IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 April 2024, accepted 12 May 2024, date of publication 17 May 2024, date of current version 28 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3402446

RESEARCH ARTICLE

XAI-IoT: An Explainable AI Framework for Enhancing Anomaly Detection in IoT Systems

ANNA NAMRITA GUMMADI, JERRY C. NAPIER, AND MUSTAFA ABDALLAH, (Member, IEEE)
Computer and Information Technology Department, Purdue School of Engineering and Technology, Indiana University-Purdue University Indianapolis (IUPUI), Indianapolis, IN 46202, USA

Corresponding author: Mustafa Abdallah (mabdall@iu.edu)

This work was supported in part by the Lilly Endowment through AnalytixIN and the Wabash Heartland Innovation Network (WHIN), in part by the Enhanced Mentoring Program with Opportunities for Ways to Excel in Research (EMPOWER), and in part by the First Year Research Immersion Program (1RIP) grants from the Office of the Vice Chancellor for Research at Indiana University-Purdue University Indianapolis.

ABSTRACT The exponential growth of Internet of Things (IoT) systems inspires new research directions on developing artificial intelligence (AI) techniques for detecting anomalies in these IoT systems. One important goal in this context is to accurately detect and anticipate anomalies (or failures) in IoT devices and identify main characteristics for such anomalies to reduce maintenance cost and minimize downtime. In this paper, we propose an explainable AI (XAI) framework for enhancing anomaly detection in IoT systems. Our framework has two main components. First, we propose AI-based anomaly detection of IoT systems where we adapt two classes of AI methods (single AI methods, and ensemble methods) for anomaly detection in smart IoT systems. Such anomaly detection aims at detecting anomaly data (from deployed sensors or network traffic between IoT devices). Second, we conduct feature importance analysis to identify the main features that can help AI models identify anomalies in IoT systems. For this feature analysis, we use seven different XAI methods for extracting important features for different AI methods and different attack types. We test our XAI framework for anomaly detection through two real-world IoT datasets. The first dataset is collected from IoT-based manufacturing sensors and the second dataset is collected from IoT botnet attacks. For the IoT-based manufacturing dataset, we detect the level of defect for data from IoT sensors. For the IoT botnet attack dataset, we detect different attack classes from different kinds of botnet attacks on the IoT network. For both datasets, we provide extensive feature importance analysis using different XAI methods for our different AI models to extract the top features. We release our codes for the community to access it for anomaly detection and feature analysis for IoT systems and to build on it with new datasets and models. Taken together, we show that accurate anomaly detection can be achieved along with understanding top features that identify anomalies, paving the way for enhancing anomaly detection in IoT systems.

INDEX TERMS Internet of Things, anomaly detection, explainable AI, SHAP, LIME, Mirai, IoT security, black-box AI, MEMS, and N-BaIoT.

I. INTRODUCTION

The Internet of Things (IoT) represents a network of interconnected objects with internet capabilities, equipped with embedded sensors for data collection and exchange [1]. Any standalone device connected to the internet, capable of remote monitoring and control, is considered an IoT device. Therefore, IoT systems become more complex and are rapidly deployed in different applications [2]. IoT systems pose certain salient technical challenges for the use of AI-based models for anomaly detection. Firstly, in IoT systems, various types of sensors concurrently generate data

The associate editor coordinating the review of this manuscript and approving it for publication was Kashif Sharif.

related to the same (or overlapping) events, each possessing distinct capabilities and costs. Secondly, the characteristics of sensor data undergo changes based on the sensors' operating points, such as the Revolutions Per Minute (RPM) of the motor in IoT-inspired smart manufacturing systems [3], which is a measure of the number of complete revolutions the motor makes in one minute. Consequently, both the inference and anomaly detection processes necessitate calibration according to the specific operating point. To address these requirements, case studies on anomaly detection deployments in such systems become imperative. The importance of these deployments and the subsequent analyses has been emphasized in prior works, including those focused on digital agriculture [4], smart manufacturing [5], IoT-based health monitoring systems [6], and other IoT systems [7], [8].

While there is a considerable body of literature on anomaly detection within various IoT-based systems that focused on traditional AI models for anomaly detection and failure detection in different IoT systems [9], [10], [11], [12], [13], [14], there is a notable gap in the documentation of the use of explainable AI (XAI) methods specifically for anomaly detection in smart IoT systems [15], [16]. In particular, these previous AI-based studies focused more on the classification accuracy of various AI algorithms, for detecting anomalies in IoT systems, without providing insights about their behavior and reasoning about main features. Moreover, they did not explore the model-specific feature importance, combining global and local explanations, and exploring different diverse set of AI models. These limitations motivate the pressing need to leverage the relatively recent XAI field for enhancing anomaly detection for IoT systems [17]. This usage of XAI can also help in the ultimate goal of building predictive maintenance frameworks for these IoT systems.

In this paper, we study the anomaly detection problem of IoT systems by detecting failures and anomalies that would have an impact on the reliability and safety of these systems. In such systems, the data are collected from different sensors via intermediate data collection points and finally aggregated to a server to further store, process, and perform useful data-analytics on the sensor readings [18], [19]. We propose an XAI framework (that we call XAI-IoT) for enhancing anomaly detection in IoT systems. Our XAI framework includes loading the IoT dataset, pre-processing the data, training black-box AI models, generating global and local feature importance graphs, and extracting top features based on different XAI methods. The low-level structure of our proposed framework is shown in Figure 1.

In particular, our framework has two main components: (i) AI-based anomaly detection of the IoT system under consideration, and (ii) feature importance analysis of the main features that help AI models identify anomalies in these IoT systems. For the first component, we consider two classes of anomaly detection models which are single AI models (including decision tree (DT) [20], deep neural network (DNN) [21], AdaBoost (ADA) [22], support vector machine (SVM) [23], and multi-layer perceptron (MLP) [24]), and ensemble methods (including random forest (RF) [25], bagging [26], blending [27], stacking [28], and voting [29]). These models are used to predict the anomalies from data collected in these IoT systems.

For the second component, we perform feature importance analysis using different XAI methods (SHAP [30], LOCO [31], CEM [32], ALE [33], PFI [34], and ProfWeight [35], and LIME [36]). These XAI methods have different methodologies for generating feature importance.

A. SHAP [30]

This popular XAI method facilitates the generation of feature explanations for AI models. It uses the game theory concept of Shapely values to explain an AI model. It generates Shapley values for each feature, and it does that by considering the prediction using all the other features except the one under evaluation. This way SHAP can understand the impact of the contribution from each feature.

B. LEAVE-ONE-COVARIATE-OUT (LOCO) [31]

This XAI strategy is used to estimate the importance of features in an AI model. When a feature is removed from the model, the LOCO technique measures the change in prediction error to assess how important the feature is. In particular, the model is retrained several times, omitting a distinct feature each time, and the effect on the model's performance is tracked. Thus, a feature is meaningful for the model's predictions if omitting it results in a notable change in model's accuracy.

C. CONTRASTIVE EXPLANATIONS METHOD (CEM) [32]

This XAI method shows which features may be altered to get a different prediction output, which sheds light on how an AI method makes decisions. This method can assist in the understanding of not just which features are significant but also how these features could be changed to affect the predictions.

D. ACCUMULATED LOCAL EFFECT (ALE) [33]

The XAI ALE plots are used to illustrate how features, taking into consideration their interactions with other features, impact on AI model's average prediction. They provide insight into the correlation between input features and anticipated prediction.

E. LIME [36]

This widely used XAI tool enables the creation of a surrogate model approximating the original AI model's behavior when assessed with local samples. In this study, we utilized LIME to generate local feature importance.

F. PERMUTATION FEATURE IMPORTANCE (PFI) [34]

this XAI method is another technique used to assess the importance of features. In PFI, a feature's importance is determined by permuting its values and tracking how the

model's performance indicators change as a result. Each feature goes through this procedure several times, and the average performance change indicates how important the feature is.

G. PROFILED WEIGHTING (PROFWEIGHT) [35]

This XAI approach assesses the significance of a feature by considering many criteria, such as the feature's weight within the model and its relationship with other features. This approach offers an indicator of the relative contribution of each attribute to model's predictions.

We apply our framework to study two real-world IoT datasets with different characteristics. The first dataset (that we call MEMS dataset) is a smart manufacturing dataset collected from deployed manufacturing sensors to detect anomalous data readings. The second dataset (that we call N-BaIoT) is an open source IoT dataset where the goal of this dataset is to detect IoT botnet attacks [37]. The N-BaIoT data was gathered from nine commercial IoT devices that were actually infected by two well-known botnets, Mirai [38] and Gafgyt [39]. In our evaluation, we first analyze the performance of our different AI models in detecting anomalies in these two datasets where we measure different standard performance metrics (including accuracy, precision, recall, and F-1). We observe that the best anomaly detection model is dataset-dependent with ensemble methods giving better performance in the anomaly detection task.

One challenge in the MEMS anomaly detection problem is the prediction from AI models using sparse data, which is often the case because of limitations of the sensors or the cost of collecting data. The choice of sensors with lower sampling rates, despite potential drawbacks, is driven by significant cost differences. For instance, MEMS sensors are much more economical ($8) compared to advanced sensors like piezoelectric sensors ($1305) [40], [41]. To tackle this challenge, we leveraged all available data collected under various operating conditions, specifically different RPMs. On the other hand, the challenge in the anomaly detection in N-BaIoT dataset is detecting the different attack classes from different kinds of botnet attacks on the IoT network. To tackle this challenge, we evaluated the performances of different AI models under different combinations of features to identify which setup is more efficient in detecting anomalies in the traffic data collected from this IoT network.

We have also provided extensive feature importance analysis using different XAI methods for our different AI models. For the MEMS dataset, we also provided feature importance for different RPMs considered in collecting MEMS data for more in-depth understanding. In our assessment of feature importance, we validate the notion that vibration data along certain axes may not convey distinct information in normal and failure scenarios. The circular motion around the motor's center occurs along the X and Z axes, resulting in vibration values that vary with the motor's condition. Conversely, the Y-axis, representing the direction of the shaft, exhibits smaller vibrations. To investigate this, we compare the model's performance with features derived from all three data axes in the default setup against a proposed approach where features are extracted exclusively from the X-axis and Z-axis data vectors. For the feature importance analysis on the N-BaIoT dataset, our results show that the features that give precise statistics summarizing the recent traffic from the host to the destination in this IoT network can help in identifying the botnet attack class on the IoT network.

We emphasize that our paper focuses on offering a thorough analysis, and comparative viewpoint of different AI models and Explainable AI (XAI) techniques. It is critical to comprehend the functionality and interpretability of these models in an era where artificial intelligence (AI) systems are becoming more and more complicated. By means of thorough experimentation and comparative analysis, our objective is to highlight the performance of various artificial intelligence methodologies and XAI approaches. Our research aims to provide practitioners and researchers in IoT security domain with essential insights for responsible AI deployment and decision-making by examining their effectiveness in a variety of datasets and use scenarios in two IoT systems.

Summary of Contributions: Based on our analysis and evaluation, we have the following contributions:

1) **XAI Framework:** Our contribution consists in the creation of an XAI framework specifically designed for anomaly identification in IoT systems. We offer a complete toolset for doing feature importance analysis by combining seven distinct XAI techniques, including SHAP, LIME, CEM, ALE, PFI, Profweight, and LOCO. This approach provides feature importance on both global and local scopes, which is important for comprehending how complicated AI models for anomaly detection in IoT applications make decisions. In our feature importance analysis, we extract the model-specific features (i.e., top important features for each AI model) and anomaly-specific features (i.e., top features for each attack type) for different classes of AI models that we have and different types of anomalies.
2) **Anomaly Detection:** We expand the use of anomaly detection to IoT systems by modifying two categories of models: individual AI techniques (including DT, DNN, SVM, ADA, and MLP) and ensemble techniques (including bagging, blending, stacking, and voting). These models are made to detect abnormal data from network traffic or deployed sensors in an efficient manner, protecting IoT infrastructure security.
3) **Testing:** We rigorously evaluate our system on two real-world datasets from IoT botnet attacks and IoT-based manufacturing sensors. We demonstrate our approach's efficacy in identifying anomalies in various IoT scenarios through comprehensive evaluation, underscoring its potential for practical use.
4) **Defect Type Classification:** We further contribute to the field by offering a way to classify the degree of fault

in IoT-based smart manufacturing data, in addition to anomaly detection. The ability to classify fault degree improves the IoT systems' diagnostic capabilities, allowing for proactive maintenance and quality control procedures.

5) **Benchmark Data and Codes:** In order to support future work in this area, we make our database corpus available, which consists of two different datasets and all developed code scripts. We hope that releasing these tools to the community will help to accelerate progress in anomaly detection and classification for IoT systems by facilitating benchmarking, replication, and extension of our work. https://github.com/agummadi1/XAI_for_IoT_Systems.

Paper Organization: The rest of the paper is organized as follows. We first present the related works in Section II. We then explain our framework, including anomaly detection and feature importance models in Section III. In Section IV, we present our evaluation results of anomaly detection models and XAI feature importance on our two IoT datasets. We present main limitations of our work and related discussions in Section VII. We conclude the paper in Section VIII.

II. RELATED WORK

A. ANOMALY AND FAILURE DETECTION MODELS IN IoT

Anomaly detection methods have been used for identifying energy consumption anomalies in IoT systems via monitoring and identifying abnormal energy consumption patterns in IoT-connected devices to detect malfunctioning or compromised devices [42]. Furthermore, anomaly detection has been explored for health monitoring IoT systems through detecting anomalies in health-related data collected by IoT devices such as wearables to identify potential health issues or irregularities [6]. Machine learning methods have been used for detecting anomalies in smart agriculture applications with the focus of detecting anomalies in environmental data collected by IoT devices in agriculture to identify potential crop diseases, irrigation issues, or pest infestations [43]. However, these work did not explore XAI feature importance and analysis, and considered different application domains from those in our current work.

Various studies have explored the detection of failures in IoT systems using either single or multiple sensors [44], [45], [46]. The main application in this context is monitoring the behavior of machinery and equipment in industrial settings to detect anomalies, potential faults, or performance deviations. Notably, a recent work [44] proposed a kernel principal component analysis-based anomaly detection system to identify cutting tool failures in a machining process in smart manufacturing system. Although this study utilized multi-sensor signals to assess the cutting tool's condition, it did not address transfer learning between different sensor types. Another recent study [46] introduced a fault detection monitoring system for detecting various failures in a DC motor, including gear defects, misalignment, and looseness. However, this study relied on a single sensor (accelerometer) to collect machine condition data, and it employed several convolutional neural network architectures for targeted failure detection without considering different rotational speeds and sensors. Consequently, these techniques would need reapplication for each new sensor type. In contrast, our approach involves considering learning main features using diverse XAI methods and sensor types, and we conduct a comparative analysis of single and ensemble learning-based models for our anomaly detection task.

B. EXPLAINABLE AI AND FEATURE IMPORTANCE IN IoT

Many studies have been conducted in the field of Explainable Artificial Intelligence (XAI) with the goal of improving machine learning models' interpretability and transparency for different applications [47]. Numerous strategies have been investigated, from decision trees and rule-based systems to more complex methods like SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations). By demystifying complex models' decision-making processes, these techniques aim to increase end users' understanding and confidence.

Feature Selection in IoT: It becomes especially important to integrate feature importance analysis when it comes to IoT systems [48], [49], [50]. In particular, the work [48] provides an extensive overview of feature selection methods in the context of IoT-based healthcare applications. The work [49] proposes a feature selection method for intrusion detection systems (IDSs) for IoT systems but with only focusing on two feature selection methods which are Information Gain (IG) and Gain Ratio (GR). It also focused mainly on the detection of denial of service (DoS) attacks. The work [50] proposes a feature selection algorithm that is based on the concepts of the Cellular Automata (CA) engine and Tabu Search (TS)-based aspiration criteria with main focus on Random Forest (RF) ensemble learning classifier to evaluate the fitness of the selected features. That work also focused on the TON_IoT dataset (created by UNSW in Australia).

The challenges in IoT systems include large volumes of data that are produced by IoT devices. Thus, it is essential to comprehend the fundamental characteristics that influence model predictions in order to maximize system efficiency [51], spot abnormalities, and guarantee the accuracy of decision results. In addition to improving model interpretability for anomaly detection for IoT systems, feature importance analysis makes it easier to identify important variables [52], [53], which helps with the development and implementation of reliable and effective IoT applications. In order to enable accountable and trustworthy AI deployment in a variety of applications, researchers are working to build approaches that strike a compromise between model

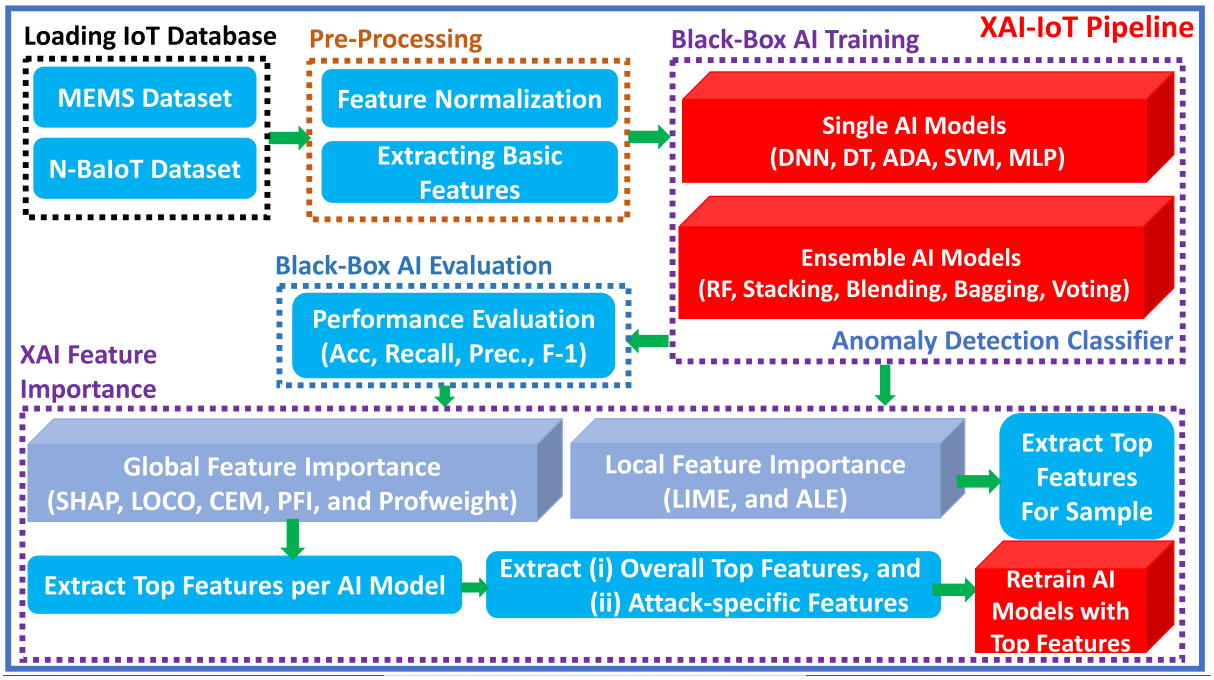


FIGURE 1. The proposed XAI-IoT framework. It has the following steps: loading IoT database, preprocessing data, black-box AI training, black-box AI evaluation, XAI feature importance (including global and local scopes), and retraining AI models.

precision and transparency as the convergence of XAI and IoT expands.

