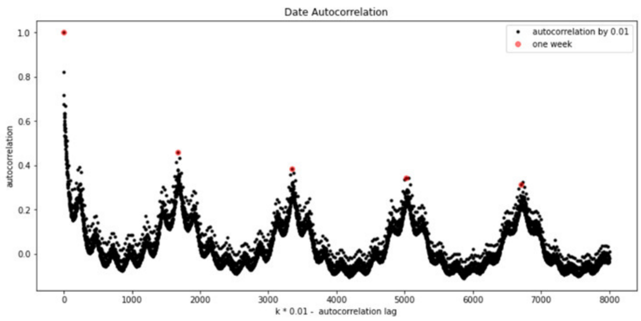

Fig. 3. Autocorrelation of date features. Autocorrelation for the number of observations recorded daily as a function of the time lag between them.

B. Experimental setup

After extracting features from the dataset, we train multiple classifiers on a balanced dataset. The classifiers include Random Forest, Decision Tree, Gradient Boosting, Ada Boosting, and XGBoost. Next, we will compare the accuracy results between 20% of testing data and the entire training dataset. Then, using the unlabeled dataset, we will predict the number of faults.

To obtain the best results, we have trained models using different combinations of features. We found that including all features (date, numerical, and categorical) resulted in the worst accuracy. Therefore, we removed the categorical features, leading to the models' highest accuracy. We also compared the results using standardized features and raw features. Standardized features yielded better results, and we used Z-Score Standardization. Z-Score Standardization is a widely used method to standardize data in machine learning. It transforms each value of a given feature in the dataset to a representative number of standard deviations away from that feature's mean. The resulting standardized value measures how far the raw value is from the mean in standard deviation units.

C. Accuracy results

Because we are dealing with an unbalanced dataset, for correct accuracy results, we have to consider true positives, true negatives, false positives, and false negatives. AUC-ROC (Area Under the Receiver Operating Characteristic Curve) plots the true positive rate (sensitivity) against the false positive rate (1-specificity) at various threshold settings and calculates the area under the curve. The AUC-ROC score ranges from 0 to 1, with higher values indicating better performance. MCC (Matthews Correlation Coefficient): MCC is a correlation coefficient used in binary classification that considers true positives, true negatives, false positives, and false negatives. MCC ranges from -1 to +1, with +1 indicating a perfect classification, 0 indicating a random classification, and -1 indicating a completely wrong classification. F1-Score (F-Measure): F1-score is the harmonic mean of precision and recall. It provides a single score that balances precision and recall and is often used to measure a model's performance in binary classification. F1-score ranges from 0 to 1, with higher values indicating better performance.

After several tests of the parameter values, the following obtained the best result using the XGBoost classifier: learning_rate=0.2, n_estimators=100, max_depth=16, min_child_weight=3, colsample_bytree=0.9, gamma=1, subsample=0.9, booster='gbtree', objective='binary:logistic'.

Table I reports results regarding AUC ROC, MCC, Precision, Recall, and F-Score on the training dataset. Table II provides accuracy scores on 20% testing data.

TABLE I
SUMMARY OF ACCURACY SCORES ON TRAINING DATASET

Model	**AUC ROC**	**MCC**	**Precision**	**Recall**	**F-Score**
Random Forest	0.705	0.148	0.53	0.71	0.51
Decision Tree	0.717	0.149	0.53	0.72	0.51
Gradient Boosting	0.691	0.22	0.56	0.69	0.59
Ada Boosting	0.629	0.187	0.57	0.63	0.59
XGBoost	**0.906**	**0.808**	**0.9**	**0.91**	**0.9**

TABLE II
ACCURACY SCORES ON 20% TEST DATA

Model	AUC ROC	MCC
Random Forest	0.801	0.613
Decision Tree	0.986	0.972
Gradient Boosting	0.917	0.837
AdaBoost	0.903	0.809
XGBoost	**0.997**	**0.994**

The evaluation metrics also provide additional insights into the performance of the models. For example, the XGBoost model achieved the highest accuracy of all the models, indicating that it correctly classified most test sets. In addition, the XGBoost model also has the highest F1 score, meaning a good balance between precision and recall. On the other hand, the other models, such as Gradieng and Ada Boosting, struggled to classify the fault class, reflected in their lower F1 scores. The MCC scores were also generally low for all models except XGBoost, indicating that the models had trouble with the imbalanced nature of the dataset. However, the AUC-ROC scores for Random Forest and Decision Tree were relatively high, suggesting they could distinguish between the positive and negative classes reasonably well. The results indicated that the XGBoost model is the most effective for this classification task.

Using the best accuracy model, XGBoost, to classify the additional unlabeled dataset, we identified 60,028 out of 1,183,748 samples as faulty. It means that the fault rate in the unlabeled dataset is approximately 5.07%. This information can help identify potential issues in the manufacturing process and improve the quality control procedures. However, the accuracy of the classification results on the unlabeled dataset may vary depending on the data's quality and representativeness and the model's performance on unseen data.

V. Conclusion

This paper presented a novel approach to predict malfunctions in manufacturing processes using the Bosch manufacturing dataset. The dataset is large and complex, containing many features with varying data types.

