Explainable Machinery Faults Prediction Using Ensemble Tree Classifiers: Bagging or Boosting

Somayeh Bakhtiari Ramezani[1], Amin Amirlatifi[2], Thomas Kirby[1], Maria Seale[3], Shahram Rahimi[1]

[1] *Computer Science and Engineering Department Engineering, Mississippi State University, Starkville, MS, 39759, US*

sb3182@msstate.edu

tmk169@msstate.edu

rahimi@cse.msstate.edu

[2] *Swalm School of Chemical Engineering, Mississippi State University, Starkville, MS, 39759, US*

amin@che.msstate.edu

[3] *U.S. Army Engineer Research and Development Center (ERDC), US*

maria.a.seale@erdc.dren.mil

Abstract

Proactive maintenance aims to accurately classify temporal trends as early as possible, detect faulty states, and pinpoint the root cause of faults. Neither late nor early maintenance is desirable, as each will incur additional operating costs. While various data-driven techniques have been used to identify faults, many fail to perform when faced with missing values at run time or lack explainability. The present work introduces a framework to identify and classify anomalous signals in spite of missing values and partially labeled data. This framework offers explainability by identifying key performance indicators for each fault family using SHapley Additive exPlanations (SHAP). A comprehensive study was performed on existing algorithms to determine the best fault classifier, and candidates with accuracy greater than 80% were selected. This paper introduces a new missing value imputation technique based on Partial Least Squares (PLS-MV). It also uses fuzzy C-means (FCM) to detect different healthy unlabeled operations in the PHME21 dataset. Our results show that boosting algorithm is best suited for creating a generalized model that is capable of classifying faulty patterns in multi-label datasets which may include missing values.

1. Introduction

The ultimate goal of Predictive Maintenance (PdM) is scheduling right-on-time maintenance (Lee & Scott, 2006). As such, early detection of faulty patterns and timely scheduling of maintenance events can minimize risk to the underlying process. Condition Based Maintenance (CBM), which aims to maximize the useful life of a system by targeting maintenance for the time it is needed, can reduce the costs associated with the system downtime and subsequent damage to other subsystems that need to endure working in a defective state. While scheduling early maintenance would reduce the production efficiency by increasing the machine's downtime and adding unnecessary costs, late maintenance would damage the operation quality (Geramifard, 2013) by cascading damage to other healthy subsystems or slowing down the whole system. Thus, the critical component in *Proactive Maintenance* (PM) is a diagnosis of the root cause behind the various types of faulty states in a complex system that can determine what subsystem is the cause of the failure (Salfner & Malek, 2007; Lee & Scott, 2006).

Several attempts have been made to achieve eXplainable Artificial Intelligence (XAI) and apply them to different fields. However, when it comes to PdM, there are few XAI applications for anomaly detection and diagnostics. Steenwinckel et al. (2021) incorporated rules and semantic knowledge into ML and created a hybrid system that performed well in anomaly detection while maintaining explainability. Alfeo et al. (2020) used an autoencoder architecture for anomaly detection with results comparable to previously published works, but unlike the others, their work benefited from XAI features. Rajendran et al. (2019) also used an adversarial autoencoder architecture to detect anomalous behavior in wireless spectrum with an average accuracy of 80% and the ability to explain the features. Langone et al. (2020) used Logistic

Regression (LR) to detect and explain the anomalous behaviors observed in high-pressure plunger pumps.

Many classification techniques are used for PdM, examples of which include Decision Trees (DT) (Gupta, Uttarakhand, & Rawat, 2017), Random Forest (RF) (Breiman, 2001), k-Nearest Neighbors (k-NN) (Wu et al., 2008), Support Vector Machines (SVM) (Wu et al., 2008), and LR (Peng, Lee, & Ingersoll, 2002); however, it should be noted that not all of the techniques mentioned above are explainable. Kamal et al. (2021) provides a comprehensive look at the XAI techniques used in PdM. The precise and explainable nature of the tree-based algorithms has made them the most commonly used non-linear technique in PdM (Lundberg et al., 2019).

While theoretical models and idealized conditions can be used to develop PdM workflows, these solutions do not always translate to real-world examples. To that end, PdM under industrial settings faces challenges that can be classified into three main categories: overcoming missing values, identifying the subset of sensors that play a critical role in pinpointing the faulty patterns, and handling unlabeled data.

This study introduces a new framework to identify anomalous signals, classify their faulty states, and determine the most important attributes that define each faulty class. This framework consists of components that can handle the missing values and partially labeled data and identify the important features that play critical roles for each class type. We evaluate the performance of various classifier algorithms on the 2021 European Conference of the Prognostic Health Management Society (PHME) data challenge data set (PHME21). With the goal of minimizing prediction errors caused by missing values, while maximizing the accuracy of class type identification, we introduce an ensemble method called Partial Least Squares-Missing Value (PLS-MV) for missing value imputation. This framework also offers explainability by identifying key performance indicators for each fault family using SHapley Additive exPlanations (SHAP).

The rest of the article is organized as follows: in section 2 we present a background on Bagging and Boosting algorithms, along with an overview of the dataset used in the present study. Section 3 discusses our methodology, followed by a discussion of the results in section 4, and conclusions.

2. BACKGROUND

When it comes to working with supervised data, the DTs are among the most widely used techniques. Their ability to be visualized, plus their simplicity and applicability to both numerical and categorical data, has made them a prime candidate for inductive inference (Gupta et al., 2017). Their downside, however, is their susceptibility to over-fitting. Several techniques, such as bootstrap aggregating (Bagging) and boosting, have been developed to overcome this shortcoming of DTs and increase their performance.

Figure 1 shows an overall comparison between a single estimator (such as DT) versus multi-estimator algorithms that combine outputs of individual estimators (Aporras, 2016).