C. DATASETS AND BENCHMARKS FOR ANOMALY DETECTION IN IoT SYSTEMS

Several papers have concentrated on providing datasets for anomaly detection in IoT systems, particularly emphasizing the unsupervised anomaly detection process [54], [55]. For instance, the study by Koizumi et al. [54] presents benchmark results from the *DCASE 2020 Challenge Task*, focusing on unsupervised detection of anomalous sounds for machine condition monitoring in smart manufacturing. This work specifically aims to determine whether the sound emitted from a target machine is normal or anomalous, addressing the challenge of anomalous sound detection (ASD). Additionally, another work by Hsieh et al. [55] introduces an unsupervised real-time anomaly detection algorithm tailored for smart manufacturing. In contrast, the study by Fathy et al. [56] delves into learning techniques for failure prediction using imbalanced smart manufacturing datasets. However, none of these works addresses the crucial aspect of feature importance via different XAI methods, which is a focal point in our investigation. The work [57] proposed an open-source IoT framework for sensor networks management in smart cities, which is a different application domain from the two IoT domains considered in our current work.

III. MATERIALS AND METHODS

We now describe our proposed framework. This framework mainly consists of algorithms for the anomaly detection, defect type (or attack) classification, and feature importance. We now explain the low-level components of our XAI pipeline. The different components of our framework (shown in Figure 1) are explained below.

Loading IoT Database: The first component in our pipeline is loading the IoT data from the database as a starting point. In our work, we use two IoT datasets which are MEMS [43], and N-BaIoT [37] datasets.

Pre-Processing: The second component in our framework is preprocessing in which we prepare the dataset for the anomaly classification task. In particular, such preprocessing is essential for building AI models for the anomaly detection task. We followed the prior works [43] for extracting the basic set of features for MEMS and N-BaIoT datasets, respectively. We also emphasize that we take advantage of our XAI-based feature selection to identify top features that affect the decisions of different AI models.

Feature Normalization: In order to prevent variations in scales among different features, we apply a standard feature normalization step (min-max feature scaling) for all columns in our datasets (where we apply feature scaling for each column; one column at a time to address inconsistencies across different features' scales). This process ensures that all features are brought to a consistent numerical scale, thereby avoiding any discrepancies in magnitude across the dataset. Such a process has been applied in several prior works [58], [59], [60], [61].

Black-Box AI Models: Once the preprocessing of the database is complete, we train the AI model where we perform splitting of the data with a split of 70% for the

training while leaving the unseen 30% for testing purposes. For this part, we have built ten popular AI classification models. They can be classified into the following two classes:

- **Single AI Models:** In this category, we included decision tree (DT) [20], deep neural network (DNN) [21], AdaBoost (ADA) [22], support vector machine (SVM) [23], and multi-layer perceptron (MLP) [24]).
- **Ensemble Models:** We selected five popular ensemble methods where we included RF [25], bagging [26], blending [27], stacking [28], and voting [29].

These models are used to predict the anomalies from sensors' readings in IoT systems, along with the attack (or defect) type. These models were also used for predictive maintenance for the MEMS IoT dataset, i.e., predicting the level of defect with the MEMS sensor and whether the machine is in normal operation, near-failure (and needs maintenance), or failure (and needs replacement). On the other hand, the models were used to detect anomaly classes in N-BaIoT dataset, i.e., predicting whether the traffic is benign, or is related to one of the 10 attack classes that represent different attack tactics employed by the Gafgyt [39] and Mirai [38] botnets to infect IoT devices.

For each model, we generated multiple variants by changing the values of hyperparameters. We then chose the model variant with the best performance for each dataset. We describe the hyper-parameters and the libraries used for all forecasting models in Appendix B.

Black-Box AI Evaluation: The next step in our framework is to study the performance of each model on unseen test data. For such performance analysis, we first create the confusion matrix for each model and use it to derive the following different metrics for each AI model: accuracy (Acc), precision (Prec), recall (Rec), F1-score (F1), Matthews correlation coefficient (Mcc) [62], balanced accuracy (Bacc), and the area under ROC curve (AucRoc). We selected these metrics for two primary reasons. Firstly, they are commonly used in numerous comparable works that emphasize IoT systems, such as [37] and [43] for our two datasets. Secondly, our choice enables an examination of the impact of XAI-based feature selection on the performance of AI models in IoT datasets. This allows for direct comparisons with previous studies conducted on the two IoT datasets under consideration.

XAI Global Explanations: The aforementioned AI models operate as black-box models. Consequently, it becomes imperative to provide explanations for these models, elucidating their accompanied features (primary IoT sensors or IoT network traffic) and labels (attack types). In the subsequent phase of our framework, we incorporate Explainable AI (XAI). In the initial segment of this step, we generate global importance values for each feature, creating various graphs to analyze the influence of each feature on the AI model's decisions. This analysis aids in forming expectations about the model's behavior. For global explanations, we employ various XAI methods, including SHAP, LOCO, CEM, PFI, and ProfWeight.

XAI Local Explanations: Our framework consists of two local XAI blocks. First, we use the recent well-known local interpretable model-agnostic explanations (LIME) [36] for giving insights of what happens inside an AI algorithm by capturing feature interactions. We first generate a model that approximates the original model locally (LIME surrogate model) and then generate the LIME local explanations. Second, we leverage ALE [33] via generating ALE local graphs (named as ALE plots).

Feature Explanation: The final component in our framework is extracting detailed metrics from the global explanations. In particular, we extract the model-specific features (i.e., top important features for each AI model) and anomaly-specific features (i.e., top features for each anomaly (or attack type) for different classes of AI models that we have and different types of anomalies. This additional feature analysis can help in providing human-understandable explanations of the decision-making of the AI model and accompanied top features.

Feature Explanation Importance: One of the important outcomes of our framework is generating the list of top features for each IoT dataset. This can help the security analyst managing the IoT network in detecting attacks and anomalies (either from IoT sensors (MEMS) readings or network traffic between IoT devices (N-BaIoT)). In our current work, we are focusing on understanding in more depth the feature importance for each model for the two considered datasets (MEMS, and N-BaIoT).

Summary and Statistics of the Datasets: Table 1 shows the number of samples for the two datasets and the distribution of samples per attack type. Note that MEMS dataset contains only three classes (near failure, failure, and normal). On the contrary, N-BaIoT has ten attack classes (five Gafgyt botnet attacks, and five Mirai botnet attacks).

Having introduced the background and the low-level details for the proposed framework, we next provide our main evaluation results for anomaly detection and feature importance tasks on our two IoT datasets.

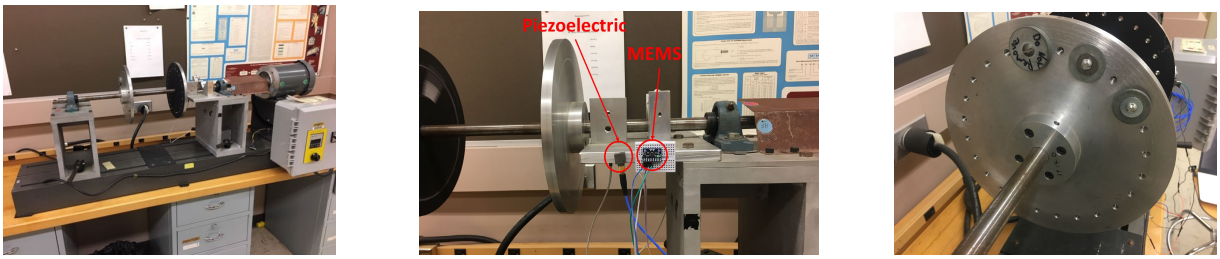
TABLE 1. Summary and statistics of the two IoT datasets used in this work, including the size of the dataset, the number of attack types (labels), and the number of intrusion features).

Dataset	Number of Labels	Number of Features	Number of Samples
MEMS	3	3	21, 577
N-BaIOT	11	115	1,681,000

IV. EXPERIMENTAL SETUP EXPLANATION

In our evaluation, we seek to answer the following five research questions:

- What is the performance of black-box AI models on the two considered IoT datasets?
- How can we detect the operational state of IoT dataset effectively (i.e., with high accuracy)?



(a) Motor Testbed. (b) Piezoelectric and MEMS sensors. (c) Balancing disk.

FIGURE 2. Motor testbed and sensors for smart manufacturing dataset. In (a), we show the testbed. In (b), we show piezoelectric and MEMS sensors when mounted on motor testbed. In (c), we show the balancing disk to make different levels of imbalance.

- What are the main features that affect the performance of different AI models in our IoT datasets?
- What are the main features for each attack (anomaly) type in the two IoT datasets?
- How does XAI help in understanding the performance of different AI models for single data instance?

A. DEPLOYMENT DETAILS AND DATASETS EXPLANATION

1) MEMS DATASET

a: MAIN GOAL OF USING THIS IoT DATASET

Anomalous data generally needs to be separated from machine failure as abnormal patterns of data do not necessarily imply machine or process failure [5]. We perform anomaly detection using vibration data to identify anomalous events and then attempt to label/link these events with machine failure information. This way, we aim to identify abnormal data and correlate the abnormal data to machine failure coming from IoT manufacturing sensors. To achieve such a goal, we build anomaly detection models to detect anomalies and failures in the IoT sensors.

b: DATASET CONSTRUCTION AND FEATURES

To construct this dataset, an experiment was carried out in the motor testbed, as depicted in Figure 2a, to gather machine condition data, specifically acceleration, under various health conditions. During the experiment, acceleration signals were acquired from MEMS sensors (Figure 2b) simultaneously, with a sampling rate of 10 Hz for the X, Y, and Z axes. Different levels of machine health conditions were induced by affixing a mass to the balancing disk (i.e., mounting a mass on the balancing disk, thus different levels of mechanical imbalance are used to trigger failures), as illustrated in Figure 2c, thereby generating varying degrees of mechanical imbalance to initiate failures. Failure conditions were categorized into one of three states: normal, near-failure, and failure.

c: OPERATIONAL SPEEDS

Acceleration data were captured at ten rotational speeds (100, 200, 300, 320, 340, 360, 380, 400, 500, and 600 RPM) for each condition while the motor was operational. Fifty samples were collected at 10-second intervals for each of the ten rotational speeds. This same dataset was utilized for both defect-type classification and feature importance tasks, as elaborated in Section V.

d: ANOMALY (DEFECT) CLASSES

The data contains different levels of defects (i.e., different labels for indicating normal operation, near-failure, and failure for MEMs dataset). These labels would be used in our evaluation.

2) N-BaIoT DATASET

The goal of this dataset is to detect IoT botnet attacks [37]. This dataset is a useful resource for researching cybersecurity issues in the context of the Internet of Things (IoT). This data was gathered from nine commercial IoT devices that were actually infected by two well-known botnets, Mirai [38] and Gafgyt [39].

(i) Details of Nine Devices: We first describe briefly each device of the nine devices used for data collection.

Device 1 - Danmini Doorbell: A smart doorbell that integrates intercom and camera systems for maintaining home security and communications systems.

Device 2 - Ecobee Thermostat: An intelligent thermostat that learns the user's preferences and modifies the heating or cooling to maximize energy savings.

Device 3 - Ennio Doorbell: A doorbell system with video features, giving visual identification of guests for better home security.

Device 4 - Philips B120N10 Baby Monitor: A baby monitor that has audio and video features so that parents can closely monitor their newborns.

Device 5 - Provision PT 737E Security Camera: A security camera with remote monitoring and motion detection capabilities that is intended for surveillance.

Device 6 - Provision PT 838 Security Camera: Another kind of security camera, with expanded functions for monitoring and surveillance purposes.

Device 7 - Samsung SNH 1011 N Webcam: A Samsung webcam utilized for video conferences or basic video recording.

Device 8 - SimpleHome XCS7 1002 WHT Security Camera: A security camera from SimpleHome, ideal for effortless home surveillance and monitoring.

Device 9 - SimpleHome XCS7 1003 WHT Security Camera: Another security camera for home surveillance with a few advanced features.

(ii) Main Features: Every data instance in the dataset is represented by a variety of features. These attributes are divided into multiple groups, which are detailed as follows:

(A) Stream Aggregation: These functions offer data that summarizes the traffic of the past few days. This group's categories comprise the following features:

H: Statistics providing an overview of the packet's host's (IP) recent traffic.

HH: Statistics providing a summary of recent traffic from the host (IP) of the packet to the host of the packet's destination.

HpHp: Statistics providing a summary of recent IP traffic from the packet's source host and port to its destination host and port.

HH-jit: Statistics that summarize the jitter of the traffic traveling from the IP host of the packet to the host of its destination.

(B) Time-frame (Lambda): This characteristic indicates how much of the stream's recent history is represented in the statistics. They bear the designations L1, L3, L5, and so forth.

(C) Features Taken Out of the Packet Stream Statistics: Among these characteristics are the following features:

Weight: The total number of objects noticed in recent history, or the weight of the stream.

Mean: The statistical mean is called the mean.

Std: The standard deviation in statistics.

Radius: The square root of the variations of the two streams.

Magnitude: The square root of the means of the two streams.

Cov: A covariance between two streams that is roughly estimated.

Pcc: An approximate covariance between two streams.

(iii) Classes: The dataset consists of the following 11 classes: benign traffic which is defined as network activity that is benign and does not have malicious intent, and 10 of these classes represent different attack tactics employed by the Gafgyt and Mirai botnets to infect IoT devices. The classes are summarized as follows:

1. benign: There are no indications of botnet activity in this class, which reflects typical benign network traffic. It acts as the starting point for safe network operations.

2. gafgyt.combo: This class is equivalent to the ''combo'' assault of the Gafgyt botnet, which combines different attack techniques, like brute-force login attempts and vulnerability-exploiting, to compromise IoT devices.

3. gafgyt.junk: The ''junk'' attack from Gafgyt entails flooding a target device or network with too many garbage data packets, which can impair operations and even result in a denial of service.

4. gafgyt.scan: Gafgyt uses the ''scan'' attack to search for IoT devices that are susceptible to penetration. The botnet then enumerates and probes these devices in an effort to locate and compromise them.

5. gafgyt.tcp: This class embodies the TCP-based attack of the Gafgyt botnet, which targets devices using TCP-based exploits and attacks.

6. gafgyt.udp: The User Datagram Protocol (UDP) is used in Gafgyt's ''udp'' assault to initiate attacks, such as bombarding targets with UDP packets to stop them from operating.

7. mirai.ack: To take advantage of holes in IoT devices and enlist them in the Mirai botnet, Mirai's ''ack'' attack uses the Acknowledgment (ACK) packet.

8. mirai.scan: By methodically scanning IP addresses and looking for vulnerabilities, Mirai's ''scan'' assault seeks to identify susceptible IoT devices.

9. mirai.syn: The Mirai ''syn'' attack leverages vulnerabilities in IoT devices to add them to the Mirai botnet by using the SYN packet, which is a component of the TCP handshake procedure.

10. mirai.udp: Based on the UDP protocol, Mirai's ''udp'' attack includes bombarding targeted devices with UDP packets in an attempt to interfere with their ability to function properly.

11. mirai.udpplain: This class represents plain UDP assaults that aim to overload IoT devices with UDP traffic, causing service disruption.

Having explained the main features and classes of the two IoT datasets, we next explain our main experimental setup used for generating our evaluation results.

B. EXPERIMENTAL SETUP

The goal is to measure the performance of our anomaly detection models to detect anomalies (and their types) for the two datasets considered in this work (MEMS, and N-BaIoT). We show the performance of our models in terms of the accuracy of detecting the anomaly correctly (measured by precision, recall, and F-1 score). For each proposed model, the training size was 70% of the total collected data while the testing size was 30%. We emphasize that our anomaly detection problem here is a multi-class classification problem in which we try to predict the class for each sample for each IoT dataset.

1) AI MODELS

By pairing ten popular AI classification algorithms (which are decision tree (DT) [20], deep neural network (DNN) [21], random forest (RF) [25], AdaBoost (ADA) [22], support vector machine (SVM) [23], multi-layer perceptron (MLP) [24], bagging [26], blending [27], stacking [28], and voting [29]), we evaluate black-box AI methods and different components of our proposed XAI framework for our two IoT datasets. We emphasize that these ten AI algorithms can be categorized into two categories: (1) single AI models (DT, DNN, ADA, SVM, and MLP), and (2) ensemble models (RF, bagging, blending, stacking, and voting).

2) CODING TOOLS

We built upon the Keras library [63] which is Python-based for creating the variants of our models.

3) EXPLAINABLE AI (XAI) TOOLS

In addition, we employed the following XAI toolkits:

a: SHAP [30]

This toolkit facilitates the generation of feature explanations for our intrusion detection AI models. We utilized SHAP to produce both local and global explanations for our AI models.

b: LIME [36]

Another widely used XAI tool, LIME enables the creation of a surrogate model approximating the original AI model's behavior when assessed with local samples. In this study, we also utilized LIME to generate global explanations by aggregating its local explanations and averaging the importance of each feature.

c: OTHER XAI TOOLKITS

We also used pandas, sklearn (permutation importance), and statistics libraries for the other XAI methods (LOCO, PFI, Profweight, ALE, and CEM).

4) COMPUTING RESOURCES

We conducted anomaly detection experiments using a workstation equipped with an Intel i7 @2.60 GHz processor, 16 GB RAM, and 8 cores. The feature extraction experiments were carried out on the BigRed Workstation at IUPUI, a high-performance computer (HPC) boasting four NVIDIA A100 GPUs, 64 GPU-accelerated nodes (each with 256 GB of memory), and a single 64-core AMD EPYC 7713 processor (2.0 GHz and 225-watt), achieving a peak performance of approximately 7 petaFLOPs. This supercomputer is specifically designed to support researchers in advanced AI and ML tasks [64].

5) PERFORMANCE METRICS

We first do benchmarking of the ten AI models for each of the two datasets (described above in Section IV-A). We show such a comparison of performances of the ten models in terms of the performance metrics (represented by the typical metrics: Precision, Recall, and F-1 Score [65]). We then show the feature importance for main features given different XAI methods considered in this work.