In this study, we addressed the challenge of handling a large number of features in the Bosch manufacturing dataset. The feature extraction process was a crucial step in the predictive modeling pipeline. We performed an iterative feature selection process to identify the most relevant features for predicting malfunctions in the manufacturing process. Additionally, time-dependent features were added to the dataset, improving the predictions' accuracy. The feature selection process was carried out carefully to ensure the selected features were relevant for the prediction task while avoiding overfitting the training data. The selected features were then used to train and evaluate various machine learning models. Handling the unbalanced dataset was another key factor in achieving high accuracy scores, and this was accomplished by performing both downsampling on the majority class and oversampling on the minority class. The study results showed that XGBoost outperformed the other models in terms of accuracy scores, including AUC ROC, MCC, and F1-score.

The proposed approach of supervised malfunction prediction using machine learning models and feature engineering can have significant implications in the manufacturing industry. Manufacturers can proactively prevent downtime, optimize maintenance schedules, and minimize production losses by accurately predicting malfunctions. As a result, it can improve manufacturing operations' efficiency and productivity, leading to cost savings and increased profitability. Moreover, the approach can also help identify patterns and insights in the data that can be used for process optimization and improvement.

In future work, we aim to extend our study to include categorical features. While we made significant progress in feature extraction and handling unbalanced data, the categorical features remain an important part of the dataset that needs further investigation. Therefore, we plan to explore various techniques for feature extraction on categorical data and evaluate their impact on the model's overall accuracy.

Acknowledgment

This work was partially financed by the Faculty of Computer Science and Engineering at the Ss. Cyril and Methodius University, Skopje, Macedonia. This work is also partially funded by FCT/MEC through national funds and, when applicable, co-funded by the FEDER-PT2020 partnership agreement under the project **UIDB/50008/2020**.

References

[1] M. Ghobakhloo, "Industry 4.0, digitization, and opportunities for sustainability," *Journal of cleaner production*, vol. 252, p. 119869, 2020.

[2] B. Natesha and R. M. R. Guddeti, "Fog-based intelligent machine malfunction monitoring system for industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 7923–7932, 2021.

[3] K. Wang and Y. Wang, "How ai affects the future predictive maintenance: a primer of deep learning," in *Advanced Manufacturing and Automation VII 7*. Springer, 2018, pp. 1–9.

[4] P. Poór, J. Basl, and D. Zenisek, "Predictive maintenance 4.0 as next evolution step in industrial maintenance development," in *2019 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, 2019, pp. 245–253.

[5] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei, "Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0," *Sustainability*, vol. 12, no. 19, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/19/8211

[6] Y. Ren, "Optimizing Predictive Maintenance With Machine Learning for Reliability Improvement," *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, vol. 7, no. 3, 05 2021, 030801. [Online]. Available: https://doi.org/10.1115/1.4049525

[7] E. Zdravevski, P. Lameski, A. Kulakov, S. Filiposka, D. Trajanov, and B. Jakimovski, "Parallel computation of information gain using hadoop and mapreduce," in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2015, pp. 181–192.

[8] E. Zdravevski, P. Lameski, A. Kulakov, B. Jakimovski, S. Filiposka, and D. Trajanov, "Feature ranking based on information gain for large classification problems with mapreduce," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 2. IEEE, 2015, pp. 186–191.

[9] E. Zdravevski, P. Lameski, C. Apanowicz, and D. Slezak, "From big data to business analytics: The case study of churn prediction," *Applied Soft Computing*, vol. 90, p. 106164, 2020.

[10] M. Grzegorowski, E. Zdravevski, A. Janusz, P. Lameski, C. Apanowicz, and D. Slezak, "Cost optimization for big data workloads based on dynamic scheduling and cluster-size tuning," *Big Data Research*, vol. 25, p. 100203, 2021.

[11] P. F. Orrù, A. Zoccheddu, L. Sassu, C. Mattia, R. Cozza, and S. Arena, "Machine learning approach using mlp and svm algorithms for the fault prediction of a centrifugal pump in the oil and gas industry," *Sustainability*, vol. 12, no. 11, p. 4776, 2020.

[12] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Machine learning-based approach for hardware faults prediction," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 11, pp. 3880–3892, 2020.

[13] W. Rhmann, B. Pandey, G. Ansari, and D. K. Pandey, "Software fault prediction based on change metrics using hybrid algorithms: An empirical study," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 4, pp. 419–424, 2020.

[14] M. H. Chegeni, M. K. Sharbatdar, R. Mahjoub, and M. Raftari, "New supervised learning classifiers for structural damage diagnosis using time series features from a new feature extraction technique," *Earthquake Engineering and Engineering Vibration*, vol. 21, no. 1, pp. 169–191, 2022.

[15] A. Mangal and N. Kumar, "Using big data to enhance the bosch production line performance: A kaggle challenge," in *2016 IEEE international conference on big data (big data)*. IEEE, 2016, pp. 2029–2035.

[16] E. Zdravevski, P. Lameski, V. Trajkovik, A. Kulakov, I. Chorbev, R. Goleva, N. Pombo, and N. Garcia, "Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering," *IEEE Access*, vol. 5, pp. 5262–5280, 2017.

[17] S. Tyagi and S. Mittal, "Sampling approaches for imbalanced data classification problem in machine learning," in *Proceedings of ICRIC 2019: Recent Innovations in Computing*. Springer, 2020, pp. 209–221.