Fault Detection and Classification for Robotic Test-bench: A Data Challenge

Kürşat İnce[1,4], Uğur Ceylan[2,4], Nazife Nur Erdoğmuş[1], Engin Sirkeci[3], and Yakup Genc[4]

[1] *Naval Combat Management Technologies Center, HAVELSAN Inc., Pendik/İstanbul, Türkiye*
{*kince,nerdogmus*}*@havelsan.com.tr*

[2] *Doruk Automation and Software Inc., Pendik/İstanbul, Türkiye*
ugrceyln@gmail.com

[3] *Defense Systems Technologies Vice Presidency, ASELSAN Inc., Yenimahalle/Ankara, Türkiye*
esirkeci@aselsan.com.tr

[4] *Computer Engineering Department, Gebze Technical University, Gebze/Kocaeli, Türkiye*
{*kince,uceylan,yakup.genc*}*@gtu.edu.tr*

ABSTRACT

Maintenance of industrial systems often cost as much as their initial investment. Implementing predictive maintenance via system health analysis is one of the strategies to reduce maintenance costs. Health status and life estimation of the machinery are the most researched topics in this context. In this paper, we present our analysis for Sixth European Conference of the Prognostics and Health Management Society 2021 Data Challenge, which introduces a fuse test bench for quality-control system, and asks fault detection and classification for the test bench. We proposed classification workflows, which deploy gradient boosting, linear discriminant analysis, and Gaussian process classifiers, and report their performance for different window sizes. Our gradient boosting based solution has been ranked 4th in the data challenge.

1. INTRODUCTION

Fault detection, diagnosis, and prognostics has been active area of research for the last few decades, which is an essential part of modern industries to ensure safety and product quality (Heo & Lee, 2018). By enabling widespread integration of diagnostics and prognostics into modern production systems, uncertainties associated with life cycle of system have reduced. Prognostics and health management (PHM) is performed with varying degrees of success for a number of different reasons. There are currently no standards to demonstrate best practices comparatively because each problem can be solved in a variety of ways. The PHM Data Challenge, an open data competition specialized in PHM, is an opportunity to competitively determine leading solutions for industrial problems. The PHM Data Challenge pioneers the development of PHM issues by presenting a wide spectrum of real-world industrial problems with abundant resources. It serves as a library of various case studies from which we can learn about the industrial problems proposed each year and the current challenges in practice, the flow of thought for addressing these challenges, and the advantages and disadvantages of different methods (Huang, Di, Jin, & Lee, 2017).

In this paper, we present our analysis for Sixth European Conference of the Prognostics and Health Management Society 2021 Data Challenge (PHME21 Challenge) (Giordano & Gagar, 2021). The challenge asks participants to demonstrate application of state-of-the-art algorithms and models to perform fault detection, classification and root cause identification for a fuse-test bench. Through a data pipeline, we compared gradient boosting (GB), Linear Discriminant Analysis (LDA), and Gaussian process (GP) based models to solve challenge's tasks.

This analysis paper is organized as follows: Section 2 describes fault detection, classification and root cause identification of a system, and gives information about the challenge

dataset. Section 3 describes how we attacked the Data Challenge, and the workflow we used for analysis. Section 4 describes our implementation and results that we had for the Data Challenge.

2. Fault Detection and Classification

Fault diagnosis of machine tools which is provided by real time condition monitoring of sensors and abnormal pattern recognition are crucial in root cause identification. Before proceeding to the stage where the root cause of the failure is identified, the data must be preprocessed, classified and then failure diagnosed. The main roles of fault detection and diagnosis are to make an effective indicator which can identify faulty status of a process and then to take a proper action against a future failure. This indicator provide prediction the correct fault which is to forecast the time left before the system losses its operation ability, based on the condition monitoring information.

PHME21 data challenge competitors are challenged to showcase their abilities on a manufacturing production line setup's rich dataset generated from a real-world industrial testbed is provided by Swiss Centre for Electronics and Microtechnology (CSEM). Sub-systems such as conveyor belt motors, infrared camera and robotic arms used in the continuous testing process of electronic components constitute the fuse test bench. The aim of the fuse test bench is to test electrical fuses on the large-scale quality-control pipeline. The fuse test bench consists of a 4-axis SCARA-robot picking up electrical fuses with a vacuum gripper, from a feeder to a fuse-test-bench. On this fuse test bench, if the electrical conductivity of the fuse is established, the fuse is heated by applying 200mA current in 1.5s time interval. Heating is measured with a thermal camera. After the testing is completed, the fuse is moved back into the feeder with two conveyor belts.