V. MEMS RESULTS EVALUATION

In this section, we start the evaluation of the models' performances using a real dataset sourced from MEMS vibration sensors, our production-grade sensors. Through data analytics applied to the vibration sensor data, we discern one of the motor's three operational modes. The model's performance is presented in terms of defect level detection accuracy, gauged by the classification accuracy of the AI prediction models on the test dataset. This accuracy is defined as the ratio of correctly classified samples to the total number of samples. We also explore various performance metrics, as detailed earlier in Section IV-A.

A. PERFORMANCE METRICS UNDER ALL FEATURES

Table 2 shows the performance metrics for each AI model for our MEMS dataset. We observe that DNN and MLP give the first and second best anomaly detection performances, respectively, (i..e., highest accuracy, precision, and recall). Furthermore, stacking and voting ensemble methods give the third best performance. On the other hand, SVM gives the worst performance for MEMS under all features. This experiment suggests using neural network-based AI models (MLP, or DNN) for classifying the data from the MEMS vibration-based IoT dataset. These neural network-based prediction models perform better than the traditional models due to the fact that the deployments generate enough data for accurate training and due to the complex dependencies among the features of the MEMS dataset.

B. PERFORMANCE METRICS OF FEATURE COMBINATIONS

We next explore the effect of each of the three features (X,Y, and Z) via comparing the performance metrics of different AI models under different combinations of these features where we eliminate one feature for each setup. Thus, we have the following setups: XY, XZ, and YZ, representing different feature combinations for rotation directions. Tables 3-5 show the main results for this experiment.

In Table 3, under the feature combination XY (excluding the most important feature 'z'), the models generally perform poorly. The accuracy, precision, recall, and F1-score are lower across all models compared to Table 2, which incorporates all features including 'z'. This decline suggests that excluding 'z' significantly impacts the models' ability to correctly identify anomalies, underlining the importance of feature z. Table 4 shows results for the feature combination YZ. The inclusion of 'z' leads to a notable improvement in performance metrics compared to Table 3. The Decision Tree, Random Forest, and DNN models all show increased accuracy, precision, recall, and F1-scores. This is consistent with the stated importance of feature 'z', as its presence alongside 'y' provides enough information for models to more accurately predict outcomes. However, they still have lower performance compared to having all features. Finally, in Table 5, the XZ combination is analyzed. Again, the metrics demonstrate better model performance than in Table 3, further supporting the significant role of 'z' in the model's prediction capability. All models experience an increase in accuracy and F1-score when 'z' is present, even when paired with 'x' instead of 'y'.

We summarize below the main findings of such an experiment: **(i) Accuracy:** In terms of accuracy, the combination XZ gives the highest accuracy with the AdaBoost model with a value of 0.655. **(ii) Precision:** In terms of precision, combination XZ gives the highest precision with

TABLE 2. Anomaly detection results (the higher the better) for each AI model. DNN gives the best performance metrics.

Model	Accuracy	Balanced Accuracy	MCC	AUC-ROC	Precision	Recall	F-1 Score
RF	0.68	0.64	0.49	**0.83**	0.67	0.68	0.67
KNN	0.68	0.63	0.48	**0.80**	0.67	0.68	0.66
ADA	0.68	0.60	0.48	**0.79**	0.69	0.68	0.63
SVM	0.67	0.57	0.46	**0.80**	0.52	0.67	0.58
MLP	0.71	0.66	0.53	**0.86**	0.71	0.71	0.70
DNN	0.72	0.65	0.54	0.86	0.71	0.72	0.71
Bagging	0.68	0.64	0.47	0.82	0.67	0.68	0.67
Blending	0.68	0.60	0.38	0.85	0.62	0.62	0.62
Stacking	0.70	0.66	0.51	0.84	0.69	0.70	0.69
Voting	0.70	0.64	0.52	0.85	0.71	0.70	0.68

TABLE 3. Performance metrics under combination XY.

Model	Accuracy	Precision	Recall	F-1 Score
Decision Tree	0.45	0.43	0.45	0.44
Random Forest	0.46	0.44	0.46	0.45
ADA	0.55	0.65	0.55	0.47
SVM	0.49	0.24	0.49	0.32
DNN	0.54	0.41	0.54	0.46

TABLE 4. Performance metrics under combination YZ.

Model	Accuracy	Precision	Recall	F-1 Score
Decision Tree	0.56	0.56	0.56	0.56
Random Forest	0.58	0.57	0.58	0.57
ADA	0.64	0.60	0.64	0.56
SVM	0.65	0.66	0.65	0.59
DNN	0.65	0.65	0.65	0.60

TABLE 5. Performance metrics under combination XZ.

Model	Accuracy	Precision	Recall	F-1 Score
Decision Tree	0.58	0.58	0.58	0.58
Random Forest	0.62	0.61	0.62	0.61
ADA	0.65	0.66	0.65	0.61
SVM	0.62	0.50	0.62	0.54
DNN	0.60	0.60	0.60	0.56

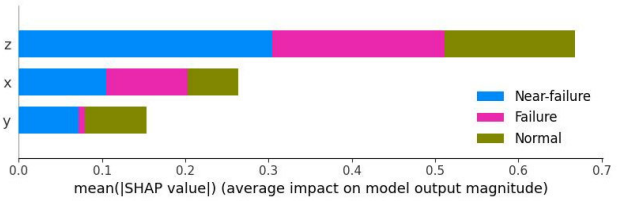


FIGURE 3. SHAP global summary plot example for MEMS using KNN AI model. It shows both top features per anomaly class and overall top feature for the MEMs dataset.

the AdaBoost model with a value of 0.664. **(iii) Recall:** In terms of recall, the combination YZ gives the highest recall with the Deep Neural Network model with a value of 0.648. **(iv) F1 Score:** In terms of F-1 score, the combination XZ gives the highest F1 score with the AdaBoost model with a value of 0.614.

In total, the combination XZ gives the best overall performance across all combination. Across all tables, the consistency of 'z' in contributing to higher performance metrics suggests that it is indeed a critical feature for anomaly detection in the MEMS dataset.

C. OVERALL FEATURE IMPORTANCE (SHAP GLOBAL SUMMARY PLOT)

We next show the overall feature importance under SHAP (SHapley Additive exPlanations) values, which measure each feature's contribution to a machine learning model's prediction. Figure 3 shows such overall feature importance using KNN model. Every horizontal bar is a unique feature, and its length signifies the mean absolute SHAP value, which reflects the average effect of the feature on the output magnitude of the model. Each bar's color—magenta for "Failure," green for "Normal," and blue for "Near-failure"—represents the percentage of each feature's influence on the various prediction classifications. For example, feature 'z' significantly affects the three'classes, but features 'x' and 'y' appear to have varying effects on both the "Normal" and "Failure" predictions. Overall, feature 'z' appears to be most influential feature in predicting all the 3 labels (anomaly states), suggesting it helps in recognizing normal operational states, detecting situations that are close to failure but not yet critical, and also identifying potential failures.

D. FEATURE IMPORTANCE USING SHAP WITH DIFFERENT AI MODELS

Having explained the global summary plot by SHAP, we next provide the overall feature importance by SHAP for the different AI models in our work. The average effect of each feature (x, y, and z) on the model's output magnitude, as determined by the mean absolute SHAP values in Figure 4.

Main Insights: By giving each feature a value that corresponds to its importance for a certain prediction, SHAP values provide a broad overview of feature relevance for different AI models. Figure 4a shows that the most significant feature in the AdaBoost model is z, which is followed by

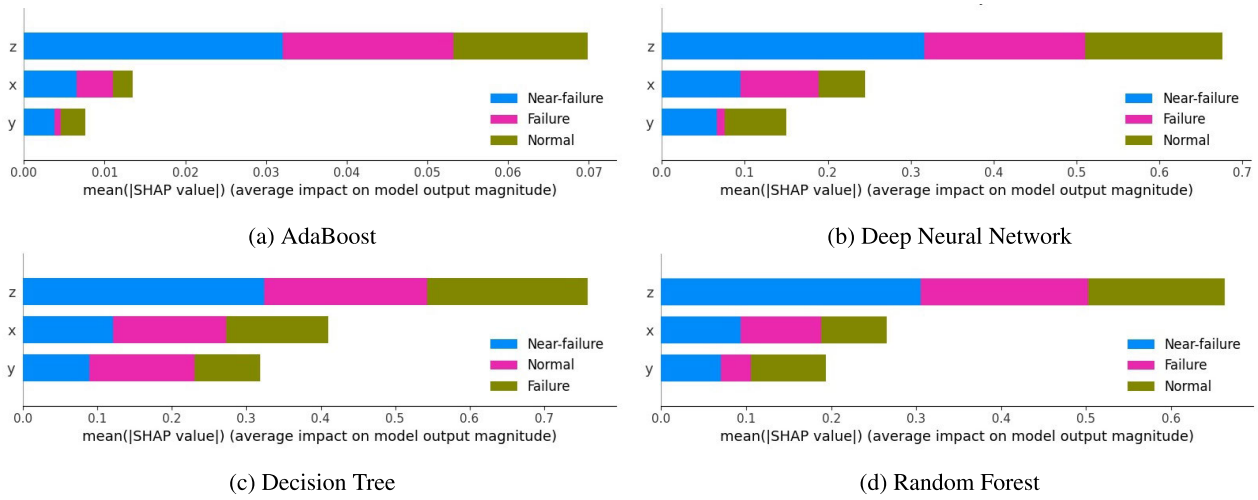


FIGURE 4. Overall feature importance for all three features (x,y, and z) using SHAP for different AI models for MEMS dataset.

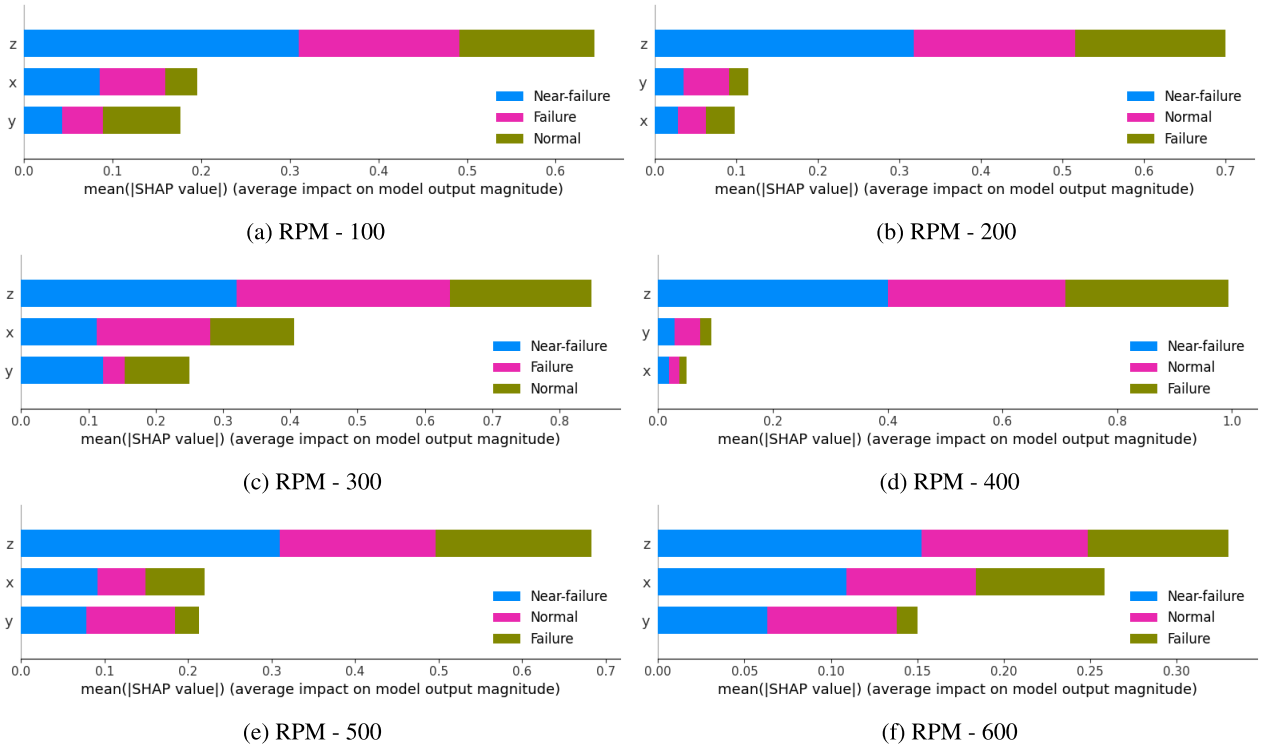


FIGURE 5. Feature importance (using SHAP global summary plots) for different RPMs used in collecting data for MEMS.

x and y. As before, Figure 4b shows that feature z is the most significant feature for Deep Neural Network model, having a significantly larger SHAP value than features x and y. Figure 4c shows that the most significant feature in the Decision Tree model is z, however y has less influence compared to x for this Decision Tree model. In this case, x has nearly as much influence as y. According to the Random Forest model summary in Figure 4d, feature z has also the biggest average influence on model predictions, closely followed by x, and y as the least significant. We have noticed similar insights for SVM and MLP models (omitted here).

E. FEATURE IMPORTANCE FOR EACH RPM

We next show the feature importance for each RPM where we built a separate AI model for each RPM for the six RPM values used for collecting the data for MEMS. The length of

each bar in the charts, which corresponds to a feature (labeled x, y, and z), indicates the mean absolute SHAP value for that feature over a large number of cases. The manner in which the bars are colored—blue for ''Near-failure,'' magenta for ''Failure,'' and green for ''Normal''—indicates how well the feature predicts the various classes or possible outcomes. A factor that has a greater impact on the model's prediction for a given class is shown by a longer bar in that color.

The charts suggest that, for all RPMs, feature 'z' usually has a significant influence on the prediction of ''Normal'' and ''Failure'' classes prediction. Throughout the classes and models/scenarios, the effects of characteristics 'x' and 'y' differ more noticeably. In particular, the feature 'x' ranked the second in four RPMs while feature 'y' has the second rank only in two RPMs. Furthermore, the various charts' varying bar lengths and colors indicate that the importance of each feature to the model's prediction varies based on the model and situation under study. This can provide important insights into the predictive dynamics of the AI model and possibly direct feature selection or model modification. It also shows different behaviors of features within each setting (e.g., feature 'x' affects the decision more in high and low rotational speeds of the motor compared to medium rotational speeds).

F. LOCAL FEATURE IMPORTANCE USING LIME

We next show local feature importance for MEMS dataset. In particular, we provide three examples (one example for each class of the three anomaly classes in MEMS) to show how LIME, which is a well-known local XAI model, that uses features to explain decisions of the AI model on local (single) instance. In these examples (shown in Figure 6), the prediction probabilities for the three classes (designated as 0 (normal), 1 (near-failure), and 2 (failure)) using a RF classifier are displayed with the feature contributions that went into the forecast for each class, with each row representing a distinct case (or example).

The anticipated probabilities for each case are shown for all of the three classes which are displayed in the bar chart on the left. Sets of rules related to features 'x', 'y', and 'z' that influence the model's classification of the instance as ''NOT 0'', ''NOT 1'', or ''NOT 2'' are located next to the bar charts. As shown by the colored boxes and matching values, these rules represent the criteria based on which feature's values that either raise or decrease the likelihood that the instance belongs to a particular anomaly class.

Actual Feature Values and LIME Rules: The actual feature values for each instance are displayed in the rightmost column. For instance, with a chance of 0.92, the instance in the top row is largely anticipated to be normal (class 0). The properties '$y > 0.10$' and '$z \leq 0.12$' are crucial requirements that support this classification, according to the standards. In a similar vein, the bottom and middle rows present the influential conditions and forecasts for additional cases that are primarily categorized as classes 1 (near failure) and 2 (failure), respectively in this example.

Understanding the AI model's reasoning behind certain predictions of the IoT instance is made easier with the help of this LIME explanation. This is particularly helpful for high-stakes applications like IoT or smart manufacturing, where decision-making must be transparent and comprehensible to ensure safety in these systems.

G. FEATURE IMPORTANCE USING LEAVE ONE COVARIATE OUT (LOCO)

We next show the feature importance with an interpretability technique called LOCO [31]. The main idea of this method is to evaluate the effect of each feature individually on the prediction. It entails retraining the AI model several times, omitting a distinct characteristic each time, and tracking how the predictions of the AI model alter. We used LOCO while using RF AI model for MEMS dataset and it shows that the prediction difference for feature 'x' is 1 which means that when we set the feature 'x' to its mean value while keeping other features unchanged, the RF model's prediction increased by 1, while the prediction difference for features y and z are 0. This means that when we set the feature y (or z) to its mean value while keeping other features unchanged, the RF model's prediction remained the same (no change).

We can also show the feature importance bar chart using LOCO for each feature. We show an example for MEMS dataset using MLP model in Figure 7. In this figure, three features (x, y, and z) are displayed in a bar chart that illustrates their relative importance according to the LOCO approach. The change in accuracy of predictions or some other performance parameter when each feature is removed is used to determine how important a feature is. With the highest relevance, feature z signifies that the MLP model's predictions are significantly affected by its value. On the other hand, the least important characteristic is feature y, while feature x is of intermediate value.

H. FEATURE IMPORTANCE USING CONTRASTIVE EXPLANATIONS METHOD (CEM)

We next explore the feature importance for MEMS datset's features using CEM method [32]. CEM is a method for deciphering machine learning models by examining how the model's predictions alter in response to perturbations in the input features. This technique looks at how changing a feature might affect the forecast in order to assist determine which features have the biggest influence on the model's output. The unique property of CEM is that it sheds light on why an AI model made a specific decision by contrasting it with alternative decisions.

We show two examples for using CEM for the MEMS dataset in Figures 8-9. The figures are bar charts that show the feature importance for two different models as evaluated by the Counterfactual Explanation Method (CEM). In these figures, the three features (x, y, z) of the MEMS dataset are displayed together with their corresponding importance scores. For the RF model, CEM shows that feature z is the most significant feature in contrast to LOCO that shows that

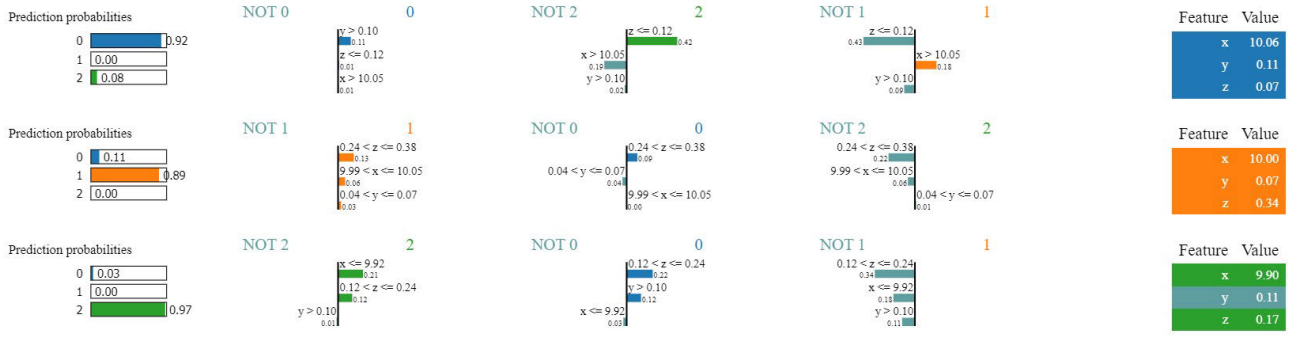


FIGURE 6. Local feature importance using LIME XAI method for three data instances from the MEMS dataset.