2.1. Bagging methods

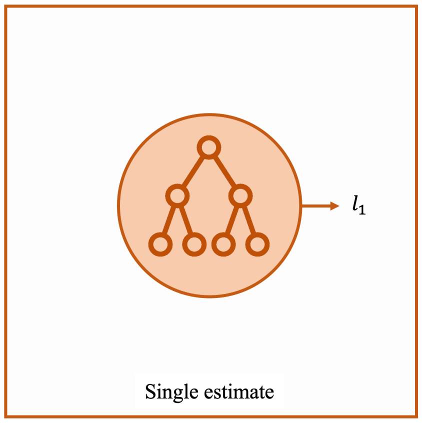
Bagging is an ensemble method introduced in (Breiman, 1996). An ensemble method incorporates many models or classifiers to create a final classification that outperforms a single classifier by balancing the differences between models and the accuracy of each individual model (Kotsiantis, 2014). To create a diversity of models and outperform the biases of a single classifier, bagging starts by randomly sampling a data set to create a new training set to be used by a unique classifier. The sampling is done with replacement, meaning that training instances are not exclusive to a single model, and that some training examples may be excluded from all models. Each model is independent of all other models in the ensemble, meaning that the classification can be done in parallel. The final classification is done by averaging all of the classifications of the models equally, as shown in Figure 1-b.

Random Forest (RF) is a specific variant of bagging that was presented in (Breiman, 2001), extending the method to decision trees. Like in bagging, randomly sampling a data set with replacement serves as input for creating individual decision trees that consist of branching *if statements* that classify the data. The changes in the randomly selected data result in a diverse set of decision trees that form a random forest. Again following the main ideas of bagging, when the random forest makes a prediction, a majority vote from the decision trees serves as the final prediction.

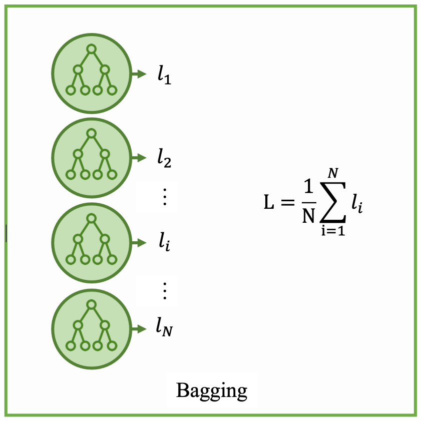
2.2. Boosting methods

Like the bagging method, the boosting method starts with classifier training, randomly selecting data with replacement. After the first classifier is trained, a new classifier is created using updated weights based on the accuracy of the previous model. Because the selected data for the next classifier depends on the results of the previous model, boosting has to be done sequentially, unlike bagging. A weighted average of all of the models' classifications is performed to get a final classification output. How well a single model in the boosting method performed determines the weight of that model in the overall classification (Freund & Schapire, 1999). This means that if a model is generating results that are similar to the expected output, it gets a higher weight in the overall decisions, thus influencing subsequent decisions.

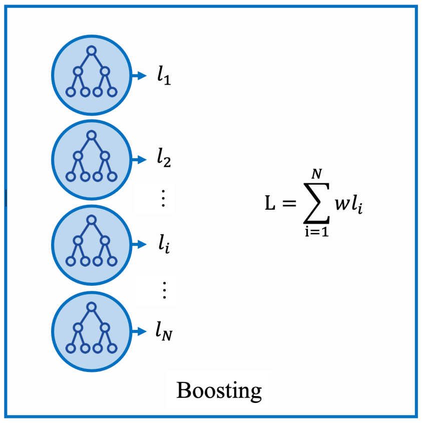
One of the earliest and most popular implementations of boosting is Adaptive Boosting (AdaB) (Freund & Schapire, 1997), which can work on any classifying algorithm, improving upon incorrectly classified instances of previous models. Using a gradient of a loss function to tell how a previous model has performed, this algorithm sequentially produces



(a) Single Learner



(b) Bagging



(c) Boosting

Figure 1. Bagging and Boosting both require a base learner. In case of a tree based classifier, the base classifier can only make a single estimate (a); Bagging (b) with the tree based classifier will average the estimations of each classifier. Boosting (c) will apply weighted averaging over the results of its classifiers, boosting the contribution of the most successful ones.

a new model that gradually becomes more and more optimal in branching the trees.

As a more efficient implementation of gradient boosting decision trees, Light Gradient Boosting (LightGB) (Ke et al., 2017a) only uses data with large gradients, resulting in a reasonable estimation of the information gain while using much less data. Histogram-based Gradient Boosting classification tree (HistGB) is another popular boosting estimator that tolerates datasets with missing values, and is inspired by the LightGB algorithm (Ke et al., 2017a).

2.3. Missing Value Imputation

While several different methods are used in the literature for missing value imputation, examples of which include known data regression *KDR* (Arteaga & Ferrer, 2002), conditional mean replacement (Nelson, Taylor, & MacGregor, 1996) iterative algorithm *IA* (Walczak & Massart, 2001), and modified nonlinear iterative partial least squares regression algorithm *NIPALS* (Geladi & Kowalski, 1986), *NIPALS* and *IA* can be considered as the two most widely used algorithms in this category (Folch-Fortuny, Arteaga, & Ferrer, 2017).

2.4. Data set

The PHME21 data challenge data set targets classification of system state and detection of faults, as well as performing root cause analysis for a manufacturing production line, illustrated in Figure 2 (PHME, 2021). The main objectives of this competition were to:

- Determine different faults
- Identify anomalous behavior in signals
- Rank input attributes to determine the key predictors
- Identify fault family from an unlabeled data stream using the minimum number of data points
- Identify system parameter configuration

The data set contains readings from 50 signals, aggregated at 10-second intervals, encompassing stationary and moving components of the line. According to the (PHME, 2021), the data provided as part of the PHME21 data challenge include signals in three categories:

- Machine health (e.g., Pressure, Vacuum, FuseHeatSlope)
- Environmental factors (e.g., Temperature, Humidity)
- Miscellaneous (e.g., CPUTemperature)

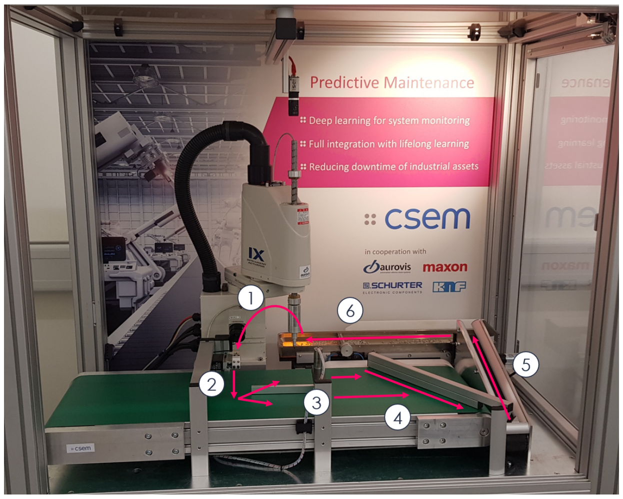

Predictive Maintenance
Deep learning for system monitoring
Full integration with lifelong learning
Reducing downtime of industrial assets
csem
in cooperation with
maxon
SCHURTER
1
2
3
4
5
6

Figure 2. Production line setup: (1) A robotic arm picks up the fuses; (2) A thermal camera checks fuses for overheating or degradation; (3) A conveyor belt carries fuses to a robotic sorting bar; (4) Sorted fuses travel down the line on a conveyor belt; (5) Fuses travel back to the start of the line via a small conveyor belt; (6) fuses are stored to be fed back to the line.