The fuse test bench, which is used in quality-control process, is shown in Figure 1. In the process, first the fuses are first picked up by the robotic arm (1). Fuses are carried to the visual field of a thermal-camera (2) responsible for finding signs of overheating or degradation. Once the analysis is terminated, fuses are placed on a conveyor belt (3) and sorted by a robotic bar (4). Fuses are moved to small conveyor belt (5) that transports them back to the feeder (6) where fuses are stored before restarting the cycle.

The experimental dataset contains 50 sensors readings which recording the evolution of a number of quantities of interest to establish the health state of the machine in real-time. It is desired to reduce the data size by calculating one statistical data point for each window and sensor. In addition, each signal is associated with a specific set of fields specific to that signal, identifying different signals' features extracted from that signal by automated data acquisition process. For each 10 seconds of time window, the signals are described by the statistical features given in Table 1.

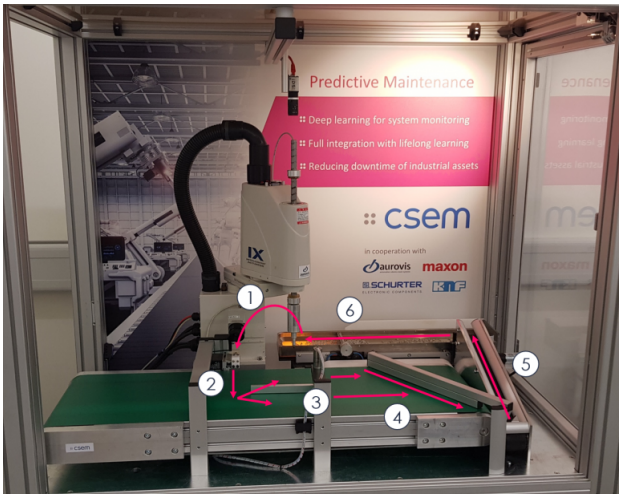


Figure 1. The Fuse Test Bench for The PHME21 Data Challenge

Table 1. Statistical features derived from sensors

Name	Description
vCnt	Number of samples recorded in the time window
vFreq	Sampling frequency
vMax	Maximum value of the samples
vMin	Minimum value of the samples
vStd	Standard deviation of the samples
vTrend	Derivative-based trend of the samples
value	Mean value of the samples

An experiment may run 1 hour to 3 hours. Experiments have been generated with a variety of seeded faults under controlled conditions. At the same time, some experimental data have been acquired under fault-free operating conditions. Data Challenge's public dataset contains experiments divided into training and validation and model refinement subsets. In the evaluations of the challenge submissions, a private test subset is allocated. Training and validation subsets contain 70 experiments describing fault-free experiments (Class 0) and 5 faulty classes (Classes 2, 3, 5, 7, and 9) in total. Model refinement subsets contain 29 experiments describing fault-free experiments (Class 0) and 3 faulty classes (Classes 4, 11, and 12).

Fault-free experiments, which have Class 0, represent the behaviour of the machine during its normal operating regime. In normal operating regime, the machine runs smoothly and does not present any problem. Fault-free experiments' system parameter configurations lead to a nominal system behaviour. Unhealthy experiments which are characterized by anomalous behaviour, have been labeled with 8 different la-

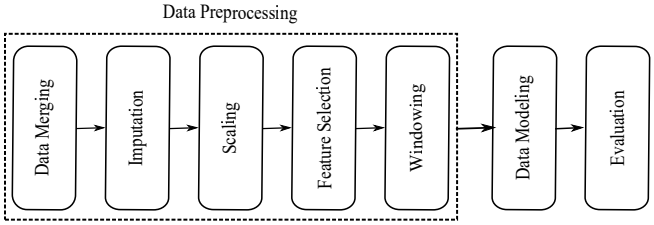


Figure 2. Data processing workflow for building classification models

bels. Each fault is characterized by an anomalous behaviour of one or more signals.

The main objective of the PHME21 Data Challenge is identification and classification of the faults in unlabeled data (*Task #1*). Other objectives include identification and ranking of the features that help us to classify faults correctly (*Task #2*), predicting the correct fault in the earliest time stamp (*Task #3*), and developing unsupervised solutions that identify the experiments' system parameter configurations (*Bonus Task*). The methods that handle challenge tasks are given in the next section.