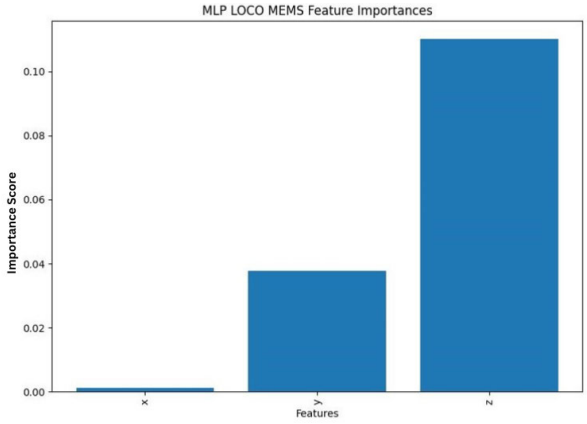


FIGURE 7. Feature importance using LOCO for the main features (x,y, and z) of MEMS dataset using MLP AI model.

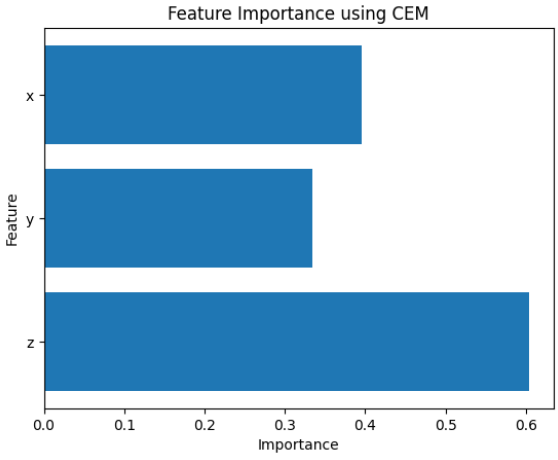


FIGURE 8. Feature importance using CEM for the main features (x,y, and z) of MEMS dataset using RF AI model.

x is the top one. For CEM, x is the second top feature for RF model. For the MLP model, both LOCO and CEM lead to same finding that feature z is the top feature.

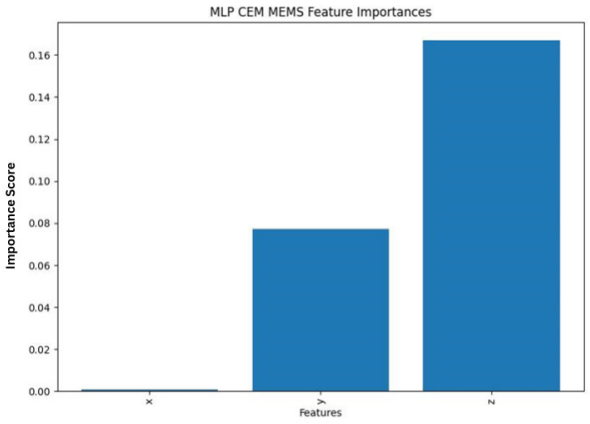


FIGURE 9. Feature importance using CEM for the main features (x,y, and z) of MEMS dataset using MLP AI model.

Main Insights: Since feature z is constantly the most significant in both AI models using CEM, it is likely a major factor in the decisions made by the anomaly detection model for the MEMS dataset. Although still significant, characteristic x is not as significant as feature z. Finally, feature y has almost no influence for MLP model in contrast to RF model using CEM XAI method.

I. FEATURE IMPORTANCE USING ACCUMULATED LOCAL EFFECT (ALE)

We next show another local feature importance illustration using accumulated local effect (ALE) method. In this experiment, our three distinct features (x, y, and z) are represented by Accumulated Local Effects (ALE) plots in Figures 10-12. The machine learning model (RF model in this experiment) predicts three classes: ''normal,'' ''near-failure,'' and ''failure.'' ALE plots are used to illustrate how characteristics, taking into consideration their interactions with other features, impact a machine learning model's average prediction. They provide insight into the connection between the input features and the anticipated prediction (result). In contrast to LIME, it shows the relationship

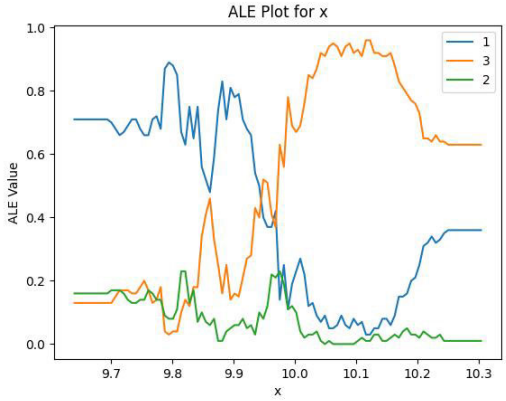


FIGURE 10. ALE plot for MEMS dataset (Feature x) for all anomaly classes (1: normal, 2: near-failure, and 3: failure).

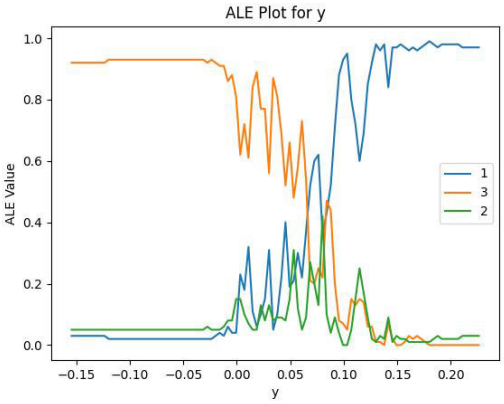


FIGURE 11. ALE plot for MEMS dataset (Feature y) for all anomaly classes (1: normal, 2: near-failure, and 3: failure).

between each feature individually (in each single ALE plot) and different anomaly classes.

With the ALE value on the y-axis plotted against the feature value on the x-axis, each figure represents a single feature (Figure 10 for x, Figure 11 for y, and Figure 12 for z). The influence of the feature on each of the three classes is represented by the lines in various colors. A steep slope or peak in the line, for example, suggests that the feature value has a significant impact on the model's prediction for that class. On the other hand, flat sections imply that the prediction remains constant when the feature value varies, which means that the feature is less important. The ALE plots (Figures 10-12) show that feature x is most important for predicting 'Failure' class, feature y is most important for predicting 'Normal' class, and feature z is most important for predicting 'Near-failure'.

Main Insights: For feature x, after a certain value of x, the effect on the prediction for class 1 ('normal') increases noticeably, whereas for class 3 ('failure'), there is a noticeable drop. Class 2 ('near-failure') has an impact that varies but is less noticeable. Regarding feature y, all classes exhibit dramatic swings, suggesting that even minor y changes

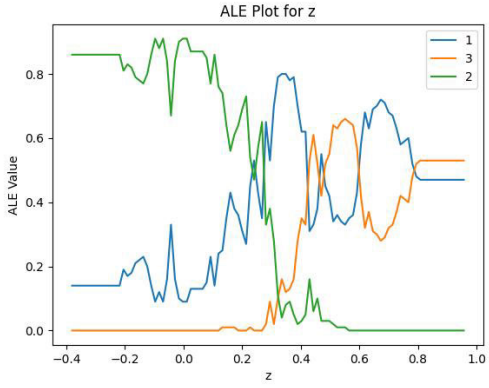


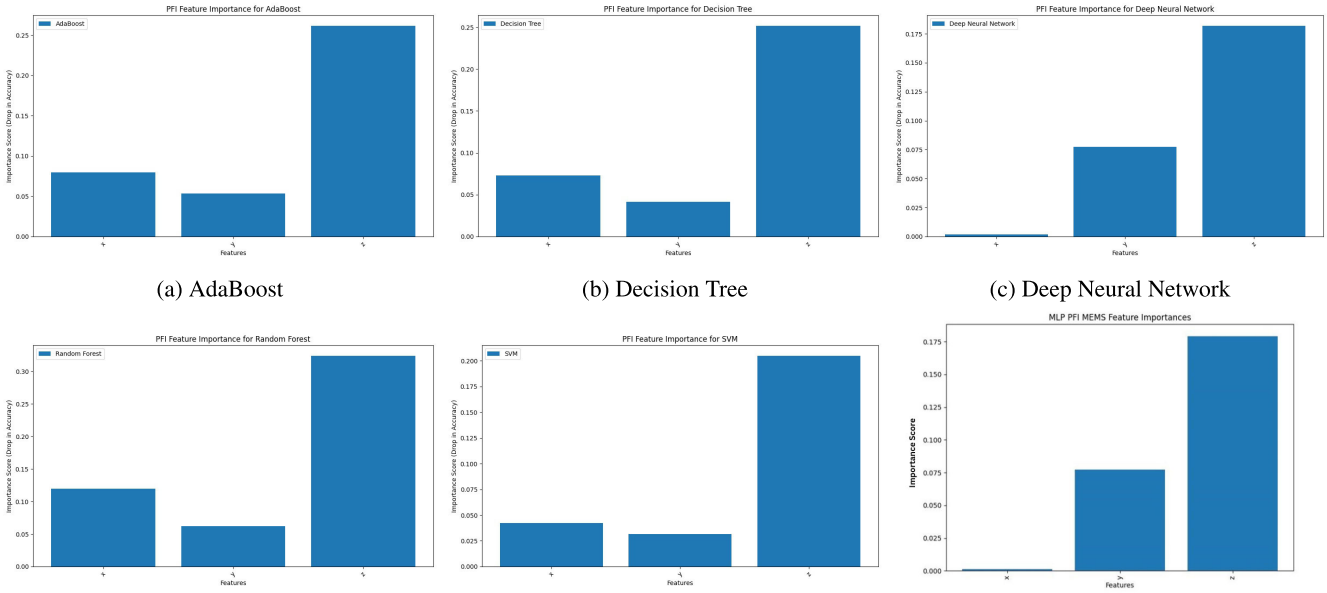
FIGURE 12. ALE plot for MEMS dataset (Feature z) for all anomaly classes (1: normal, 2: near-failure, and 3: failure).

can have a big influence on the prediction. This implies a complex and non-linear relationship between the variable and the expected prediction. With regard to feature z, the plot illustrates notable peaks and troughs for all classes, suggesting intricate relationships where specific z ranges have an enormous effect on each class's prediction.

J. FEATURE IMPORTANCE USING PERMUTATION FEATURE IMPORTANCE (PFI) WITH DIFFERENT AI MODELS

The Permutation Feature Importance (PFI) approach was used in our evaluation experiments to determine the feature importance for several AI models. By randomly shuffling the feature values and so disrupting the correlation between the feature and the target, this method evaluates the significance of each feature by monitoring the shift in the AI model's performance, typically accuracy. We show the feature importance for all of our three features (x, y, and z) for all of the six AI models considered in our work (i.e., AdaBoost, Deep Neural Network, Multi Layer Perceptron, Random Forest, Support Vector Machine, and Decision Tree). The feature importance of these different models are compared in the bar charts in Figure 13. The x-axis represents the features (x, y, z), and the y-axis quantifies the feature importance score, inferred as the drop in model accuracy when each feature's values are permuted.

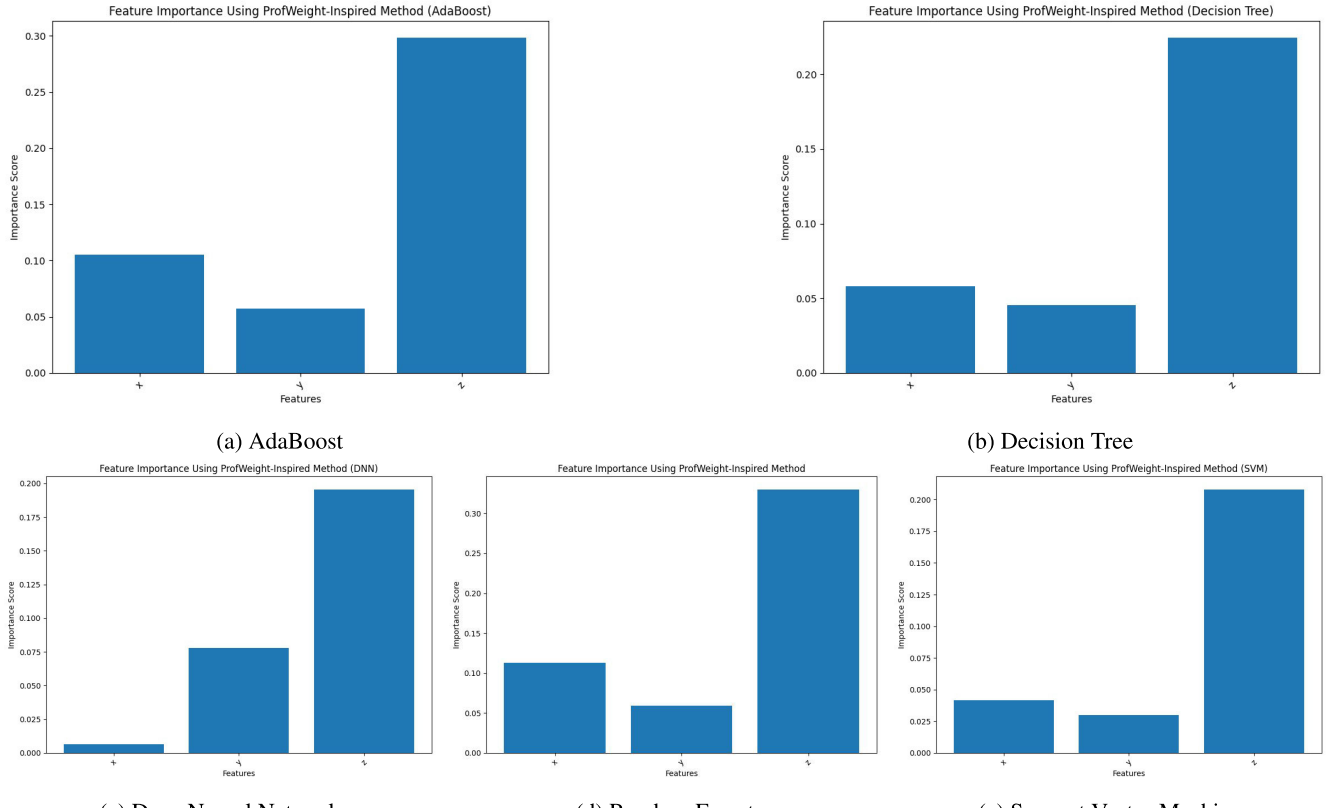
Main Insights: Figure 13 shows that feature z is likely a crucial predictor, as indicated by the consistent pattern in which feature z is the most important feature under PFI throughout the six AI models for MEMS dataset. Thus, the model building and evaluation process should give close consideration to such a feature. For the other two features, feature x was the second top feature in four of the six AI models (which are ADA, DT, RF, and SVM), while feature y was the second top feature in only two AI models (DNN, and MLP). Given that feature y is the least significant, the main explanation for such behaviour for feature y is that the Y-axis is representing the direction of the shaft which exhibits smaller vibrations compared to Z-axis and X-axis.



(a) AdaBoost (b) Decision Tree (c) Deep Neural Network

(d) Random Forest (e) Support Vector Machine (f) Multi Layer Perceptron

FIGURE 13. The feature importance for three main features (x,y, and z) in MEMS dataset using Permutation Feature Importance (PFI) for all AI models (AdaBoost, Decision Tree, Deep Neural Network, Random Forest, Support Vector Machine, and MLP).



(a) AdaBoost (b) Decision Tree

(c) Deep Neural Network (d) Random Forest (e) Support Vector Machine

FIGURE 14. The feature importance for three main features (x,y, and z) in MEMS dataset using Profweight for different AI models for anomaly detection (AdaBoost, Decision Tree, Deep Neural Network, Random Forest, and Support Vector Machine).

TABLE 6. The feature importance (given by top three features) for each anomaly type for the MEMS dataset. The feature *z* is the top feature across all different attack types.

Feature Rank	Normal	Near-failure	Failure
1	z	z	z
2	x	y	x
3	y	x	y

K. FEATURE IMPORTANCE USING PROFWEIGHT WITH AI MODELS

We next explore the ProfWeight-inspired method to evaluate and display feature relevance for MEMS dataset. The ProfWeight approach assesses the significance of a feature by considering many criteria, such as the feature's weight within the model and its relationship with other features. This approach appears to offer an indicator of the relative contribution of each attribute to the model's predictions.

We evaluated Profweight for a variety of machine learning models (considered in our work), including AdaBoost, Decision Tree, Random Forest, Deep Neural Network, and Support Vector Machine. Figure 14 shows the Profweight feature importance for MEMS dataset. Again, three features are displayed on x-axis of each graph, with the y-axis representing each feature's relevance ranking.

Main Insights: Similar to PFI, Figure 14 shows that the feature 'z' clearly outweighs the others in each graph in terms of relevance (importance) score across all models for Profweight XAI method. The relevance scores of the other two variables are noticeably lower, indicating that they have less of an effect on the model's functionality. Regardless of the model type, this constant ranking across models indicates that z is the most significant feature and that it has a strong and consistent influence on the model's predictions. These insights can be critical for activities like feature selection, model interpretation, and enhancing model performance by concentrating on the most informative properties. They can also be useful for determining which aspects influence anomaly detection model decisions which can also help in preventing adversarial attacks on such AI methods.

L. FEATURE IMPORTANCE FOR EACH ANOMALY TYPE

We next show the anomaly-specific feature importance (given by the top three features based on each anomaly type). This is calculated using averaging SHAP feature importance for different AI models. Table 6 shows such a list for MEMS dataset. We observe that for different anomaly types, feature 'z' is the top feature. For 'near-failure' anomaly class, feature 'y' has the second rank. For 'near-failure' anomaly class, feature 'y' has the second rank. For 'failure' anomaly class, feature 'x' has the second rank. This anomaly-specific feature importance can help in tuning AI models for detecting different conditions of the IoT device (here, the condition of the motor) given such feature importance knowledge.

M. SUMMARY OF FEATURE ANALYSIS FOR MEMS

We have provided overall feature importance analysis using six XAI methods (SHAP, LOCO, CEM, PFI, and ProfWeight). We have also provided local feature importance analysis using the popular LIME method and ALE method. Furthermore, we provided feature importance for different RPMs considered in collecting MEMS data for more in-depth understanding. All things considered, feature z (motor acceleration in Z direction) seems to have the most impact on the predictions across all models from SHAP feature importance results. Despite differing by model in terms of its significance in relation to z, feature x also exhibits a significant impact.