The data were captured in a controlled environment and was free of any faults; however, modifications were made to the system to manufacture faults that resemble the real-life faults in the line (such as failures encountered in the testing machine or conveyor belts) (PHME, 2021). In total, the data set contains 99 experiments, each run from one to three hours, with 70 of them resembling healthy operation and 29 of them signifying eight classes of faulty conditions. Out of the 50 attributes, 14 have no values, and five attributes contain more than 72% null values. It should be noted that only 5.57% of the data contains readings for all attributes. Table 2 details different classes of operation and their corresponding number of experiments.

3. METHODOLOGY

The present work introduces a framework consisting of six components to identify and classify anomalous signals in spite of missing values and partially labeled data (Figure 3), while offering explainability and high accuracy. Table 1 shows the list of components in the proposed framework and the purpose of each component.

3.1. Data preparation

The data provided as part of the PHME21 data challenge is aggregated over 10-second intervals; thus, an initial averaging was already performed on the raw input data. Additionally, a Gaussian filter with a 15-cycle rolling window was applied to reduce noise.

The generally accepted practice of data transformation, on the one hand, is to standardize the data (i.e., mean of the data is subtracted from each data point; thus, the set has a mean of zero) and subsequently normalized (i.e., transformed into a 0 - 1 range). On the other hand, unseen real data may include values that are less than the minimum or greater than the maximum values observed during training and testing. As a result, scaling of input data may have inadvertent outcomes where transformed data becomes less than 0 or greater than 1. To avoid such an obstacle, the present work opts out of scaling raw data or any transformations that otherwise needed prior knowledge of the full breadth of the data. Instead, we tried to use algorithms that were tolerant of the raw data and the possibility of having data that exceed observed values.

Table 1. Framework components and their purpose

Component name	Purpose
PLS-MV	Missing-value imputation
Denoiser	Noise removal
FCM	Generating labels for the unlabeled healthy operations
KPI component	Extracting key performance indicators before classification
Classifier (HistGB)	Detecting and classifying the faulty patterns
SHAP	Explaining the anomalous signal/feature for each class of fault

Table 2. Number of recordings for each fault class

Operation Class	Number of Operation
0 (Healthy)	70
2 (Faulty)	4
3 (Faulty)	4
4 (Faulty)	3
5 (Faulty)	4
7 (Faulty)	4
9 (Faulty)	4
11 (Faulty)	3
12 (Faulty)	3

3.2. Healthy operation clustering

Based on the information provided by the data challenge team, the healthy operations were captured from two different system settings that caused intentional differences in the system behavior, but still reflected healthy operations. This study uses the FCM algorithm to cluster multi-dimensional data and detect the two different operational settings in healthy operations. FCM is an unsupervised clustering method that allows a data point to belong to two or more clusters by assigning membership to each data point (Dunn, 1973). The closer the data is to the cluster center, the higher its membership grade for that particular cluster. We used the scikit-fuzzy implementation of this algorithm and evaluated the performance of FCM before and after imputation of missing values. The Fuzzy Partition Coefficient (FPC) is a performance metric defined from 0 to 1, with one being the best performance (Bezdek, 2013). A maximized FPC means that the FCM had the best performance in describing the data.

3.3. Missing value imputation

We used the NIPALS algorithm as the foundation of the partial least squares (PLS) (Geladi & Kowalski, 1986) in our approach, illustrated in Figure 4. We introduce an ensemble method combining conditional mean replacement (Nelson et al., 1996) and principal component analysis model building with missing data (Folch-Fortuny et al., 2017) to fill the missing values. To avoid introducing errors to the predictive models due to biased imputed values (Harel & Zhou, 2007), our algorithm starts with the attribute that has the minimum number of null values and sets it as the *TARGET* attribute. Next, a subset of data without missing values is selected, such that it can explain the *TARGET* attribute by PLS and fill the Missing Values (PLS-MV). During the training phase of PLS-MV, the algorithm drops any null values (thus using a complete and clean set of *INPUT* attributes and the *TARGET* value). It then trains the PLS regressor on the *INPUT* attributes, aiming to explain the *TARGET* attribute.