3. Methods and Techniques

Our main motivation for the challenge was to build a fault classifier, which would be used to solve Task #1 of the challenge. We have built a data processing workflow (Figure 2), which contains data preprocessing, model training, and evaluation phases, to satisfy classification objective.

3.1. Data Preprocessing

The dataset is preprocessed and made ready for training and evaluation using the following steps.

Data Merging: The dataset contains individual data files for each experiment. We merged individual data files to have three separate datasets for training and validation (70 experiments) and model refinement (29 experiments). We have added experiment setups for each sample of reading for further processing. The merged training and validation subsets and model refinement dataset have readings from sensors. The specific set of fields of each sensors were considered as features.

Missing Value Imputation: The dataset contains missing data for each experiment. Because missing data can create problems for analyzing data, we have used interpolation to fill in missing data and avoid pitfalls involved cases that have missing values.

Scaling: Data normalization is performed before data modelling using RobustScaler from Scikit-Learn Library (Pedregosa et al., 2011). While in operation, RobustScale removes the median and scales the data according to the given quantile range, which defaults to the range between the 1st quartile and the 3rd. Thus, the nominal data transforms gracefully while the outliers are respected. Initially training data is scaled, and then the scaling parameters are applied to validation data.

Feature Selection: We used Leave One Feature Out Importance (LOFO-Importance) package (Erdem & Collot, n.d.) to select which features to use during model training. By leaving out one feature at a time during iterations, LOFO-Importance calculates each features contribution to the learning task. We used balanced accuracy metric which handled classification tasks on imbalance dataset.

Windowing: We have used window sizes of multiples of 5 up to 50 to create a context for the time series data. Windowed dataset have passed to model training phase.

3.2. Data Modeling and Optimization

We have used tree based ensemble learning algorithms, namely gradient boosting for modeling stage. We also deployed linear discriminant analysis both as dimension reduction technique and as a classifier. Aside from single classification model, we have also deployed Gaussian process in two phase classification processes. For gradient boosting methods, we performed hyper-parameter optimization using genetic algorithms.

Gradient Boosting (GB): GB, one of the most powerful techniques for performing classification and regression tasks, builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function (Friedman, 2001). GB is an ensemble learner: a final model based on a collection of individual models. These individual models have poor predictive power and are prone to overfitting, but combining many such weak models in an ensemble will lead to a much better outcome overall. We have used XGBoost (XGB) (Chen & Guestrin, 2016) and LightGBM (LGBM) (Ke et al., 2017) implementations in our analysis.

Linear Discriminant Analysis (LDA): LDA, or discriminant function analysis is a generalization of Fisher's linear discriminant, a method used in statistics and other fields, to find an accurate representation of two or more object objects denoting their class or distinguishing features. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification. Three steps needed be performed to achive the LDA goal. The first step is to calculate the separability between different classes. The second step is to calculate the in-class variance (the distance between the mean and the samples of each class). The last step is to construct the lower-dimensional space that maximizes inter-class variance and minimizes intra-class variance (Tharwat, Gaber, Ibrahim, & Hassanien, 2017). We have used LDA implementation in

Scikit-Learn Library (Pedregosa et al., 2011).

Gaussian Process (GP): A GP is a probability distribution over possible functions (Rasmussen & Williams, 2005). GPs are a generic supervised learning method designed to solve regression and probabilistic classification problems. Their greatest practical advantage is that they can give a reliable estimate of their own uncertainty. GP extend multivariate Gaussian distribution to infinite dimensionality. The key idea of GP is to model the underlying distribution training data as a multivariate normal distribution. Learning a distribution enables the model to output a prediction and an uncertainty associated with the prediction. We have used GP implementation in Scikit-Learn library (Pedregosa et al., 2011).

Genetic Algorithms (GA): GA is a evolution based metaheuristic search algorithm, which is inspired by natural selection process (Back, Fogel, & Michalewicz, 2000). In machine learning, GA is commonly used for optimization of the hyperparameters of the classification (or regression) models. GA requires the genetic representation of the hyper-parameters, called individuals, as in gene sequences, and a fitness function to evaluate the gene's adaptation to the "environment." The GA process starts with a randomly generated initial population of individuals. For each iteration, also called a generation, fitness of the individuals are evaluated. The genes of the most fit individuals are selected to form the next generation's individuals. Individuals of the new generations are created randomly via mating (crossover of the gene sequences), or via mutation (random changes on the genes). The iterations continue until enough number of generations are produced, or until satisfactory fitness level is reached. We have used DEAP (Fortin, De Rainville, Gardner, Parizeau, & Gagné, 2012) package for GA optimizations.