Generally speaking, feature y has the least effect. As illustrated in the experimental setup (Figure 2a), the motor's rotation, coupled with the imbalanced disk due to the mounted mass (eccentric weight), leads to unbalanced centripetal forces, causing repeated vibrations in multiple directions. Given the circular movement around the motor's center, the primary vibrations occur along the X and Z axes, while the Y-axis (aligned with the shaft) shows relatively minor vibrations. These variations in data patterns might not be distinctly discernible with changes in machine health. Such feature selection necessitates domain knowledge, specifically an understanding of how the motor vibrates and the relative sensor placement. The rationale is that the selection of discriminative features, informed by this domain knowledge, can significantly influence the AI model's learning process.

The link between features and predictions appears to be extremely model-dependent, based on the diversity in feature impact across different models. Thus, we have used several XAI methods for confirming which features appear frequently as top important features for different AI models. By highlighting the features that are most useful for making predictions, our analysis can help direct feature engineering and model selection.

Having provided different results for MEMS dataset, we next provide the main evaluation results for N-BaIoT dataset.

VI. N-BaIoT RESULTS EVALUATION

A. PERFORMANCE METRICS WITH ALL 115 FEATURES

We first show the main performance metrics (accuracy, precision, recall, and F-1 score) for the different 10 AI models (both single AI models and ensemble methods). Table 7 shows such results. Using aforementioned variety of machine learning models, the table gives a thorough summary of the anomaly detection outcomes for the 9 IoT devices that were discussed in Section IV-A. The performance measures (accuracy, precision, recall, and F-1 score) are displayed in the columns, with each row representing a distinct AI model. Notably, the different ensemble methods (Random Forest, Bagging, Blending, Stacking, and Voting) show consistently high scores for all measures, with recall,

TABLE 7. N-BaIoT - anomaly detection results for the nine devices (the higher the better) for each model.

	DEVICE 1 (Danmini Doorbell)				DEVICE 2 (Ecobee Thermostat)			
Model	**Accuracy**	**Precision**	**Recall**	**F-1 Score**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Decision Tree	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Random Forest	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
ADA	0.94	0.96	0.94	0.94	0.82	0.80	0.82	0.79
SVM	0.86	0.82	0.86	0.83	0.86	0.81	0.86	0.83
MLP	0.91	0.86	0.91	0.88	0.91	0.95	0.91	0.88
DNN	0.90	0.95	0.90	0.87	0.91	0.95	0.90	0.87
Bagging	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Blending	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Stacking	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Voting	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	DEVICE 3 (Ennio Doorbell)				**DEVICE 4 (Philips Baby Monitor)**			
Model	**Accuracy**	**Precision**	**Recall**	**F-1 Score**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Decision Tree	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Random Forest	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
ADA	0.91	0.94	0.91	0.90	0.75	0.75	0.75	0.90
SVM	0.79	0.88	0.79	0.73	0.86	0.82	0.86	0.83
MLP	0.82	0.86	0.82	0.76	0.91	0.87	0.91	0.88
DNN	0.83	0.91	0.83	0.78	0.91	0.95	0.91	0.88
Bagging	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Blending	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Stacking	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Voting	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	DEVICE 5 (Provision PT 737E Security Camera)				**DEVICE 6 (Provision PT 838 Security Camera)**			
Model	**Accuracy**	**Precision**	**Recall**	**F-1 Score**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Decision Tree	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Random Forest	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
ADA	0.81	0.81	0.81	0.77	0.83	0.81	0.83	0.78
SVM	0.85	0.81	0.85	0.82	0.86	0.82	0.86	0.83
MLP	0.90	0.92	0.91	0.88	0.91	0.95	0.91	0.88
DNN	0.90	0.89	0.90	0.87	0.90	0.95	0.90	0.87
Bagging	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Blending	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Stacking	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Voting	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	DEVICE 7 (Samsung SNH 1011 N Webcam)				**DEVICE 8 (SimpleHome 1002 Security Camera)**			
Model	**Accuracy**	**Precision**	**Recall**	**F-1 Score**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Decision Tree	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Random Forest	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
ADA	0.83	0.80	0.83	0.77	0.80	0.77	0.80	0.77
SVM	0.78	0.70	0.78	0.73	0.86	0.82	0.86	0.83
MLP	0.83	0.92	0.83	0.78	0.91	0.92	0.91	0.88
DNN	0.83	0.91	0.83	0.77	0.90	0.86	0.90	0.87
Bagging	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Blending	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Stacking	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Voting	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	DEVICE 9 (SimpleHome 1003 Security Camera)							
Model	**Accuracy**	**Precision**	**Recall**	**F-1 Score**				
Decision Tree	0.99	0.99	0.99	0.99				
Random Forest	0.99	0.99	0.99	0.99				
ADA	0.80	0.77	0.80	0.76				
SVM	0.85	0.80	0.85	0.82				
MLP	0.91	0.95	0.91	0.88				
DNN	0.90	0.86	0.90	0.87				
Bagging	0.99	0.99	0.99	0.99				
Blending	0.99	0.99	0.99	0.99				
Stacking	0.99	0.99	0.99	0.99				
Voting	0.99	0.99	0.99	0.99				

F-1 Score, and accuracy and precision all approaching 0.99. These algorithms demonstrate remarkable efficacy in precisely detecting data anomalies. From single AI models, Decision Tree, provides the best results with performance similar to ensemble methods. The MLP and DNN models show also promising performances for seven of the nine devices. On the other hand, the ADA model has the worst performance scores, suggesting that it would be less robust for anomaly detection for this dataset. Finally, the performance of the SVM model varies for different devicesm but generally has much lower performance compared to DT, and MLP.

Main Insights: The main insight from such performance evaluation using all features for all nine devices of N-BaIoT is that ensemble methods work best for predicting different anomaly classes since combining different AI methods in these ensemble methods harness the strengths of diverse models and mitigate the weaknesses of lower performed ones.

We next provide feature importance analysis for one of the nine devices (Samsung webcam IoT device which is ''device 7''). We chose this device since it has the best average performance across different devices, thus the generated feature analysis would be more trustworthy compared to other devices. We also emphasize that we also generated the feature importance for other 8 IoT devices (not shown here in the interest of space). Again, we perform different feature importance analysis using our different XAI methods (LOCO, CEM, SHAP, PFI, Profweight, ALE, and LIME), as detailed next.

B. OVERALL FEATURE IMPORTANCE (SHAP GLOBAL SUMMARY PLOT)

We next show the overall feature importance under SHAP (SHapley Additive exPlanations) values, which measure each feature's contribution to a machine learning model's prediction. Figure 15b shows such overall feature importance using Random Forest model. Every horizontal bar is a unique feature, and its length signifies the mean absolute SHAP value, which reflects the average effect of the feature on the output magnitude of the model. Each bar's color—red for ''benign,'' green for ''gafgyt.udp,'' blue for ''gagfyt.scan,'' purple for ''gagfyt.junk,'' orange for ''gagfyt.combo,'' and yellow for gagfyt.tcp —represents the percentage of each feature's influence on the various prediction classifications for different botnet attack classes. For example, feature 'HH_L1_magnitude' significantly affects the ''benign'', ''gagfyt.tcp'', and ''gagfyt.combo'' classes, but features 'HH_L1_weight' have more effect for ''gagfyt.junk'' and ''gagfyt.udp' attack classes. One other class of features have moderate effects for specific attack classes (e.g., 'HH_L5_mean' for ''gagfyt.scan'' and 'HH_L5_weight' for ''gagfyt.udp''). On the other hand, some features have almost no effect for most classes under ranodm forest model (e.g., 'HH_L5_pcc', and 'HH_L5_covariance').

C. FEATURE IMPORTANCE USING SHAP WITH DIFFERENT MODELS

We next provide the overall feature importance by SHAP for the different AI models for N-BaIoT dataset. The average effect of top-20 features on the model's output magnitude determined by the mean absolute SHAP value is displayed in Figure 15.

Main Insights: By giving each feature a value that corresponds to its importance for a certain prediction, SHAP values provide a broad overview of feature relevance for different AI models. Figure 15a shows that the most significant feature in the AdaBoost model is ''HH_L1_weight'', which is followed by ''HH_L1_magnitude'' and ''HH_L5_weight''. Figure 15b shows that feature ''HH_L1_magnitude' is the most significant feature for Random Forest model, which is followed by ''HH_L3_weight'' and ''HH_L3_magnitude''. The feature importance pattern of Decision Tree has similar pattern to that of Random Forest. According to the MLP model summary in Figure 15c, feature ''HH_L1_magnitude'' has also the biggest average influence on model predictions, closely followed by ''HH_L5_magnitude'', and ''HH_L5_mean'' has the third rank. We have noticed similar insights for SVM and DNN models (omitted in interest of space).

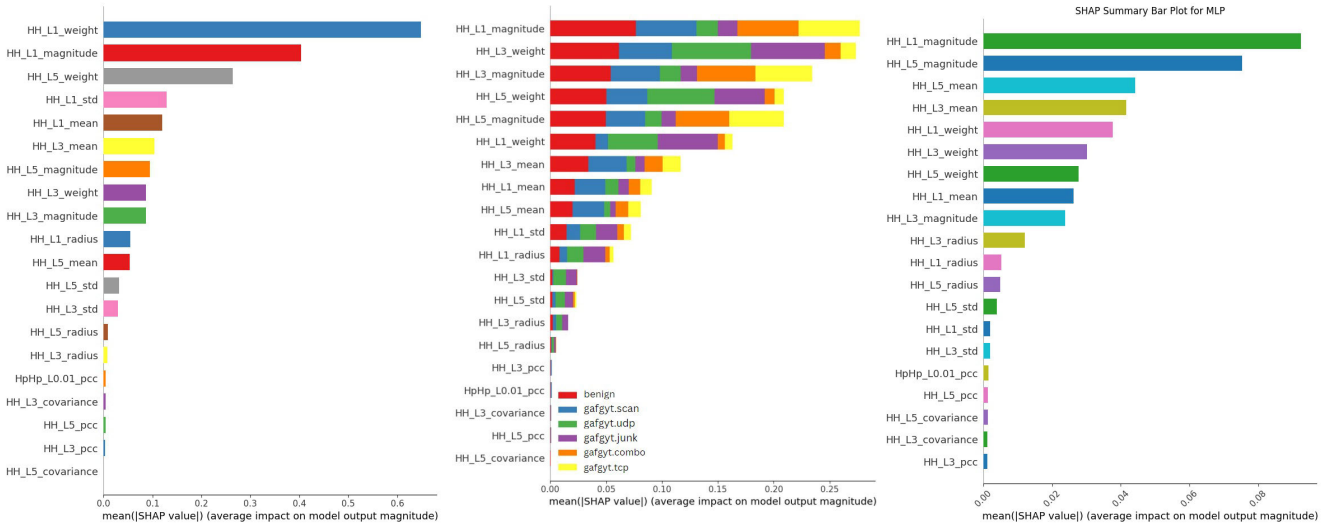
D. FEATURE IMPORTANCE WITH CONTRASTIVE EXPLANATIONS METHOD (CEM)

Recall that contrastive explanations method (CEM) shows what may be altered in the input features to get a different prediction output, which sheds light on how a machine learning model makes decisions. They give complicated models like MLPs some interpretability. Figure 16 shows the feature importance using CEM for the top features of N-BaIoT dataset using different AI models. Each bar represents a different feature as explained in Section IV-A. The y-axis, which has values between 0 and 0.4, measures the relevance. When it comes to the model's performance or predictions, the features with the tallest bars are the most significant. For CEM, the top important features are ''HH-L1-weight'' with Adaboost and Random Forest and ''HH-L1-magnitude'' with Decision Tree and MLP models.

E. FEATURE IMPORTANCE WITH LOCO

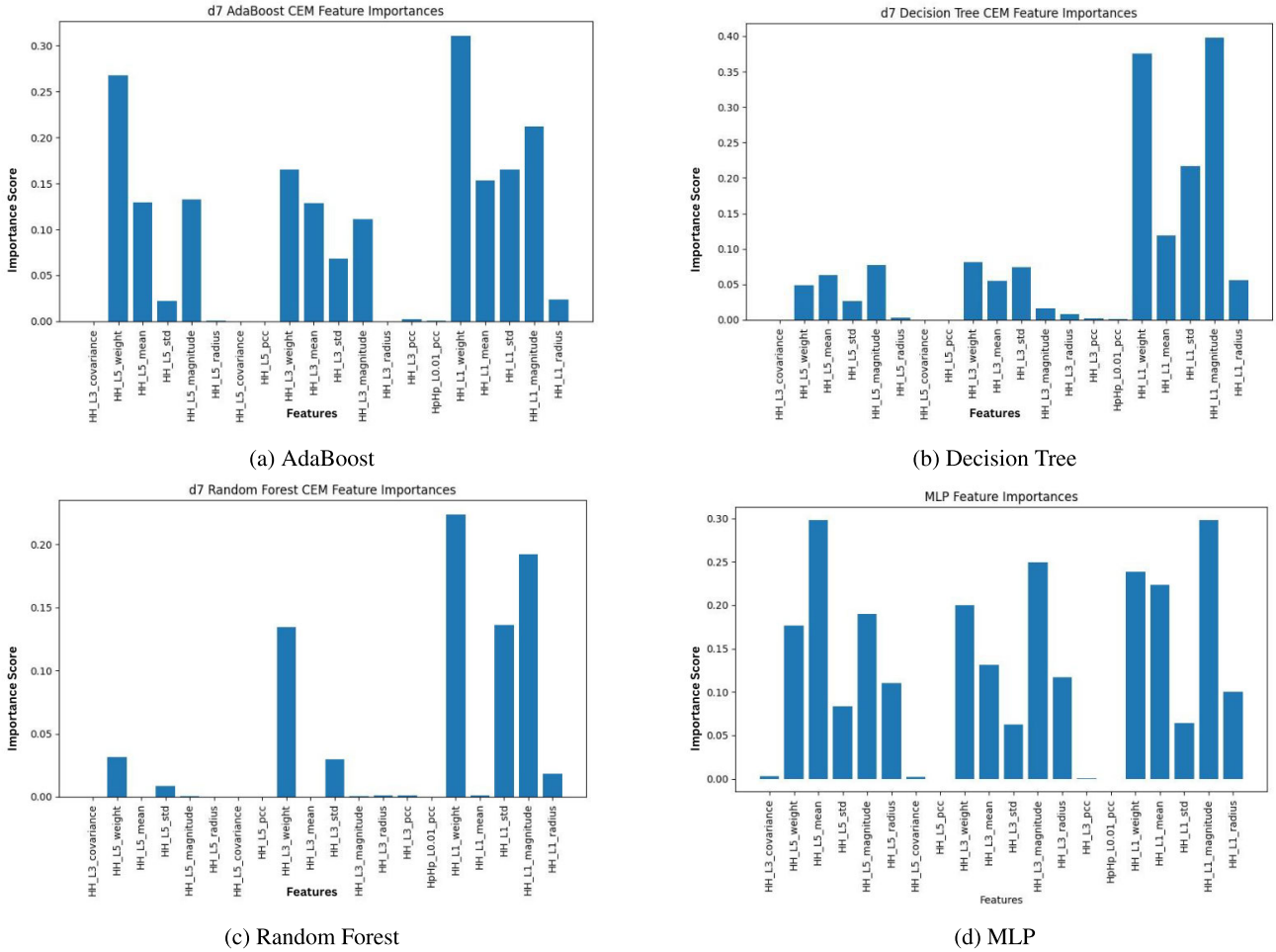
Recall that Leave-One-Covariate-Out (LOCO) is a strategy used in machine learning models to estimate the importance of features. When a feature is removed from the model, the LOCO technique measures the change in prediction error to assess how important the feature is. To be more precise, the model is retrained several times, omitting a distinct feature each time, and the effect on model's performance is tracked.

Figure 17 shows the main outcomes of using the LOCO approach on different AI models such as AdaBoost, Decision Tree, Random Forest, and MLP. The x-axis lists the features by name, and the y-axis, which is scaled on logarithmic



(a) AdaBoost (b) Random Forest (c) MLP

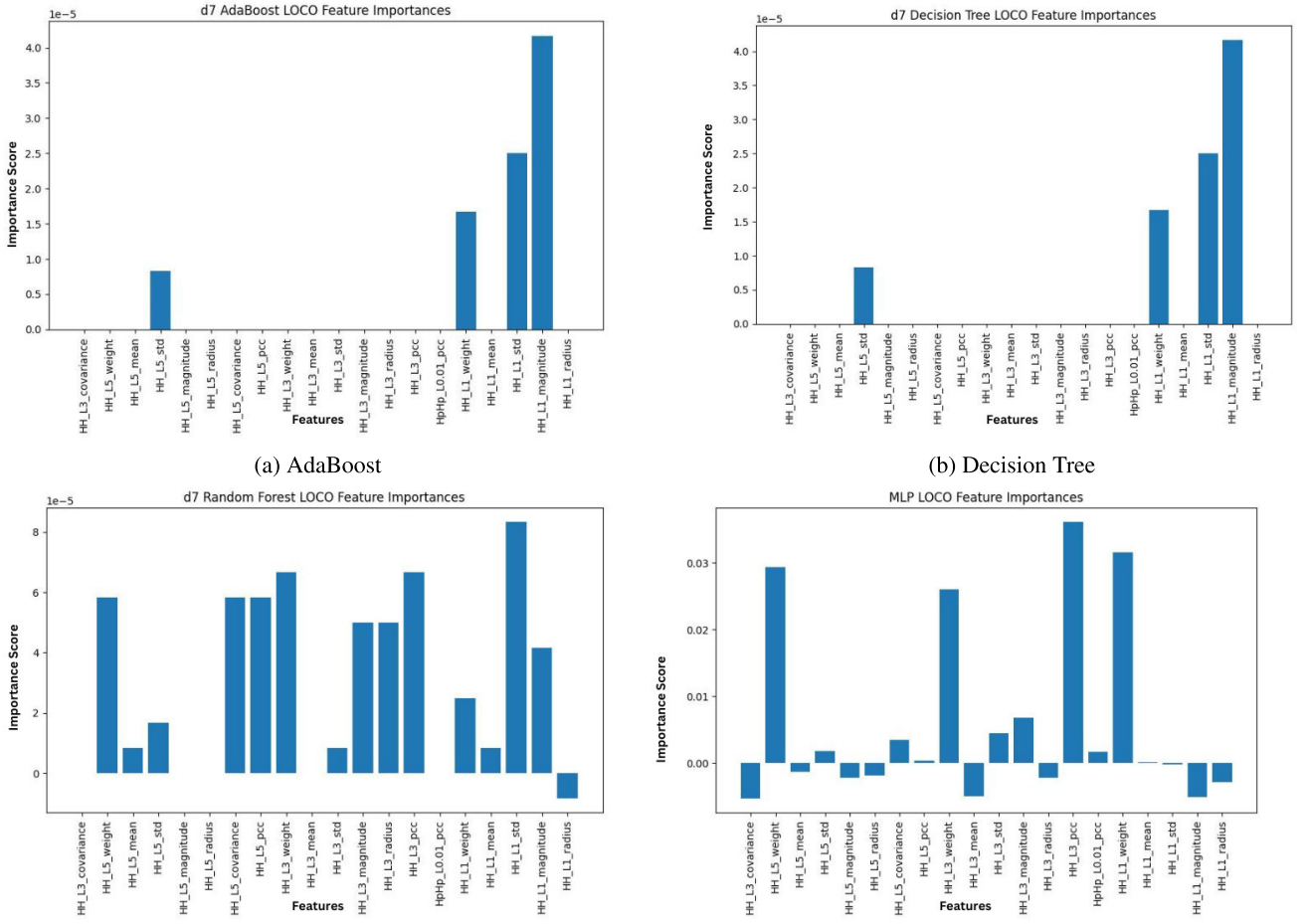
FIGURE 15. Overall feature importance for all top-20 features using SHAP for different AI models for N-BaIoT dataset.