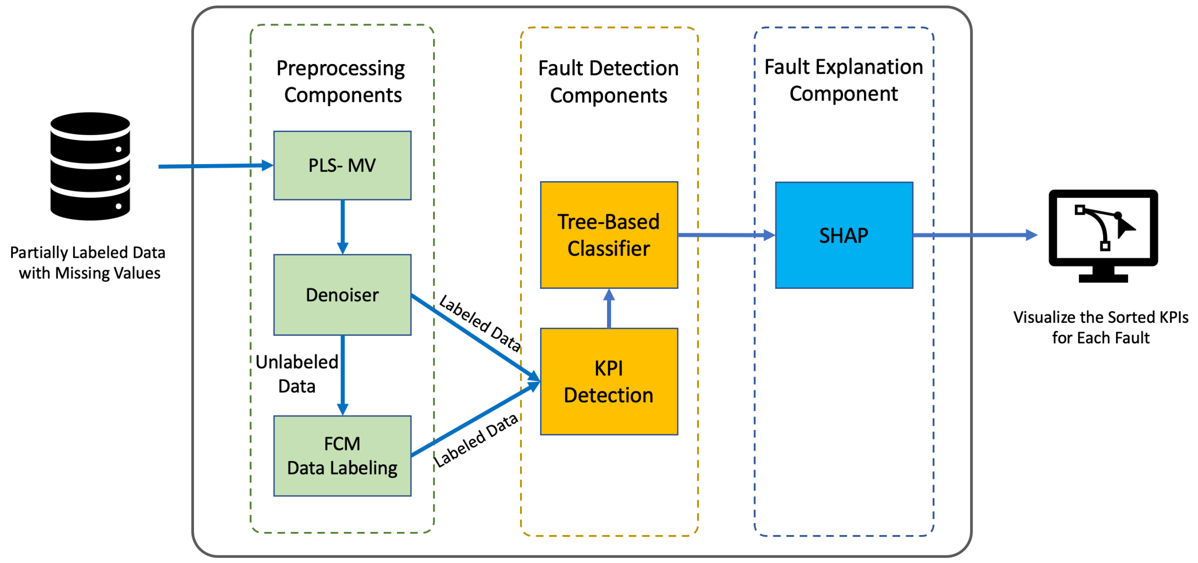

Partially Labeled Data with Missing Values
Preprocessing Components
PLS- MV
Denoiser
Unlabeled Data
FCM Data Labeling
Labeled Data
Labeled Data
Fault Detection Components
Tree-Based Classifier
KPI Detection
Fault Explanation Component
SHAP
Visualize the Sorted KPIs for Each Fault

Figure 3. Proposed framework for explainable fault detection and identification in noisy, partially labeled, and containing missing value data

If PLS fails to give a proper fit (e.g., score or R^2 value less than a user defined threshold), an alternative imputation algorithm is used. It was determined that the backward linear interpolation was the most suitable alternative imputation algorithm for the present data set. The trained *PLS-MV* is used to fill in the missing values for the *TARGET* attribute (the attribute with the least number of null values). This approach results in a new set of data, with an additional attribute that does not have any missing information. The new set of non-null attributes is used in a subsequent *PLS-MV* iteration to fill the next attribute with the minimum number of missing values, and the cycle continues until none of the attributes have missing values.

3.4. Key performance indicators

Key Performance Indicators (KPI) are important attributes that play a critical role in a predictive model. Permutation Feature Importance (PFI) of the RF classifier was used as the basis for initial KPI identification. The PFI is model agnostic and denotes the change in the overall score of the model as an attribute is randomly changed (Breiman, 2001). To determine the most impactful attributes, we limited the number of attributes to those with a positive PFI score, greater than a user-defined threshold. After each elimination of low performers, the RF classifier was run again, and the KPIs were examined until all remaining attributes produced acceptable PFI scores. A series of iterative eliminations was performed to remove low-importance attributes.

3.5. Classification and root cause analysis

Different algorithms including k-nearest neighbors (k-NN), linear and nonlinear support vector machine (SVM), logistic regression, Gaussian Naive Bayes, Decision Trees (DT), and Random Forest (RF) were tested to implement a robust classifier. Additionally, Bagging and Boosting ensemble techniques were evaluated. Among the ensemble boosting techniques, AdaB, LightGB (Ke et al., 2017b) and HistGB (Ke et al., 2017b) were applied. It should be noted that balanced or imbalanced datasets may result in overfitting or biased results in some of the classifiers (e.g., the neural network-based classifiers); however, use of the bagging or boosting approach used in the present work eliminates the risk of overfitting (Ghojogh & Crowley, 2019).

Several approaches are used to analyze the importance of attributes in decisions made by the trees; these include tracing the path taken for each data point (local explanation) or via multiple executions of the model with model agnostic algorithms that examine variations in the input attributes and the resulting predictions. This goal can also be achieved heuristically with the use of tree interpreters (Lundberg et al., 2020). However, the optimal way of explaining the importance of features is considered to be the Shapley Values (Lundberg & Lee, 2017). Borrowing from the original concept proposed by Shapley on cooperative game theory (Shapley, 1951), SHAP (SHapley Additive exPlanations) values are used to interpret the model outputs (Lipovetsky & Conklin, 2001) and determine the importance of each attribute in explaining each class type.

4. RESULTS AND DISCUSSION

This study was limited to the *"Value"* of attributes, and as a result, any attributes that did not possess values (e.g., attributes that only had counts) were dropped. As discussed

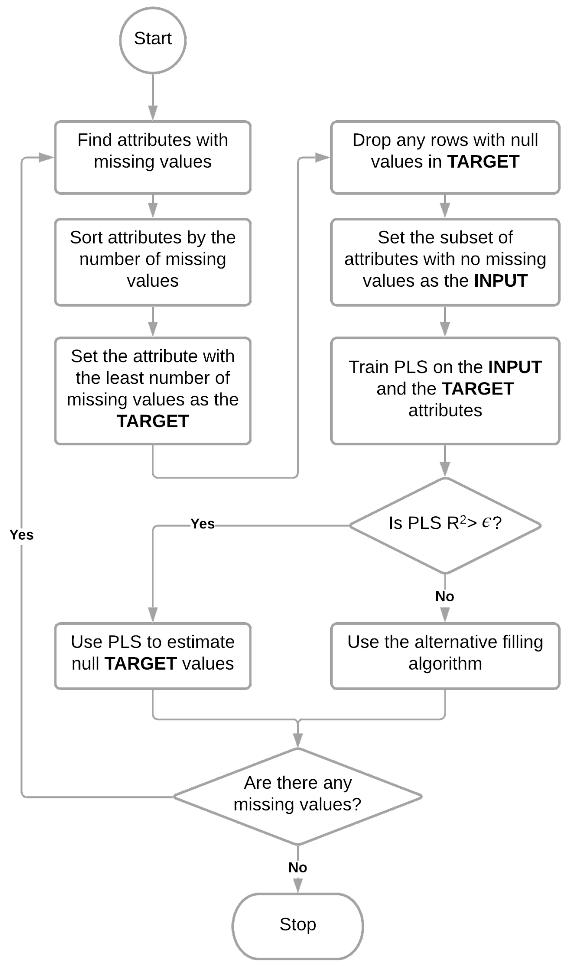

Start
Find attributes with missing values
Sort attributes by the number of missing values
Set the attribute with the least number of missing values as the TARGET
Drop any rows with null values in TARGET
Set the subset of attributes with no missing values as the INPUT
Train PLS on the INPUT and the TARGET attributes
Is PLS R2> ϵ?
Yes
No
Use PLS to estimate null TARGET values
Use the alternative filling algorithm
Are there any missing values?
Yes
No
Stop

Figure 4. Proposed PLS-MV missing value imputation algorithm

earlier, the healthy portion of the data (class 0) pertains to two system settings; however, it is desirable to differentiate between the two settings. The FCM algorithm applied to the raw members of class 0 resulted in an FPC score of 0.5; however, application of the PLS-MV algorithm with backwards linear interpolation, which resulted in a clean database with no missing values, raised the FPC score to 0.9930. Subsequently, a new class 1 was extracted from the original class 0. It should also be noted that the original data did not include any data for classes 6, 8 or 10.