Our GB and LDA models are built as a pipeline, which contains a scaler, a dimension reducer, and a classifier. Our two phase classification model deploys two pipeline for each phase. The first phase uses 8 one-vs-one classifiers, which are trained for Class 0 vs Class N (N is for each of the faulty classes). Modeled with GP, this phase is used to predict the probability distribution each possible classifier. The second phase uses LDA on the concatenation of these probability distributions to predict the true class labels.

3.3. Evaluation Metrics

Since the dataset is imbalanced, evaluation of the model is performed using F1 score (F1), and Matthews correlation coefficient (MCC) (Matthews, 1975) metrics in a 3-fold cross validation setup. MCC is more informative than other metrics, because it takes all the balance ratios of the all the measures, namely true positives (tp), true negatives (tn), false positives (fp), false negatives (fn), into account. Thus, MCC is our chosen metric for imbalance datasets. Calculations of F1 and MCC metrics is are given in Equation 1 and Equation 2, respectively.

$$F1 = 2 \times \frac{tp}{tp + \frac{1}{2} \times (fp + fn)} \tag{1}$$

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \tag{2}$$

We used cross validation scores to compare model performances. For the final predictions during the test experiments, we use most occurring fault label as the final classification label of the experiment.

3.4. Other Challenge Tasks

Classification of the fault is the main driver for the challenge. When we have good enough classifier, we can solve Task #2 and #3 of the challenge.

For solving Task #2, i.e. identification and ranking of the sensors per faulty class, we used LOFO based approach on sensors. In this approach, first we use all sensor features to calculate a base score for the binary classification model. Then we iterate over the features. By removing one feature at each iteration, and calculating a new score from the model using the remaining features, we calculate each feature's contribution to the base score. Sorted list of score contributions gives us the feature rank for the fault.

The final classification label for an experiment is assigned as the most occurring prediction label of the experiment's sensor data. For solving Task #3, i.e. the shortest time prediction, we search for a cut time that satisfies the above requirements, namely the two most occurring labels for the full experiment data are also the two most occurring labels for the trimmed experiment data.

Feature selection step in data preprocessing stage reports humidify and temperature as the most important features for the fault-free experiments (Class 0). For the Bonus Task, i.e. identifying system configurations, these features are adopted to build a clustering model using K-Means algorithm with k=2.

4. Experiments and Results

We used Scikit-Learn and other python libraries to implement the classification workflow. Our analysis with the training data showed that some experiments of the Class 0, namely 6, 23, 35, 49, 54, 56, 74, and 83, decreased the classification performance. Since Class 0 had $\times 16$ more experiments than other classes, we simply removed these runs from the dataset.

Initially, we used the classification methods with their default settings. We adopted 3-fold cross validation for different window sizes on the dataset. During the evaluation, we

watched the model performance through evaluation metrics F1 and MCC, and through confusion matrix (CM) for individual folds. While F1 and MCC metrics showed models' overall performance numerically, CMs showed the models performance on class separation. CMs of a sample run for XGB model is given in Figure 3.

The final 3-fold cross validation results are given in Table 2. Two-phase classification (i.e. GP+LDA) gave better results when the window size is small (i.e. 5 or 10). This is because of the classification power of GP to deal with uncertainty and lesser data. When the window size increases other methods perform better. Although GB implementations gave similar results for different window sizes, XGB's MCC scores were 0.01 to 0.02 points better than LGBM. LDA score are between LGBM and XGB scores.

Using DEAP Library (Fortin et al., 2012), we also performed Genetic Algorithms (GA) based optimization on the hyper-parameters of LGBM and XGB to increase model performances. Hyper-parameters for GA evolutions are given in Table 3. GA optimization parameters are selected manually after a few experimental runs. The hyper-parameters for the best model is constructed from these optimizations. The hyper-parameters and evaluation results are given in Table 4. Although GA optimization added 0.01 point to our previous results, we preferred to submit XGB model for window size 40.

The notebooks and other material we used throughout the challenge is available at https://github.com/zakkum42/phme21-public.