(a) AdaBoost (b) Decision Tree

(c) Random Forest (d) MLP

FIGURE 16. Feature importance using CEM XAI method for the top features of N-BaIoT dataset using different AI models.



(a) AdaBoost

(b) Decision Tree

(c) Random Forest

(d) MLP

FIGURE 17. Feature importance using LOCO XAI method for the top features of N-BaIoT dataset using different AI models.

scale (notice the ''1e-5'' that shows the scale factor), shows the feature importance scores. Every bar illustrates how the prediction performance of the model is affected by eliminating a single feature. Higher bars denote features that are crucial for producing accurate predictions of different anomaly classes because their absence dramatically reduces the model's accuracy. From the figures we can infer that feature ''HH-L1-magnitude'' is most important with the Adaboost and Decision Tree Ai models, followed by ''HH-L1-std'', while feature ''HH-L1-std'' ranks most important with the RF model followed by ''HH-L3-pcc'' which in-turn ranks the highest with the MLP model.

F. FEATURE IMPORTANCE WITH PERMUTATION FEATURE IMPORTANCE (PFI)

Recall that the Permutation Feature Importance (PFI) method is another technique used in machine learning models to assess the importance of features. In PFI, a feature's importance is determined by permuting its values and tracking how the model's performance indicators (accuracy in our experiments) change as a result. More specifically, the values of a given feature are randomly permuted or shuffled between instances once the model is trained on the original dataset. Each feature goes through this procedure several times, and the average performance change indicates how important the feature is. Permuting the values of a feature that is critical to the model's predictions would probably result in a notable decline in performance. Thus, the PFI technique provides information about how each feature affects the overall performance of the model. This motivates our usage of PFI for feature importance for N-BaIoT dataset.

Figure 18 shows the outcomes of different models under Permutation Feature Importance (PFI) for N-BaIoT dataset where one feature at a time is changed and all other features remain unaltered, and this is done for all features. The significance scores are displayed on the y-axis; higher values signal that the feature is more important because permuting it has a bigger detrimental effect on the model's performance. The features are listed on the x-axis and reflect various statistical attributes of different AI models. Comparing the figures of all the models, we can observe that features

TABLE 8. N-BaIoT - anomaly detection results for the nine devices only using the top-20 features.

	DEVICE 1 (Danmini Doorbell)				DEVICE 2 (Ecobee Thermostat)			
Model	**Accuracy**	**Precision**	**Recall**	**F-1 Score**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Decision Tree	0.68	0.85	0.68	0.67	0.69	0.89	0.69	0.68
Random Forest	0.68	0.83	0.68	0.87	0.70	0.82	0.70	0.68
ADA	0.59	0.64	0.59	0.55	0.56	0.47	0.56	0.48
SVM	0.53	0.58	0.53	0.48	0.53	0.37	0.38	0.32
DNN	0.63	0.78	0.63	0.62	0.64	0.68	0.64	0.62
Bagging	0.68	0.83	0.68	0.87	0.70	0.90	0.70	0.69
Blending	0.68	0.83	0.68	0.87	0.70	0.90	0.70	0.69
Stacking	0.68	0.83	0.68	0.87	0.70	0.90	0.70	0.69
Voting	0.68	0.83	0.68	0.87	0.70	0.90	0.70	0.69
	DEVICE 3 (Ennio Doorbell)				**DEVICE 4 (Philips Baby Monitor)**			
Model	**Accuracy**	**Precision**	**Recall**	**F-1 Score**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Decision Tree	0.82	0.73	0.82	0.76	0.68	0.80	0.68	0.68
Random Forest	0.82	0.74	0.82	0.76	0.69	0.78	0.69	0.68
ADA	0.66	0.56	0.66	0.56	0.58	0.65	0.58	0.55
SVM	0.43	0.40	0.43	0.34	0.49	0.44	0.49	0.43
DNN	0.73	0.64	0.73	0.67	0.63	0.71	0.63	0.63
Bagging	0.82	0.74	0.82	0.77	0.69	0.79	0.69	0.68
Blending	0.82	0.74	0.82	0.77	0.69	0.79	0.69	0.68
Stacking	0.82	0.74	0.82	0.77	0.69	0.79	0.69	0.68
Voting	0.82	0.74	0.82	0.77	0.69	0.79	0.69	0.68
	DEVICE 5 (Provision PT 737E Security Camera)				**DEVICE 6 (Provision PT 838 Security Camera)**			
Model	**Accuracy**	**Precision**	**Recall**	**F-1 Score**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Decision Tree	0.64	0.77	0.64	0.63	0.64	0.79	0.64	0.63
Random Forest	0.64	0.78	0.64	0.64	0.64	0.84	0.64	0.64
ADA	0.52	0.65	0.52	0.48	0.53	0.63	0.53	0.50
SVM	0.26	0.39	0.26	0.18	0.26	0.27	0.26	0.18
DNN	0.58	0.71	0.58	0.58	0.57	0.70	0.57	0.57
Bagging	0.64	0.77	0.64	0.64	0.64	0.84	0.64	0.64
Blending	0.64	0.77	0.64	0.64	0.64	0.84	0.64	0.64
Stacking	0.64	0.86	0.64	0.64	0.64	0.84	0.64	0.64
Voting	0.64	0.77	0.64	0.64	0.64	0.84	0.64	0.64
	DEVICE 7 (Samsung SNH 1011 N Webcam)				**DEVICE 8 (SimpleHome 1002 Security Camera)**			
Model	**Accuracy**	**Precision**	**Recall**	**F-1 Score**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Decision Tree	0.82	0.74	0.82	0.76	0.71	0.80	0.71	0.69
Random Forest	0.83	0.74	0.83	0.77	0.71	0.80	0.71	0.70
ADA	0.66	0.49	0.66	0.54	0.56	0.50	0.56	0.47
SVM	0.46	0.51	0.46	0.33	0.53	0.54	0.53	0.48
DNN	0.74	0.66	0.74	0.68	0.65	0.71	0.65	0.73
Bagging	0.83	0.74	0.83	0.77	0.71	0.81	0.71	0.70
Blending	0.83	0.74	0.83	0.77	0.71	0.81	0.71	0.70
Stacking	0.83	0.74	0.83	0.77	0.71	0.81	0.71	0.70
Voting	0.83	0.74	0.83	0.77	0.71	0.81	0.71	0.70
	DEVICE 9 (SimpleHome 1003 Security Camera)							
Model	**Accuracy**	**Precision**	**Recall**	**F-1 Score**				
Decision Tree	0.70	0.78	0.70	0.69				
Random Forest	0.71	0.80	0.71	0.70				
ADA	0.55	0.46	0.55	0.47				
SVM	0.41	0.51	0.41	0.36				
DNN	0.65	0.75	0.65	0.74				
Bagging	0.71	0.79	0.71	0.70				
Blending	0.71	0.79	0.71	0.70				
Stacking	0.71	0.79	0.71	0.70				
Voting	0.71	0.79	0.71	0.70				

"HH-L1-weight" and "HH-L1-magnitude" visibly rank as the top two important features in all models except with Adaboost, where "HH-L1-std" is the second highest ranked feature, making "HH-L1-magnitude" the third one.

G. FEATURE IMPORTANCE USING PROFWEIGHT

ProfWeight, which stands for Profiled Weighting, is a technique designed to explain how various features affect an AI model's predictions. It entails systematically changing the weights given to various features and tracking how this affects the predictions made by the model. Through profiling the model with different weight configurations, IoT developers can learn which features have a major impact on the model's output. Evaluating the model's sensitivity to various features and their individual contributions is the fundamental concept. ProfWeight offers a sophisticated understanding of feature importance, which is very helpful when working with complex AI models.

Figure 19 shows the outcomes of different models under Profweight for N-BaIoT dataset. The significance scores of different top features in N-BaIoT are displayed on the y-axis; higher values signal that the feature is more important because permuting it has a bigger detrimental effect on the model's performance. The features are listed on the x-axis and reflect various statistical attributes of the dataset. Comparing all the AI models, the feature 'HH-L1-Magnitude' has the highest importance except with the MLP model, where 'HH-L3-mean' has the first rank (i.e., with the highest importance). Another interesting finding here is that RF model only depends on three main features ('HH-L1-magnitude', 'HH-L1-weight', and 'HH-L1-std') for making its classification decisions.

H. FEATURE IMPORTANCE USING LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS (LIME)

We next provide local feature importance using LIME. Designed to enable local interpretability for any machine learning model (i.e., LIME is model agnostic XAI method). By fitting a straightforward, interpretable model (such as a linear model) to roughly represent the behavior of the complex black-box model around a particular data point, LIME produces locally faithful explanations. This localized approximation improves the interpretability of the model at the local level by providing insights into how it functions for a specific instance. When working with complicated models for predicting anomalies for IoT networks or needing transparency in certain predictions without having to describe the entire model, LIME is especially helpful.

The feature importance scores as calculated by LIME for different AI models for N-BaIoT dataset are displayed in Figure 20. For any AI model, each bar on the graph represents a feature, and its length and direction show how much the feature contributed to the model's prediction for that particular instance. Positive values show a positive influence on the prediction, while negative values show a negative impact. The graph provides a clear image of how each element affects a specific prediction by. This is especially helpful for comprehending the behavior of the model on a local basis. Overall, the features "H1_L1_magnitude" and "H1_L3_std" are the top two features for LIME for most AI models except MLP that has "H1_L1_mean" as the top feature that affects its anomaly detection decisions.

I. PERFORMANCE METRICS WITH TOP-20 FEATURES

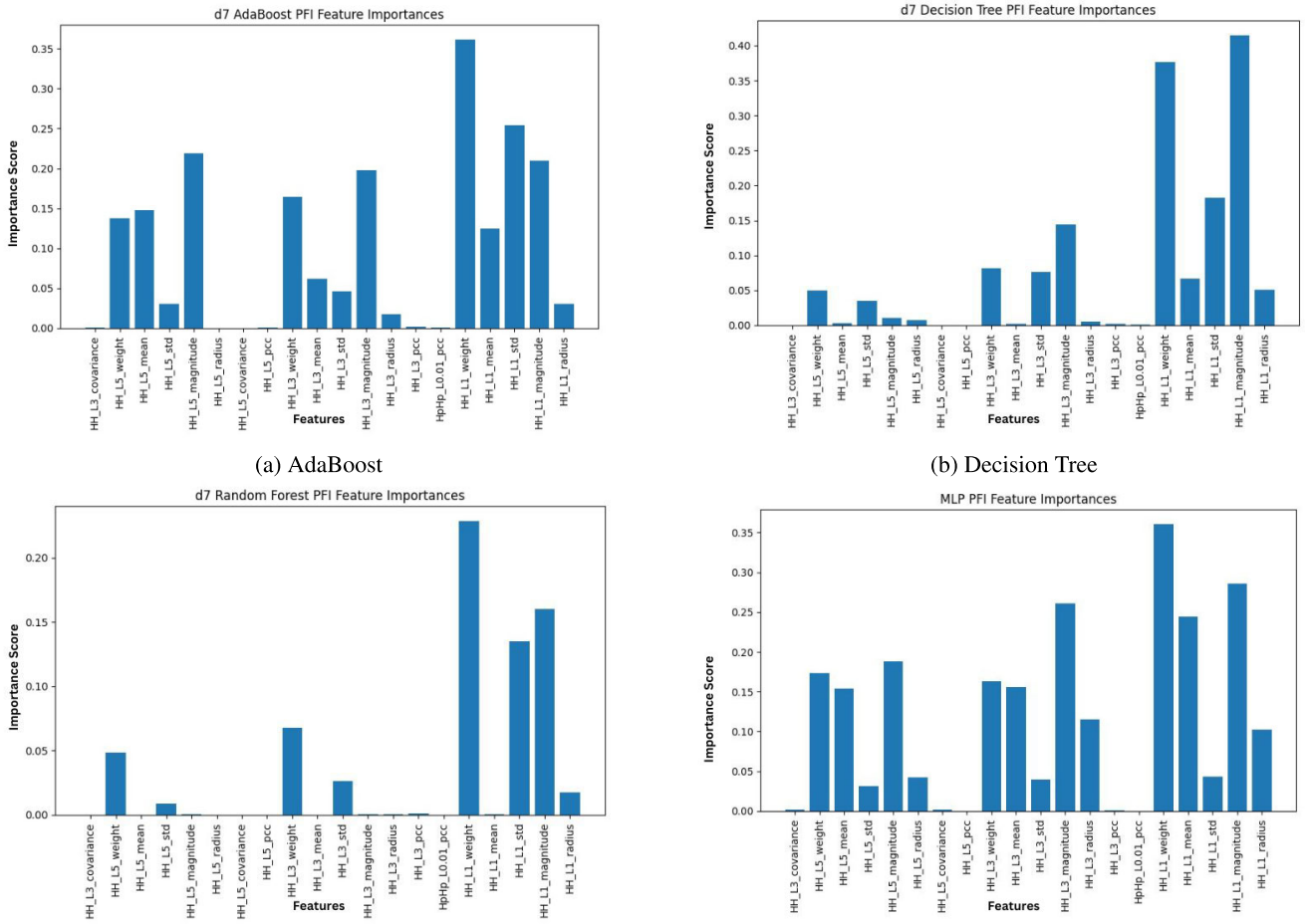
We next show the performance metrics under top-20 features for different AI methods to test the effect of generating top features for the N-BaIoT dataset. Table 8 shows such performance metrics under top-20 features for all nine IoT devices in the N-BaIoT dataset. Overall, the top-20 features have worse performance for all ensemble methods compared to using the full list of features. On the other hand, some of the models with worst performance under all features has improvements in some performance metrics in several devices under top-20 features. The main insight from this experiment is that feature selection can be more helpful for single AI models with lower performance, compared to ensemble methods. Moreover, the anomaly detection models seem to benefit from the majority of features when detecting anomalies for N-BaIoT dataset. Furthermore, such feature selection would depend on the nature of the IoT dataset.

J. FEATURE IMPORTANCE FOR EACH ATTACK TYPE

We next show the attack-specific feature importance (given by the top five features based on each attack type). Table 9 show such a list for N-BaIoT dataset. This is extracted from SHAP average scores for each attack using different AI models. We observe that the feature 'HH_L1_magnitude' is the common top feature across the different attack types. Furthermore, the feature 'HH_L1_weight' is the top feature for 'gagfyt.junk' and 'gagfyt.udp' botnet attacks. There are also other features that are common across different attack types (e.g., 'HH_L3_weight', 'HH_L3_magnitude', 'HH_L5_magnitude', 'HH_L1_mean', and 'HH_L3_mean'). Overall, these results show the main features that we should look for when investigating specific network botnet attack for IoT devices in the N-BaIoT dataset. This attack-specific feature importance can help in tuning AI models for detecting different conditions of the IoT network (here, the network traffic between host and destination IoT devices) given such feature importance knowledge.

K. SUMMARY OF RESULTS FOR N-BaIoT

We have provided overall feature importance analysis using five XAI methods (SHAP, LOCO, CEM, PFI, and ProfWeight). We have also provided local feature importance analysis using the popular LIME method. Furthermore, we provided performance metrics under different AI methods and different combinations of features (top-20 features, and all features). All things considered, features



(a) AdaBoost (b) Decision Tree

(c) Random Forest (d) MLP

FIGURE 18. Feature importance using PFI XAI method for the top features of N-BaIoT dataset using different AI models.

TABLE 9. The feature importance (given by top five features) for each attack type for N-BaIoT dataset. The feature *HH_L1_magnitude* is the common top feature across the different attack types.