Reduction of attributes was achieved through successive elimination of low performing PFIs (section 3.4). Table 3 lists the KPIs selected for subsequent classification analysis.

We kept 20% of the input data for testing and limited the training data to 80% of the raw data, with the exception of the LightGB algorithm which used a 10-fold cross-validation. Accuracy, precision and recall of each classifier were examined, and to maintain conciseness, we ignored classifiers that resulted in accuracies less than 80%.

Table 3. KPIs selected for subsequent analysis

Attributes		
DBF	DFB	DPTP
FP	P	SI
SMPE	SMS	V
VFP	VVC	

All classifier algorithms were run on an 80 core CPU workstation with 128 GB RAM. The runtime of each algorithm is reported as RT. Where applicable, the maximum tree depth was set to 11, which is the same as the number of selected KPIs. It should be noted that lower depths resulted in less accuracy, and higher depths resulted in large trees, which are computationally expensive in boosting algorithms that sequentially iterate to boost the accuracy.

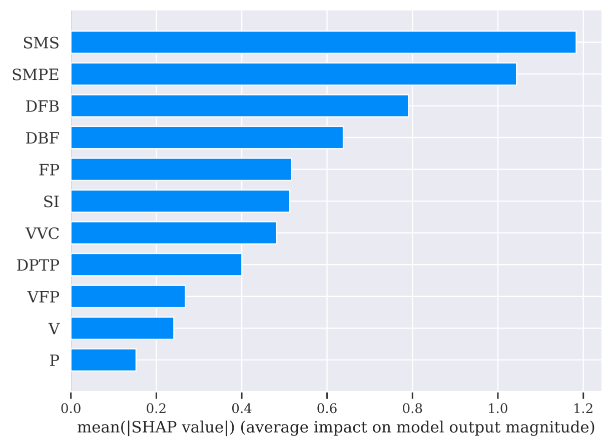
Table 4 shows the training and test Accuracies (Acc.), Precision (Prec.) and Recall (Rec.) for algorithms that resulted in accuracies over 80%. Since the present dataset includes multi-label targets, macro averaging was used to calculate the metrics for each label, and find their unweighted mean. Additionally, Figures 6, 7, 8, and 9 show the confusion matrix of the top four classifier algorithms used in this study.

As shown in Figure 6, the RF algorithm (RT = 1.38s) failed to capture class 4, and its performance in capturing classes 2 and 7 was subpar. The HistGB algorithm (RT = 1h 30min 57s), Figure 7, improved identification of classes 2 and 7, but its long training execution time and inability to capture class 4 are considered as negative factors. The LightGB algorithm (RT = 17min 28s), Figure 8, offered a similar performance to that of the histGB, except for class 4, at a more reasonable training time.

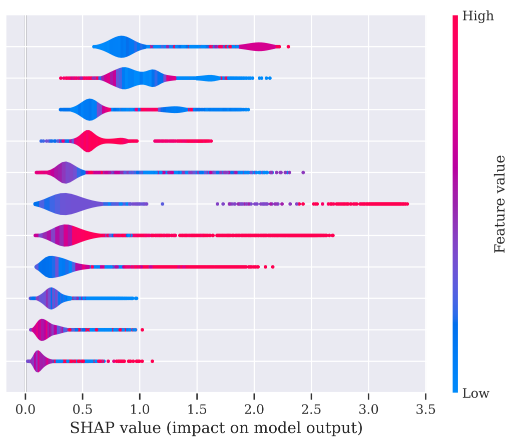
The AdaB algorithm (RT = 43S), Figure 9, outperformed the previous two boosting techniques, while offering a slightly better accuracy compared to the other ensemble models; however, the inability of AdaB to be efficiently explained by SHAP renders it less useful than the other algorithms. To overcome this, the HistGB classifier combined with SHAP values was used to determine the most important features for the test data set.

Table 4. Accuracy (Acc.), Precision (Prec.) and Recall (Rec.) scores for training and test sets of selected models.

Model	**Training Scores**			**Test Scores**		
	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.
DT	0.975	0.980	0.946	0.967	0.965	0.931
RF	0.974	0.994	0.929	0.970	0.989	0.916
LightGB	1.0	1.0	1.0	0.994	0.993	0.987
AdaB	1.0.	1.0	1.0	0.995	0.999	0.986
HistGB	0.999	0.999	0.999	0.996	0.996	0.991



(a) Global



(b) Local

Figure 5. Comparison of Global (a) and Local (b) SHAP values summary plot for the histGB decision tree for all class types.

Additionally, the HistGB classifier was capable of offering robust classifications, as it handled missing values in the real-time prediction of data stream feeding to the model, while avoiding over-fitting.

Figure 5 shows the SHAP for the HistGB decision tree. Compared to Global feature importance (Figure 5-a), using the SHAP summary plot or local importance (Figure 5-b) makes it possible to expand the singular magnitude of feature importance, and explain the effect of individual data points on the overall trend. The local importance shows how positive or negative changes in the magnitude of an attribute changes the target value, thus changing direction of the model's behavior (Lundberg et al., 2020).

The SHAP summary plot of Figure 5-b shows non-linear behavior among the data. Large values of the SMS attribute can have a positive impact on the target value, while the SMPE attribute shows the opposite behavior. Large values of SI show significant impact on the target value, and while VVC is exhibiting a very nonlinear trend, its large values have a great impact on the target value as well.

Table 5 lists the ranking of features for each class, in decreasing order as determined by SHAP. As an example, the main differentiating factors between the healthy sub-classes (i.e., classes 0 and 1) are DBF and SMS.

As evidenced here, the boosting algorithm is best suited for creating a generalized model that is capable of classifying faulty patterns in multi-label datasets, with the possibility of encountering missing values.