5. Conclusion

We presented our analysis for PHME21 Data Challenge, which asks for fault detection and classification of a fuse test bench for quality-control system. We built a data pipeline, and used gradient boosting (LGBM, and XGB), linear discriminant analysis (LDA) and Gaussian process (GP) classification algorithms. Two-phase Gaussian process classifier predicted better than other algorithms for smaller window sizes. Performance of LGBM, XGB, and LDA classifications were better with the increased window sizes. We also performed hyper-parameter optimization using genetic algorithms, which added 0.01 points to out previous results. Our XGB based solution has been ranked 4th in the data challenge.

References

Back, T., Fogel, D. B., & Michalewicz, Z. (2000). *Evolutionary computation 1: Basic algorithms and operators* (1st ed.). IOP Publishing Ltd.

Chen, T., & Guestrin, C. (2016). Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*. doi: https://dx.doi.org/10.1145/2939672.2939785

Erdem, A., & Collot, S. (n.d.). *Lofo (leave one feature out) importance.* (https://github.com/aerdem4/lofo-importance)

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012, jul). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*, 2171–2175.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232.

Giordano, D., & Gagar, D. (2021). *Sixth european conference of the prognostics and health management society 2021 data challenge.* (https://phm-europe.org/)

Heo, S., & Lee, J. H. (2018). Fault detection and classification using artificial neural networks. *IFAC-PapersOnLine*, *51*(18), 470-475. (10th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2018) doi: https://doi.org/10.1016/j.ifacol.2018.09.380

Huang, B., Di, Y., Jin, C., & Lee, J. (2017, 05). Review of data-driven prognostics and health management techniques: Lessons learned from phm data challenge competitions..

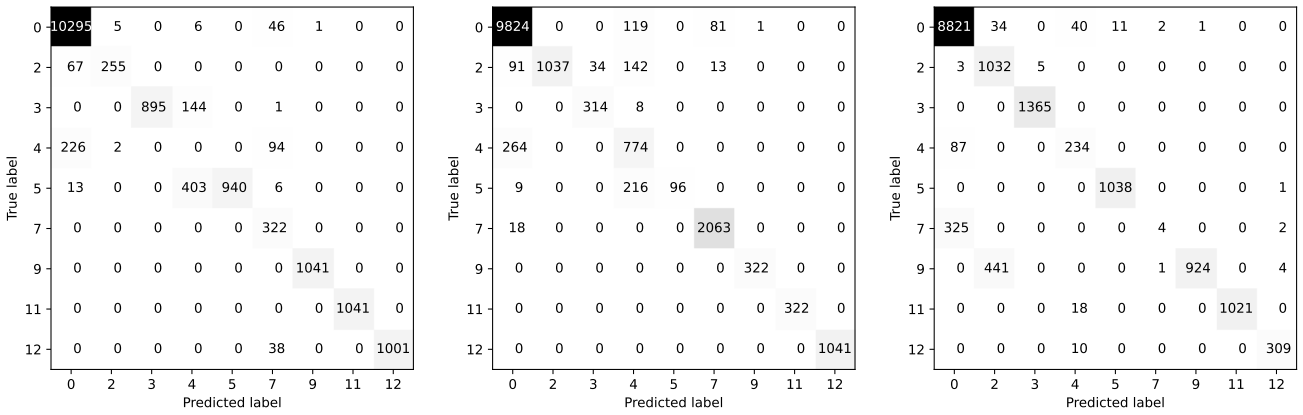
Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon et al. (Eds.), *Proceedings of the 31st international conference on neural information processing systems* (Vol. 30, p. 3149–3157). Curran Associates Inc.

Matthews, B. (1975, oct). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, *405*(2), 442–451. doi: https://doi.org/10.1016/0005-2795(75)90109-9

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian processes for machine learning (adaptive computation and machine learning)*. The MIT Press.

Tharwat, A., Gaber, T., Ibrahim, A., & Hassanien, A. E. (2017, 05). Linear discriminant analysis: A detailed tutorial. *Ai Communications*, *30*, 169-190,. doi: https://doi.org/10.3233/AIC-170729



(a) Fold 0 (b) Fold 1 (c) Fold 2

Figure 3. The figure displays sample confusion matrices for different folds during training of an XGB model. The CMs suggest that the model fails to separate Class 4, 5 and 7, and that these classes are confused with each other and with Class 0.