Feature Rank	benign	gagfyt.combo	gagfyt.junk
1	HH_L1_magnitude	HH_L1_magnitude	HH_L1_weight
2	HH_L1_weight	HH_L3_magnitude	HH_L1_magnitude
3	HH_L3_magnitude	HH_L5_magnitude	HH_L3_weight
4	HH_L3_weight	HH_L3_mean	HH_L3_magnitude
5	HH_L5_magnitude	HH_L5_mean	HH_L1_mean
Feature Rank	**gagfyt.udp**	**gagfyt.tcp**	**gagfyt.scan**
1	HH_L1_weight	HH_L1_magnitude	HH_L1_magnitude
2	HH_L1_magnitude	HH_L3_magnitude	HH_L3_weight
3	HH_L3_weight	HH_L3_mean	HH_L1_weight
4	HH_L5_weight	HH_L5_magnitude	HH_L5_weight
5	HH_L3_magnitude	HH_L3_weight	HH_L1_std

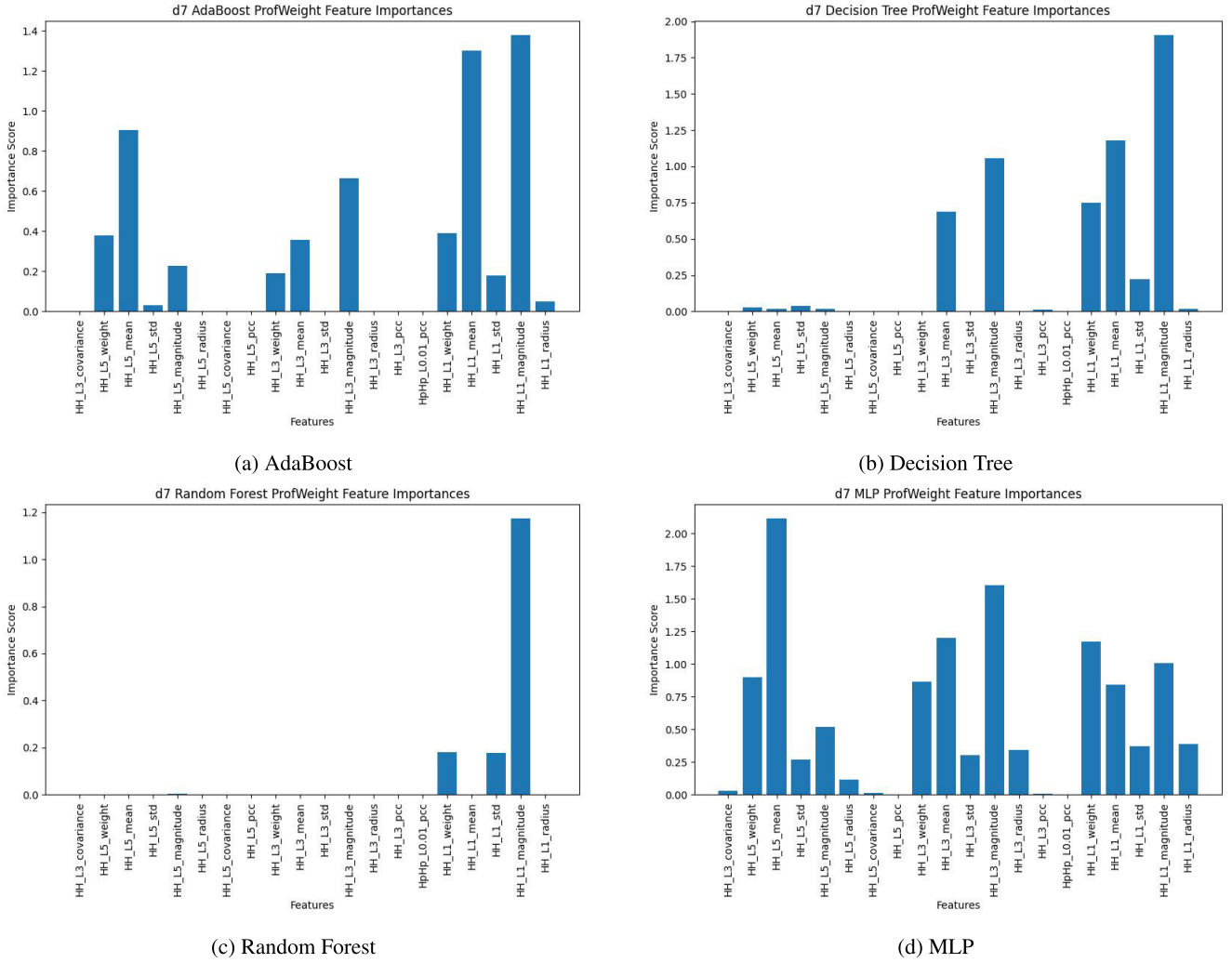
"H1_L1_magnitude", and "H1_L1_mean" have the most impact on the predictions across all models from feature importance results for different XAI methods considered in our evaluation for N-BaIoT dataset. Similar to MEMS, the link between features and predictions appears to be model-dependent, based on the diversity in feature impact across different models. Furthermore, the feature importance problem is more challenging in this dataset since it has 115 features compared to 3 features in MEMS dataset. Thus, different AI models seem to have different features that affect their decision-making.

Having provided our extensive evaluation for both MEMS and N-BaIoT datasets, we next provide a discussion of the main findings and main limitations of our work.

VII. DISCUSSION

A. OVERALL DISCUSSION OF RESULTS

We first observe that each dataset has a different best model (e.g., DNN gave the best performance for MEMS dataset while ensemble methods were the best for N-BaIoT dataset). We also observe that each XAI method can help in understanding one aspect of our anomaly detection problem. In particular, SHAP can give top features for each anomaly class, LIME can explain anomaly detection decisions on local samples of the IoT dataset, LOCO and Profweight can explain more the importance of each single feature, and ALE can provide correlation between each single feature and



(a) AdaBoost

(b) Decision Tree

(c) Random Forest

(d) MLP

FIGURE 19. Feature importance using Profweight XAI method for top features of N-BaIoT dataset using different AI models.

predictions. Our analysis provides main insights about the top features for each of the two datasets (MEMS and N-BaIoT), along with the performance of anomaly detection (represented by multi-class classification) for our two IoT datasets. Recall that we focused on sensors' readings for MEMS dataset and network traffic between IoT devices for N-BaIoT dataset, giving different angles of IoT systems.

B. DISCREPANCY IN MODELS' PERFORMANCES ON THE TWO DATASETS

There are several factors that explain the discrepancy in classifiers' performance between the N-BaIoT and MEMS datasets, which are detailed below.

1) DATASET SIZE AND BALANCE

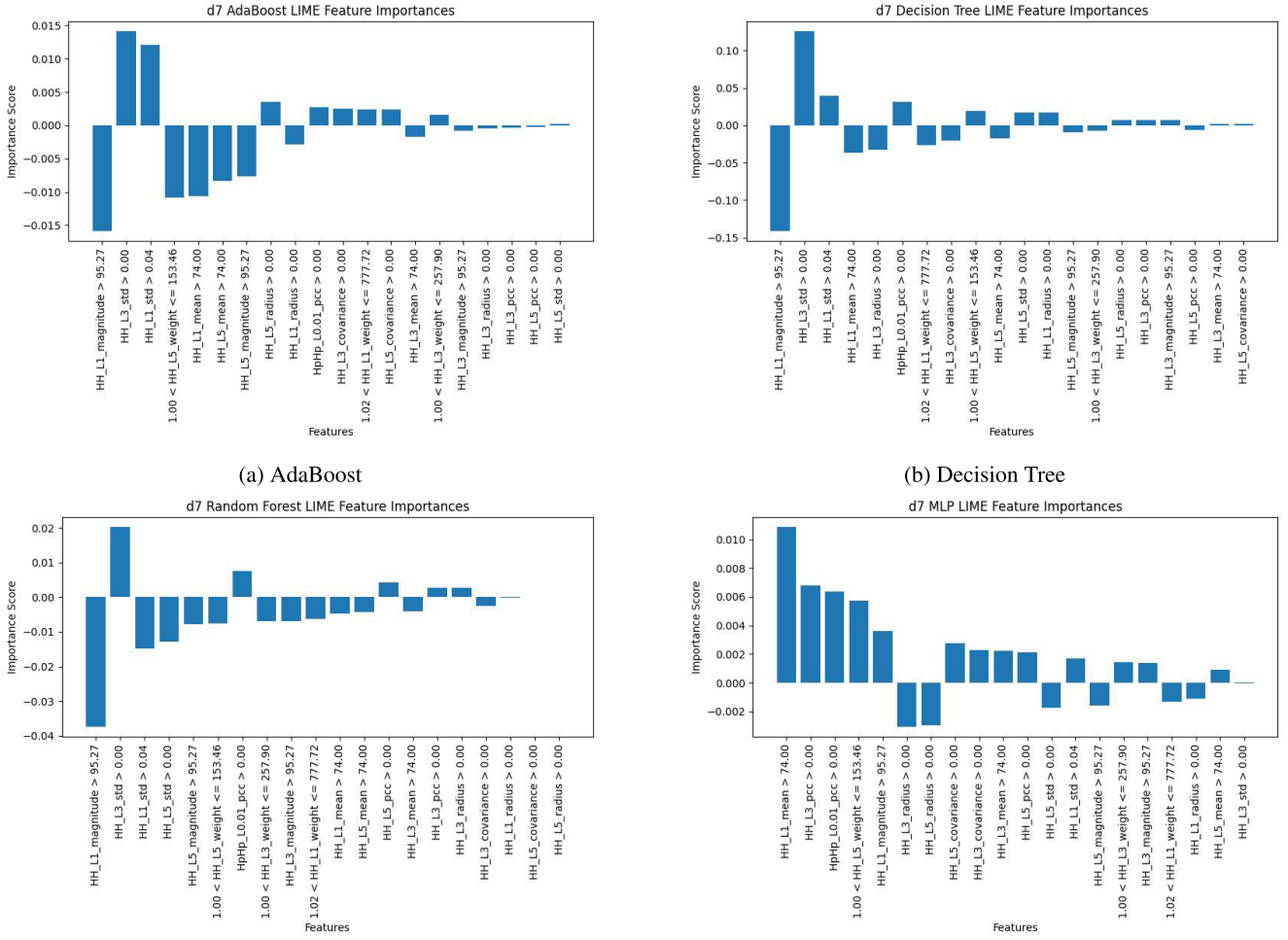
One important note is that bigger datasets typically result in greater model performance since they have more examples for the AI model to train from. This intuition is consistent with our analysis in which performances on N-BaIoT dataset is better than that of MEMS. This is due to the fact that the MEMS sensor dataset has only thousands samples, due to very low sampling rate (10 Hz) and thus its AI models' performances are much lower compared to N-BaIoT dataset (which has more than million and half samples).

2) CLASS OVERLAP AND DATA DISTRIBUTION

Compared to the N-BaIoT dataset, which have more distinct class separations, the MEMS dataset may have classes that are intrinsically closer together in feature space (such as "Near Failure" and "Failure"), making it more difficult for models to distinguish between them. Thus, better performance may also result from the fact that N-BaIoT dataset is having a more balanced distribution of classes than the MEMS dataset.

3) FEATURE IMPORTANCE

The 'z' feature was found to be the most important, and it had a major impact on model projections in MEMS dataset. Thus,



(a) AdaBoost (b) Decision Tree

(c) Random Forest (d) MLP

FIGURE 20. Feature importance using LIME XAI method for the top features of N-BaIoT dataset using different AI models.

the MEMS dataset might be more vulnerable to changes in this crucial characteristic, resulting in notable performance variations, because it only has three features, one of which is 'z'. On the other hand, the N-BaIoT dataset might be less impacted by changes in a single feature because it has a larger feature set, leading to more consistent performance.

4) DATA QUALITY

The N-BaIoT dataset may have characteristics that are less noisy or more discriminative for the anomaly detection task, which could explain why the classifiers performed better on it than on MEMS dataset.

C. FAIRNESS AND BIAS OF AI MODELS

We discuss here several metrics to measure the fairness and bias on the data collection technique and the accountability issues of AI models. These metrics include Demographic Parity Score, Predictive Parity Score, and Calibration Disparity Score [66]. We explain and measure such scores for both MEMS and N-BaIoT datasets considered in our work. These metrics and their related results are detailed below.

1) DEMOGRAPHIC PARITY SCORE

Demographic Parity, also known as statistical parity, evaluates whether outcomes are independent of protected attributes. It compares the proportion of positive outcomes across different labels from the dataset.

2) PREDICTIVE PARITY SCORE

Predictive parity examines whether the predictions made by the model are equally accurate across different labels. It assesses whether the model's performance is consistent regardless of the label.

3) CALIBRATION DISPARITY SCORE

Calibration metric assesses whether the predicted probabilities align with the true probabilities of outcomes. miscalibration across different groups can indicate bias in the AI model.

4) METRICS PERFORMANCES ON MEMS

We start by explaining the results of these metrics for our first dataset, MEMS dataset.

(a) Confusion Matrix: There are 4316 instances in the test set where the model's predictions match the true labels.

(b) Demographic Parity Score: 1.0

The Demographic Parity Score is 1.0, which indicates perfect demographic parity. This means that the proportion of positive predictions is the same across different groups.

(c) Predictive Parity Score: 0.0

The Predictive Parity Score is 0.0, which means that the positive predictive value (PPV) is equal between the groups of interest ('Near-failure' class) and the other group ('Normal' and 'Failure' classes). In other words, the model's predictions are equally accurate for both groups.

(d) Calibration Disparity Score: 1.0

The Calibration Disparity Score is 1.0, indicating perfect calibration disparity. This means that the calibration curves, which show the relationship between predicted probabilities and the true probabilities of positive outcomes, are the same for the group of interest ('Near-failure' class) and the other group ('Normal' and 'Failure' classes).

5) METRICS PERFORMANCES ON N-BaIoT

We finally explain the results of these metrics for our second dataset, N-BaIoT dataset.

(a) Confusion Matrix: There are 24,000 instances in the test set where the model's predictions match the true labels.

(b) Demographic Parity Score: 1.0

A Demographic Parity Score of 1.0 indicates perfect demographic parity. This means that the proportion of positive predictions is the same across different demographic groups. In this case, it suggests that the model's predictions are balanced across different groups.

(c) Predictive Parity Score: 0.0

The Predictive Parity Score of 0.0 suggests that there is no difference in the positive predictive value (PPV) between the group of interest (the 'gafgyt.scan' class) and the other group (remaining anomaly classes). It means that the model's predictions are equally accurate for both groups.

(d) Calibration Disparity Score: 1.0

A Calibration Disparity Score of 1.0 indicates perfect calibration disparity. This means that the calibration curves, which show the relationship between predicted probabilities and the true probabilities of positive outcomes, are the same for the group of interest ('gafgyt.scan class') and the other group (remaining classes). Essentially, it suggests that the model's predicted probabilities are well-calibrated for both groups.

Main Insight: Overall, these results suggest that the anomaly detection models are performing well in terms of fairness and bias according to the specified metrics for both datasets considered in our work.

D. CONFUSION OF AI MODELS BETWEEN ANOMALY CLASSES (CONFUSION MATRICES)

1) CONFUSION MATRICES COMPARISON

Here, we show the Random Forest AI model's performance confusion matrices for both IoT datasets in Figure 21. For the MEMS dataset, the confusion matrix (shown in Figure 21a) shows its three classes. Although it still tends to make accurate predictions, there is clear confusion between Class 1 (normal) and Class 2 (near failure), as seen by the lighter shades off the diagonal. Figure 21b shows RF model with six classes for the N-BaIoT datset, which shows high accuracy in some classes with most predictions focused along the diagonal, indicating accurate classifications. Interestingly, a higher percentage of true positives is suggested by the darker shades for classes like 1, 2, 3, and 5. Under such comparison of confusion matrices with the Random Forest algorithm as the anomaly classification algorithm, we show the differences in performances on the two datasets where N-BaIoT dataset have better prediction power compared to MEMS dataset (with lower number of samples).

2) MAIN EXPLANATIONS OF CONFUSION MATRIX OF MEMS DATASET

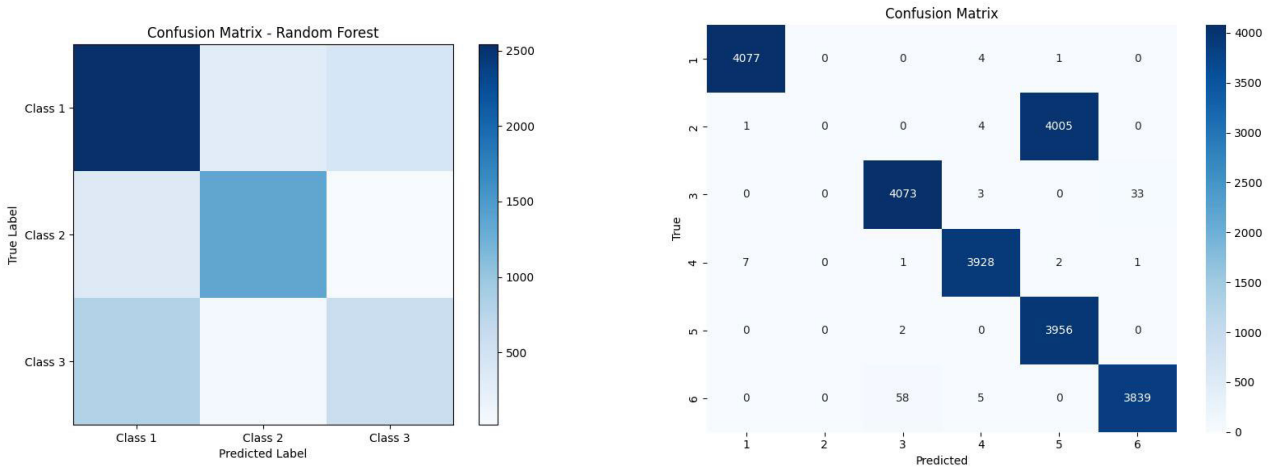
In the confusion matrix for the MEMS dataset (Figure 21a), there are three classes. Lighter hues in the off-diagonal cells show some misclassification across the classes, but substantially fewer than in the N-BaIoT dataset. The darker blue square in the upper left indicates a high number of true positives for class 1 (Normal). The lighter blue squares in these cells indicate that there is some uncertainty between class 2 (Near Failure) and class 3 (Failure), but this matrix still demonstrates a high degree of accuracy for class 1 (Normal) predictions.

3) MAIN EXPLANATIONS OF CONFUSION MATRIX OF N-BaIoT DATASET

Six class are displayed in the confusion matrix for the N-BaIoT dataset as seen in Figure 21b. The true positives for each class are shown by the darker hues along the diagonal, which show that the model correctly predicted each case. Class 1 (benign), for example, has 4077 true positives. Lighter hues in non-diagonal cells indicate misclassifications—erroneous predictions made by the model when it mistaken one class for another. The concentration of darker hues along the diagonal and fewer light spots elsewhere indicate that the RF model has a relatively low misclassification rate and high accuracy for both the benign class and the other attack classes for N-BaIoT dataset.

4) COMPARISON BETWEEN MEMS AND N-BaIoT MATRICES

The different shades of blue show the frequency of predictions where lighter cells indicate fewer occurrences, whereas darker cells indicate a high frequency of predictions for that combination of true and projected classes. Off-diagonal cells should be brighter, indicating fewer misclassifications,



(a) MEMS RF Matrix

(b) N-BaIoT Device 7 RF Matrix

FIGURE 21. Confusion matrices for the top classes of N-BaIoT and MEMS datasets using RF AI model.

whereas diagonal cells should ideally be the darkest, indicating accurate classifications. While the MEMS matrix (Figure 21a) demonstrates high performance primarily on one class with some misunderstanding between the other two classes, the other matrix of N-BaIoT (Figure 21b) reveals a model that is doing well across many classes.

E. DISCUSSION ABOUT PERFORMANCE AFTER FEATURE SELECTION

In our evaluation, we notice that most of the AI models work better under all features. However, we emphasize that the main goal of our feature importance analysis is understanding the top features that affect the decisions of AI models, even under the usage of most features in building these AI models. Eventually, this can lead to better explainable frameworks explaining decision-making of AI models.

F. NEED FOR XAI AND FEATURE IMPORTANCE FOR IoT

We emphasize that random forests and KNN, which are frequently thought to be interpretable in machine learning applications, also have interpretability issues. First, Random forests' ensemble nature leads to complicated decision processes that go beyond single tree clarity [67]. In low-dimensional spaces, KNN's simplicity contrasts with the interpretability concerns in high-dimensional spaces, where defining the 'nearest' neighbour becomes not clear [68]. Moreover, simple decision trees that have lower depth and containing a few leaves can be used as an explainability tool as shown in prior works for intrusion detection [69], [70], [71]. Since each level of the tree has a clear rule for each node until it reaches a leaf, it shows exactly the model decision process behind every decision which has improved explainability when compared to black-box AI models. However, this comes with much lower performance in terms of accuracy, recall, and precision. On the other hand, decision trees with higher depths and more leaves usually enhance performance metrics but do not have the explainability power of simpler decision trees. This sheds the light on the importance of having XAI methods to extract the main features for anomaly detection in IoT applications, which we tackle in our current work. For developers and users of these IoT devices, knowing the "why" behind a decision of an AI model is just as important as the decision itself.