5. CONCLUSION

The present work introduced a new framework to identify the KPIs for each type of fault and classify faulty patterns in proactive maintenance. It also introduced a novel missing value imputation approach, PLS-MV, which is capable of deducing missing values based on the trend observed in other similar attributes and data points. The proposed workflow was applied to the PHME21 data challenge data set. Since the healthy portion of the operations was recorded under two different sets of operating conditions, FCM was employed to subdivide the healthy class 0 into two classes that reflected different operating settings.

While different classifier algorithms were investigated, it was determined that boosting algorithms are best suited for the classification of faults in this type of application, and to overcome missing values in test sets.

We were able to achieve a very high accuracy of 0.999 in identifying different classes with tree based boosting, compared to other classifiers such as the Bagging algorithm. However, the non-linear trend of data, and the close similarity of one of the classes (class 4) to a subset of healthy classes (class 0), resulted in subpar classification of this class.

Although the HistGB algorithm showed the longest training time, it performed equally well as the AdaB algorithm; however, AdaB could not be used with the SHAP explainer, and we resorted to the HistGB algorithm to perform root cause analysis, as well as the identification of the feature ranking for each class.

Lastly, we conclude that the boosting algorithm is best suited for creating a generalized model that is capable of classifying faulty patterns in multi-label datasets that may include missing values.

Table 5. Feature ranking for each class type using SHAP

Ranking	**Class 0**	**Class 1**	**Class 2**	**Class 3**	**Class 4**	**Class 5**	**Class 7**	**Class 9**	**Class 11**	**Class 12**
1	DBF	SMS	DPTP	SI	SMPE	VVC	FP	SMS	SMPE	SMPE
2	DFB	DFB	V	DPTP	SMS	FP	VFP	SMPE	DFB	DBF
3	SMPE	SMPE	SI	P	VVC	V	VVC	DPTP	SMS	DFB
4	DPTP	DBF	VFP	VVC	V	DPTP	SMPE	VVC	DBF	V
5	FP	P	VVC	DBF	DBF	VFP	V	V	DPTP	VFP
6	SMS	VFP	DFB	V	FP	SMPE	DFB	VFP	V	VVC
7	VFP	VVC	FP	VFP	DFB	SI	DBF	SI	SI	SMS
8	VVC	FP	SMPE	DFB	VFP	DFB	DPTP	DBF	FP	DPTP
9	SI	V	P	SMPE	SI	P	SMS	DFB	VFP	FP
10	V	SI	DBF	SMS	P	DBF	SI	P	VVC	P
11	P	DPTP	SMS	FP	DPTP	SMS	P	FP	P	SI

NOMENCLATURE

DBF	DurationRobotFromTestBenchToFeeder
DFB	DurationRobotFromFeederToTestBench
SMPE	SmartMotorPositionError
DPTP	DurationPickToPick
FP	FusePicked
SMS	SmartMotorSpeed
VFP	VacuumFusePicked
VVC	VacuumValveClosed
SI	SharpnessImage
V	Vacuum
P	Pressure
RF	Random Forest
RT	Run Time (for training the model)
DT	Decision Tree
KPI	Key Performance Indicator
PFI	Permutation Feature Importance
FPC	Fuzzy Partition Coefficient
HistGB	Histogram Gradient Boosting
LightGB	Light Gradient Boosting
AdaB	Adaptive Boosting

REFERENCES

Alfeo, A. L., Cimino, M. G., Manco, G., Ritacco, E., & Vaglini, G. (2020). Using an autoencoder in the design of an anomaly detector for smart manufacturing. *Pattern Recognition Letters*, *136*, 272–278. Retrieved from `https://doi.org/10.1016/j.patrec.2020.06.008` doi: 10.1016/j.patrec.2020.06.008

Aporras. (2016). *What is the difference between Bagging and Boosting?* Retrieved from `https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/`

Arteaga, F., & Ferrer, A. (2002). Dealing with missing data in MSPC: Several methods, different interpretations, some examples. *Journal of Chemometrics*, *16*(8-10), 408–418. doi: 10.1002/cem.750

Bezdek, J. C. (2013). *Pattern recognition with Fuzzy Objective Function Algorithms.*

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2). doi: 10.1023/A:1018054314350

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Dunn, J. C. (1973, 1). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, *3*(3), 32–57. Retrieved from `http://www.tandfonline.com/doi/abs/10.1080/01969727308546046` doi: 10.1080/01969727308546046

Folch-Fortuny, A., Arteaga, F., & Ferrer, A. (2017). PLS model building with missing data: New algorithms and a comparative study. *Journal of Chemometrics*, *31*(7). doi: 10.1002/cem.2897

Freund, Y., & Schapire, R. E. (1997, 8). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1). doi: 10.1006/jcss.1997.1504

Freund, Y., & Schapire, R. E. (1999). *A Short Introduction to Boosting* (Vol. 14; Tech. Rep. No. 5). Retrieved from `www.research.att.com/fyoav,`

Geladi, P., & Kowalski, B. R. (1986). Partial least-squares regression: a tutorial. *Analytica Chimica Acta*, *185*(9), 1–17. Retrieved from `https://linkinghub.elsevier.com/retrieve/pii/0003267086800289` doi: 10.1016/0003-2670(86)80028-9

Geramifard, O. (2013). Hidden Markov Model-based Methods In Condition Monitoring of Machinery Systems.

Ghojogh, B., & Crowley, M. (2019, 5). The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial. , 1–23. Retrieved from http://arxiv.org/abs/1905.12787

Gupta, B., Uttarakhand, P., & Rawat, I. A. (2017). *Analysis of Various Decision Tree Algorithms for Classification in Data Mining* (Vol. 163; Tech. Rep. No. 8).