Table 2. Results of Classification Models for Different Window sizes

Classifiers	ws=5		ws=10		ws=15		ws=20		ws=30		ws=40		ws=50	
	F1	MCC	F1	MCC	F1	MCC	F1	MCC	F1	MCC	F1	MCC	F1	MCC
LGBM	0.8	0.67	0.87	0.79	0.9	0.84	0.9	0.86	0.9	0.86	0.91	0.87	0.9	0.85
XGB	0.8	0.69	0.89	0.82	**0.9**	**0.86**	**0.91**	**0.87**	**0.91**	**0.88**	**0.91**	**0.89**	**0.92**	**0.88**
LDA	0.83	0.7	0.88	0.81	0.9	0.83	0.9	0.83	0.9	0.84	0.92	0.86	0.91	0.85
GP+LDA	**0.89**	**0.82**	**0.9**	**0.84**	0.87	0.78	0.87	0.77	0.49	0.43	0.53	0.43	0.54	0.41

Table 3. Genetic Algorithms Search Parameters for Model Hyper-parameter Optimization

GA Parameter	Value
Initial population size	30
Number of generations	4
Population size per generation	10
Mate probability	0.5
Mutation probability	0.5
Number of selected individuals per generation	3

BIOGRAPHIES

Kürşat İnce received his BSc and MSc degrees from Bilkent University Computer Engineering Department in 1996 and in 1999, respectively. As a software developer, he joined HAVELSAN Inc. in 1996. Currently, he is employed as R&D coordinator in HAVELSAN's Istanbul R&D Center. Mr. İnce started PhD in Gebze Technical University Computer Engineering Department in 2015. Still working on his doctoral thesis, his research interests include machine learning, especially applications of deep learning methods in Industry 4.0. He is also one of the coordinators in Data Istanbul Meetup Group, a community for democratizing machine learning, since 2016.

Uğur Ceylan received BSc degree from Marmara University, Computer Engineering Department, İstanbul, Turkey in 2017. He joined Doruk Automation and Software Inc. as a AI-researcher and he is currently employed as a data scientist at the same firm. On the other hand he is currently doing a master's degree within Department of Computer Engineering, Gebze Technical University, Kocaeli, Turkey. His research interests include artificial intelligence, machine learning, deep learning, reinforcement learning and computer vision.

Nazife Nur Erdoğmuş received her BSc degree in Industry Engineering from the University of Necmettin Erbakan, Turkey, in 2020. She did also double major in same university with Computer Engineering. Currently, she is employed as software engineer in HAVELSAN Inc. Her research interests include optimization and data science.

Table 4. Hyper-parameters and Results with GA optimizations

Classifier	**Parameters**	**F1**	**MCC**
LGBM	ws=49 learning rate=0.01 boosting type='gbdt' no of leaves=10 max depth=-1 no of estimators=300 min split gain=0 subsample=0.5 subsamplefreq=0 colsample=0.8 objective='multiclass'	0.91	0.88
XGB	ws=20 learning rate=0.001 booster='dart' max depth=40 no of estimators=750 min child weight=0.5 gamma=0 max leaves=10 subsample=0.8 colsample=0.8 objective='multi:softmax'	0.93	0.90

Engin Sirkeci received the bachelor's degree and Master's degree within the Department of Industrial Engineering from Gazi University, Ankara, Turkey in 2009 and in 2015, respectively. He worked toward the Ph.D. degree within the Department of Computer Engineering, Gebze Technical University, Kocaeli, Turkey from 2018 to 2020. He is currently a Senior Specialist Engineer with System Engineering Directory at the Defense System Technologies Vice Presidency, ASELSAN Inc., Ankara, Turkey. His research interests include industrial AI, industrial big data, reliability and maintenance modeling, and prognostic and health management.

Yakup Genc received his PhD in Computer Science from the University of Illinois at Urbana-Champaign. Right after graduation, Dr. Genc joined Siemens Corporate Research (SCR) in September 1999. As research scientist, project manager, program manager and group manager, he developed technology and research strategy in the areas of computer vision, augmented reality and machine learning. His tenure at SCR produced numerous publications and patents. Since September 2012, as a member of the faculty of the Computer Engineering department at the Gebze Technical University, he continuous to conduct research in fields of computer vision, augmented reality, autonomous vehicles, machine learning and deep learning while maintaining close ties with the industry for practical applications of his research.