Harel, O., & Zhou, X. H. (2007, 7). Multiple imputation: Review of theory, implementation and software. *Statistics in Medicine*, *26*(16), 3057–3077. Retrieved from https://onlinelibrary.wiley.com/doi/full/10.1002/sim.2787https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.2787https://onlinelibrary.wiley.com/doi/10.1002/sim.2787 doi: 10.1002/sim.2787

Kamal, A., Mohd, B. I. N., Pedapati, S. R., & Muhammad, M. (2021). Explainable AI (XAI) for PHM of Industrial Asset: A State-of-The-Art, PRISMA-Compliant Systematic Review. (arXiv:2107.03869v1 [cs.AI]). *arXiv Computer Science*. Retrieved from http://arxiv.org/abs/2107.03869?utm_source=researcher_app&utm_medium=referral&utm_campaign=RESR_MRKT_Researcher_inbound

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T. Y. (2017a). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems* (Vol. 2017-Decem, pp. 3147–3155). Retrieved from https://github.com/Microsoft/LightGBM. doi: 10.5555/3294996

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017b). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In I. Guyon et al. (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Kotsiantis, S. B. (2014, 1). Bagging and boosting variants for handling classifications problems: a survey. *The Knowledge Engineering Review*, *29*(1). doi: 10.1017/S0269888913000313

Langone, R., Cuzzocrea, A., & Skantzos, N. (2020, 11). Interpretable Anomaly Prediction: Predicting anomalous behavior in industry 4.0 settings via regularized logistic regression tools. *Data & Knowledge Engineering*, *130*, 101850. doi: 10.1016/J.DATAK.2020.101850

Lee, J., & Scott, L. W. (2006). Zero-breakdown machines and systems: Productivity needs for next-generation maintenance. In *Proceedings of the 1st world congress on engineering asset management, wceam 2006* (pp. 31–43). Springer-Verlag London Ltd. Retrieved from www.imscenter.net doi: 10.1007/978-1-84628-814-2{_}2

Lipovetsky, S., & Conklin, M. (2001). Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, *17*(4), 319–330. doi: 10.1002/asmb.446

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., ... Lee, S.-I. (2019, 5). Explainable AI for Trees: From Local Explanations to Global Understanding. Retrieved from http://arxiv.org/abs/1905.04610

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., ... Lee, S.-I. (2020). Explainable AI for Trees: From Local Explanations to Global Understanding. *Nature Machine Intelligence*, *2*(1), 56–67. doi: 10.1038/s42256-019-0138-9

Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In *31st conference on neural information processing systems (nips 2017), long beach, ca, usa.* (p. 4768–4777). Long Beach, CA, USA: 31st Conference on Neural Information Processing System (NIPS 2017).

Nelson, P. R., Taylor, P. A., & MacGregor, J. F. (1996). Missing data methods in PCA and PLS: Score calculations with incomplete observations. *Chemometrics and Intelligent Laboratory Systems*, *35*(1), 45–65. doi: 10.1016/S0169-7439(96)00007-X

Peng, C. Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An introduction to logistic regression analysis and reporting. *Journal of Educational Research*, *96*(1). doi: 10.1080/00220670209598786

PHME, E.-p. C. o. t. P. H. M. S. (2021). *Data Challenge – PHME21.* Retrieved from https://phm-europe.org/data-challenge

Rajendran, S., Meert, W., Lenders, V., & Pollin, S. (2019). Unsupervised Wireless Spectrum Anomaly Detection with Interpretable Features. *IEEE Transactions on Cognitive Communications and Networking*. doi: 10.1109/TCCN.2019.2911524

Salfner, F., & Malek, M. (2007). Using hidden semi-Markov models for effective online failure prediction. *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, 161–174. doi: 10.1109/SRDS.2007.4365693

Shapley, L. S. (1951). *Notes on the n-Person Game II: The Value of an n-Person Game* (Vol. RM-670).

Steenwinckel, B., De Paepe, D., Vanden Hautte, S., Heyvaert, P., Bentefrit, M., Moens, P., ... Ongenae, F. (2021, 3). FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning. *Future Generation Computer Systems*, *116*, 30–48. doi:

10.1016/J.FUTURE.2020.10.015

Walczak, B., & Massart, D. L. (2001). Dealing with missing data: Part II. *Chemometrics and Intelligent Laboratory Systems*, *58*(1), 29–42. doi: 10.1016/S0169-7439(01)00132-0

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Qiang, , Hiroshi Motoda, Y. , . . . Motoda, H. (2008). Top 10 algorithms in data mining. *Knowl Inf Syst*, *14*, 1–37. Retrieved from `http://www.cs.uvm.edu/~icdm/` doi: 10.1007/s10115-007-0114-2

APPENDIX

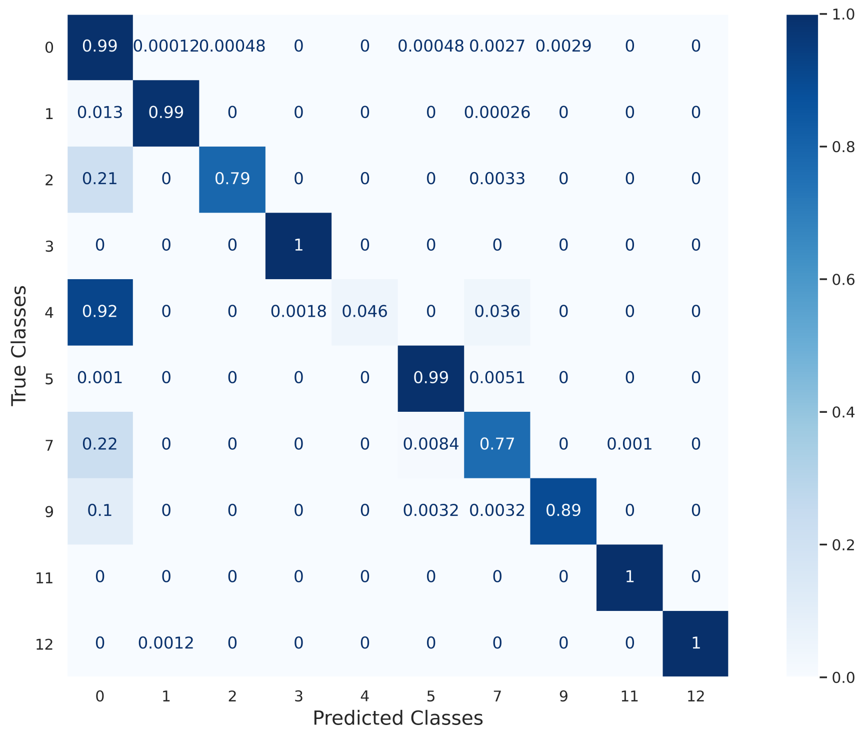


Figure 6. Confusion matrix for Random Forest

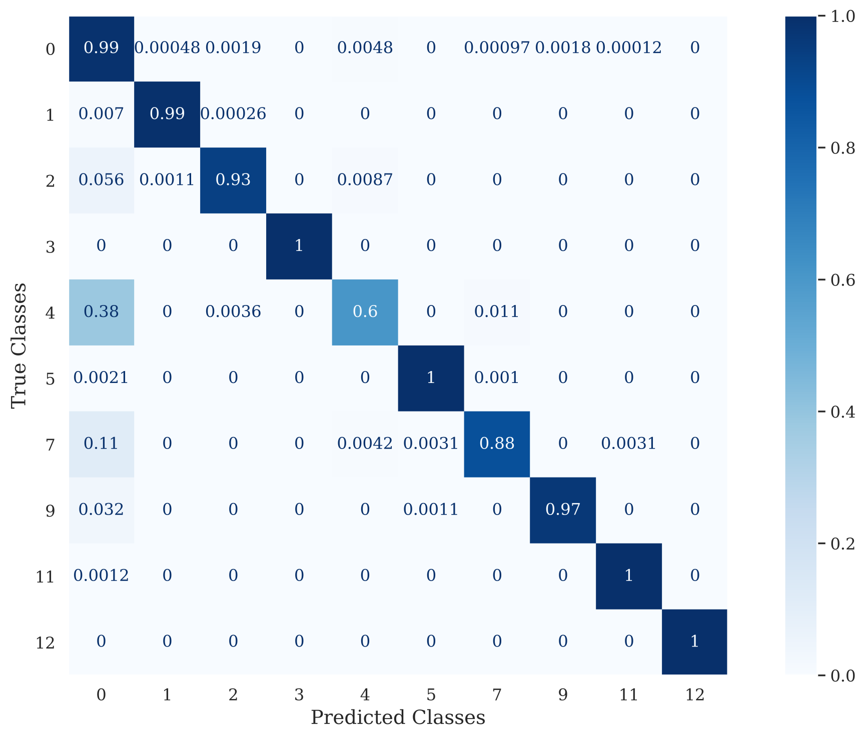


Figure 7. Confusion matrix for HistGB

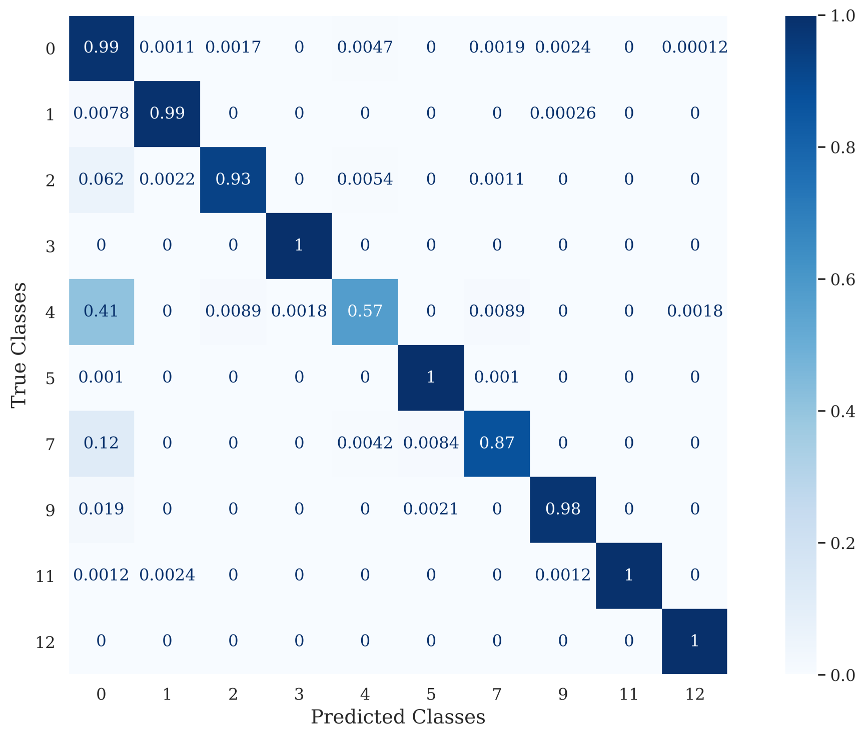


Figure 8. Confusion matrix for LightGB

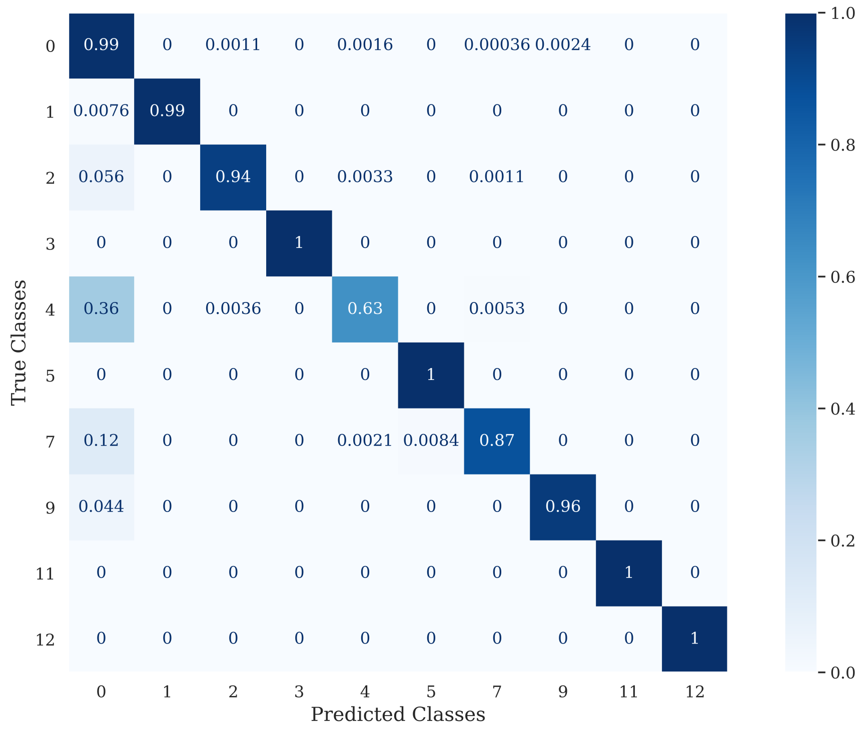


Figure 9. Confusion matrix for AdaB