There is also compelling economic rationale for the deployment and analysis of such XAI-based systems. In IoT-based smart manufacturing, a variety of sensors (e.g., vibration, ultrasonic, pressure sensors) are utilized for tasks like process control, automation, production planning, and equipment maintenance. For instance, in equipment maintenance, continuous monitoring of the operating equipment's condition using proxy measures (e.g., vibration and sound) is implemented to avert unplanned downtime and reduce maintenance costs [72]. Real-time analysis of data from these sensors and building XAI models for this data plays a pivotal role in predictive maintenance tasks, employing the anomaly detection process [73], [74].

In other IoT domains with IoT networks (such as N-BaIoT application), the manual intervention in data collection (e.g., replacing a failing sensor) or mitigation actions (e.g., adjusting the position of many IoT security cameras) is a costly endeavor. Consequently, precise anomaly detection can significantly alleviate these labor costs. Therefore, our proposed XAI-based anomaly detection technique finds applicability in different IoT applications.

G. COMPARATIVE ANALYSIS WITH PRIOR RELATED WORK

We now provide a comparative analysis between our current work and developed techniques with similar solutions by other scientists in anomaly detection and feature importance for IoT domain. Table 10 shows such a comparison where

TABLE 10. A comparative analysis of the available features between the prior related works in IoT systems and our framework. Our work provides an anomaly detection framework that incorporates detecting different attack types. Our framework also considers XAI-based feature importance analysis for both global and local scopes for different AI models.

Framework	Sensor Failure Detection	XAI Support	Benchmarking AI Models	Attack Type Classification	Feature Selection	Code Release (Opensource)
Alfeo et.al. [75]	Yes	No	No	No	No	No
Lee et.al. [44]	Yes	No	Yes	No	No	No
Teng et.al. [45]	Yes	No	No	Yes	No	No
Wang et.al. [76]	No	Yes	Yes	No	No	No
Udmale et.al. [51]	No	Yes	No	Yes	No	No
Fathi et.al. [56]	Yes	No	Yes	No	No	Yes
Lewis et.al. [57]	Yes	No	No	No	No	Yes
Kevin et.al. [77]	No	Yes	No	No	No	Yes
XAI-IoT (Ours)	Yes	Yes	Yes	Yes	Yes	Yes

it shows the main differences between the main features of our work and those of prior related works. Our work provides an anomaly detection framework that incorporates detecting different attack types. Our framework also considers XAI-based feature importance analysis for both global and local scopes for different AI models.

H. REPRODUCIBILITY

We have made our source codes and benchmark data publicly available, facilitating the replication of our research. We are releasing our IoT database corpus consisting of two datasets, aiming to encourage standardization in benchmarking anomaly detection and feature importance within this crucial domain. We invite the community to contribute to and expand this resource by sharing their new datasets and models. The website containing our database and source codes can be accessed at: https://github.com/agummadi1/XAI_for_IoT_Systems. Recall that detailed information about our framework and various model categories has been provided in Section IV-A. Additionally, the descritions of the two datasets is available in Appendix A, and details about hyper-parameter selections and libraries used are presented in Appendix B.

VIII. CONCLUSION

This paper explored several interesting challenges to an important application area, internet of things (IoT). We proposed an explainable AI framework for studying *anomaly detection* and *failure classification* for securing IoT systems. We proposed a multi-class anomaly detection technique and an efficient defect-type classification technique for IoT applications. We then performed feature importance analysis using seven XAI methods (SHAP, LIME, CEM, LOCO, PFI, and Profweight, ALE). We tested our framework on two real-world data-sets (MEMS, and N-BaIoT). We compared single AI and ensemble-based models for anomaly detection using different performance metrics. Our evaluation showed that the single AI models lead to better anomaly detection prediction for MEMS dataset while ensemble-based models were better for N-BaIoT dataset. We also identified the top features that affect the decision of different AI models for both datasets using our different XAI methods.

We release our database corpus and codes for the community to build on it with new datasets and models. We believe that the XAI framework is useful in IoT domain, especially when large anomaly detection datasets can be costly to collect and are normally thought to be very specific to a single application. Future avenues of research include leveraging the data from multiple IoT sensors, exploring ensemble learning on XAI methods to enhance feature analysis, and detecting the device health by merging information from multiple, potentially different, IoT sensors.

APPENDIX A
MEMS DATASET COLLECTION

Raw data was collected from the real-world sensors from August 2021–May 2022. Hardware (sensors mounted on motor testbeds) and software (code to collect the data from the sensors and save them to a desktop machine).

A. MOTIVATION FOR DATA RELEASE OF MEMS DATASET

The main motivation for releasing our datasets is for performing ML-based anomaly detection and feature importance for IoT-based smart manufacturing systems. The manufacturing of discrete products typically involves the use of equipment termed machine tools. The health of a machine is often directly related to the health of the motors being used to drive the process. Given this dependence, health studies of manufacturing equipment may work directly with equipment in a production environment or in a more controlled environment on a ‘‘motor testbed’’.

APPENDIX B
BENCHMARKS: MODELS, AND HYPER-PARAMETER SELECTION

A. MODELS AND HYPER-PARAMETER SELECTION

We now provide details on the models used to study the anomaly detection problem in our work. We explain the anomaly detection (prediction) algorithms and the hyper-parameters used for each classification model. This can help reproducing our results for the future related works. Following standard tuning of the model, we created different variants of the models to choose the best parameters (by comparing the performance of the multi-class classification problem). The details for each model can are explained below.

(1) Deep Neural Network (DNN): The initial classifier is a Deep Neural Network (DNN) with an architecture comprising an input layer, where the count of neurons

corresponds to the number of features used, employing the Rectified Linear Unit (ReLU) activation function. This is succeeded by a dropout layer with a dropout rate of 0.01, a hidden layer featuring 16 neurons and ReLU activation, and concludes with a ''softmax'' layer. The loss function is set to ''categorical_crossentropy,'' utilizing the adaptive momentum (ADAM) optimization algorithm. Training the model requires eleven epochs with a batch size of 1024, and default values are retained for the remaining parameters.

(2) Random Forest (RF): The subsequent classifier for detecting malicious samples in IoT datasets (network traffic in N-BaIoT and sensor samples in MEMS) is the RandomForest (RF). Hyperparameters include setting the number of estimators (trees) to 100, maximum tree depth to 10, and the minimum number of samples required to split an internal node to 2. Default values are maintained for the rest of the parameters.

(3) AdaBoost (ADA): AdaBoost is employed as the next classifier with the maximum number of estimators set at 50, the weight applied to each classifier during boosting iterations set to 1, and the base estimator being the Decision_Tree_Classifier.

(4) Decision Tree (DT): The subsequent classifier for detecting malicious samples in IoT datasets is the Decision Tree (DT). Hyperparameters include setting the maximum tree depth to 10, and the minimum number of samples required to split an internal node to 2. Default values are maintained for the rest of the parameters.

(5) Support Vector Machine (SVM): SVM is utilized with a kernel set to 'linear', gamma to 0.5, probability set to 'True', and regularization set to 0.5.

(6) Multi-layer Perceptron (MLP): The MLP classifier adopts the same setup as DNN.

(7) Bagging: Bagging ensemble method was used with base_estimator as decision tree classifier, and n_estimators with 100, and random_state = 42. The remaining parameters were set to default.

(8) Voting: Voting ensemble method was used with three estimators which are bagging classifier, Adaboost classifier, and random forest classifier. The voting method was set to ''hard'', and the remaining parameters were set to default.

(9) Blending: Blending ensemble method was used with three base learners which are bagging classifier, Adaboost classifier, and random forest classifier. The blending method was choosing the base learner with the maximum prediction probability for each sample. The remaining parameters were set to default.

(10) Stacking: Stacking ensemble method was used with three base learners which are bagging classifier, Adaboost classifier, and random forest classifier. The meta_classifier method was set to LogisticRegression, and use_probas was set to 'True'. The remaining parameters were set to default.

(11) K-nearest Neighbour (KNN): The KNN classifier is used in one experiment (SHAP explanation for MEMS) with default hyperparameters: the number of neighbors is set to five, uniform weights, and the search algorithm is set to 'auto'.

ACKNOWLEDGMENT

All the opinions, findings, and recommendations expressed in this publication are those of the authors of the article and they do not reflect the opinions or views of sponsors.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, ''The Internet of Things: A survey,'' *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.
[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, ''Internet of Things (IoT): A vision, architectural elements, and future directions,'' *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.
[3] M. Abdallah, B.-G. Joung, W. J. Lee, C. Mousoulis, N. Raghunathan, A. Shakouri, J. W. Sutherland, and S. Bagchi, ''Anomaly detection and inter-sensor transfer learning on smart manufacturing datasets,'' *Sensors*, vol. 23, no. 1, p. 486, Jan. 2023.
[4] L. Barreto and A. Amaral, ''Smart farming: Cyber security challenges,'' in *Proc. Int. Conf. Intell. Syst. (IS)*, Sep. 2018, pp. 870–876.
[5] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, ''Deep learning for smart manufacturing: Methods and applications,'' *J. Manuf. Syst.*, vol. 48, pp. 144–156, Jul. 2018.
[6] J. S. Sunny, C. P. K. Patro, K. Karnani, S. C. Pingle, F. Lin, M. Anekoji, L. D. Jones, S. Kesari, and S. Ashili, ''Anomaly detection framework for wearables data: A perspective review on data concepts, data analysis algorithms and prospects,'' *Sensors*, vol. 22, no. 3, p. 756, Jan. 2022.
[7] T. E. Thomas, J. Koo, S. Chaterji, and S. Bagchi, ''Minerva: A reinforcement learning-based technique for optimal scheduling and bottleneck detection in distributed factory operations,'' in *Proc. 10th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2018, pp. 129–136.
[8] L. Scime and J. Beuth, ''Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm,'' *Additive Manuf.*, vol. 19, pp. 114–126, Jan. 2018.
[9] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal, ''IoT healthcare analytics: The importance of anomaly detection,'' in *Proc. IEEE 30th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2016, pp. 994–997.
[10] G. Shahzad, H. Yang, A. W. Ahmad, and C. Lee, ''Energy-Efficient intelligent street lighting system using traffic-adaptive control,'' *IEEE Sensors J.*, vol. 16, no. 13, pp. 5397–5405, Jul. 2016.
[11] R. Mitchell and I.-R. Chen, ''A survey of intrusion detection techniques for cyber-physical systems,'' *ACM Comput. Surveys*, vol. 46, no. 4, pp. 1–29, Apr. 2014.
[12] B. Chatterjee, D.-H. Seo, S. Chakraborty, S. Avlani, X. Jiang, H. Zhang, M. Abdallah, N. Raghunathan, C. Mousoulis, A. Shakouri, S. Bagchi, D. Peroulis, and S. Sen, ''Context-aware collaborative intelligence with spatio-temporal in-sensor-analytics for efficient communication in a large-area IoT testbed,'' *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6800–6814, 2020.
[13] V. Chandola, A. Banerjee, and V. Kumar, ''Anomaly detection: A survey,'' *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.
[14] F. Sabahi and A. Movaghar, ''Intrusion detection: A survey,'' in *Proc. 3rd Int. Conf. Syst. Netw. Commun.*, Oct. 2008, pp. 23–26.
[15] A. L. Bowler, S. Bakalis, and N. J. Watson, ''Monitoring mixing processes using ultrasonic sensors and machine learning,'' *Sensors*, vol. 20, no. 7, p. 1813, Mar. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/7/1813
[16] F. Lopez, M. Saez, Y. Shao, E. C. Balta, J. Moyne, Z. M. Mao, K. Barton, and D. Tilbury, ''Categorization of anomalies in smart manufacturing systems to support the selection of detection mechanisms,'' *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 1885–1892, Oct. 2017.
[17] A. Das and P. Rad, ''Opportunities and challenges in explainable artificial intelligence (XAI): A survey,'' 2020, *arXiv:2006.11371*.
[18] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, ''Big IoT data analytics: Architecture, opportunities, and open research challenges,'' *IEEE Access*, vol. 5, pp. 5247–5261, 2017.
[19] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang, ''Multitier fog computing with large-scale IoT data analytics for smart cities,'' *IEEE Internet Things J.*, vol. 5, no. 2, pp. 677–686, Apr. 2018.

[20] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, pp. 660–674, Jun. 1991.

[21] C. Tang, N. Luktarhan, and Y. Zhao, "SAAE-DNN: Deep learning method on intrusion detection," *Symmetry*, vol. 12, no. 10, p. 1695, Oct. 2020.

[22] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving AdaBoost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset," *J. Phys., Conf. Ser.*, vol. 1192, Mar. 2019, Art. no. 012018.

[23] P. Tao, Z. Sun, and Z. Sun, "An improved intrusion detection algorithm based on GA and SVM," *IEEE Access*, vol. 6, pp. 13624–13631, 2018.

[24] J. O. Mebawondu, O. D. Alowolodu, J. O. Mebawondu, and A. O. Adetunmbi, "Network intrusion detection system using supervised learning paradigm," *Sci. Afr.*, vol. 9, Sep. 2020, Art. no. e00497.

[25] S. Waskle, L. Parashar, and U. Singh, "Intrusion detection system using PCA with random forest approach," in *Proc. Int. Conf. Electron. Sustain. Commun. Syst. (ICESC)*, Jul. 2020, pp. 803–808.

[26] P. Bühlmann, "Bagging, boosting and ensemble methods," in *Handbook of Computational Statistics: Concepts and Methods*. Berlin, Germany: Springer, 2012, pp. 985–1022.

[27] Y. Wang, M. Bellus, J.-F. Geleyn, X. Ma, W. Tian, and F. Weidle, "A new method for generating initial condition perturbations in a regional ensemble prediction system: Blending," *Monthly Weather Rev.*, vol. 142, no. 5, pp. 2043–2059, May 2014.

[28] R. Lazzarini, H. Tianfield, and V. Charissis, "A stacking ensemble of deep learning models for IoT intrusion detection," *Knowl.-Based Syst.*, vol. 279, Nov. 2023, Art. no. 110941.

[29] H. G. Ayad and M. S. Kamel, "On voting-based consensus of cluster ensembles," *Pattern Recognit.*, vol. 43, no. 5, pp. 1943–1953, May 2010.

[30] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.

[31] J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman, "Distribution-free predictive inference for regression," *J. Amer. Stat. Assoc.*, vol. 113, no. 523, pp. 1094–1111, Jul. 2018.

[32] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das, "Explanations based on the missing: Towards contrastive explanations with pertinent negatives," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–12.

[33] S. R. Islam, W. Eberle, S. K. Ghafoor, and M. Ahmed, "Explainable artificial intelligence approaches: A survey," 2021, *arXiv:2101.09429*.

[34] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: A corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, May 2010.

[35] A. Dhurandhar, K. Shanmugam, R. Luss, and P. A. Olsen, "Improving simple models with confidence profiles," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.

[36] J. Dieber and S. Kirrane, "Why model why? Assessing the strengths and limitations of LIME," 2020, *arXiv:2012.00093*.

[37] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul. 2018.

[38] M. Antonakakis et al., "Understanding the Mirai botnet," in *Proc. 26th USENIX Secur. Symp. (USENIX Security)*, 2017, pp. 1093–1110.

[39] E. Cozzi, P.-A. Vervier, M. Dell'Amico, Y. Shen, L. Bilge, and D. Balzarotti, "The tangled genealogy of IoT malware," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2020, pp. 1–16.

[40] R. Jeff. *Considerations for Accelerometer Selection When Monitoring Complex Machinery Vibration*. Accessed: Sep. 30, 2019. [Online]. Available: http://www.vibration.org/Presentation/IMI%20Sensors%20Accel%20Presentation%200116.pdf

[41] A. Albarbar, S. Mekid, A. Starr, and R. Pietruszkiewicz, "Suitability of MEMS accelerometers for condition monitoring: An experimental study," *Sensors*, vol. 8, no. 2, pp. 784–799, Feb. 2008.

[42] Y. Himeur, K. Ghanem, A. Alsalemi, F. Bensaali, and A. Amira, "Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives," *Appl. Energy*, vol. 287, Apr. 2021, Art. no. 116601.

[43] M. Abdallah, W. Jae Lee, N. Raghunathan, C. Mousoulis, J. W. Sutherland, and S. Bagchi, "Anomaly detection through transfer learning in agriculture and manufacturing IoT systems," 2021, *arXiv:2102.05814*.

[44] W. J. Lee, G. P. Mendis, M. J. Triebe, and J. W. Sutherland, "Monitoring of a machining process using kernel principal component analysis and kernel density estimation," *J. Intell. Manuf.*, vol. 31, no. 5, pp. 1175–1189, Jun. 2020, doi: 10.1007/s10845-019-01504-w.

[45] S.-H. G. Teng and S.-Y. M. Ho, "Failure mode and effects analysis," *Int. J. Quality Rel. Manage.*, vol. 13, no. 5, pp. 8–26, 1996.

[46] W. J. Lee, H. Wu, A. Huang, and J. W. Sutherland, "Learning via acceleration spectrograms of a DC motor system with application to condition monitoring," *Int. J. Adv. Manuf. Technol.*, vol. 106, nos. 3–4, pp. 803–816, Jan. 2020, doi: 10.1007/s00170-019-04563-8.

[47] O. Arreche, T. Guntur, and M. Abdallah, "XAI-IDS: Towards proposing an explainable artificial intelligence framework for enhancing network intrusion detection systems," *Appl. Sci.*, vol. 14, no. 10, 2024, Art. no. 4170. [Online]. Available: https://www.mdpi.com/2076-3417/14/10/4170, doi: 10.3390/app14104170.

[48] H. K. Bharadwaj, A. Agarwal, V. Chamola, N. R. Lakkaniga, V. Hassija, M. Guizani, and B. Sikdar, "A review on the role of machine learning in enabling IoT based healthcare applications," *IEEE Access*, vol. 9, pp. 38859–38890, 2021.

[49] P. Nimbalkar and D. Kshirsagar, "Feature selection for intrusion detection system in Internet-of-Things (IoT)," *ICT Exp.*, vol. 7, no. 2, pp. 177–181, Jun. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959521000588

[50] A. Nazir, Z. Memon, T. Sadiq, H. Rahman, and I. U. Khan, "A novel feature-selection algorithm in IoT networks for intrusion detection," *Sensors*, vol. 23, no. 19, p. 8153, Sep. 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/19/8153

[51] S. S. Udmale, S. K. Singh, R. Singh, and A. K. Sangaiah, "Multi-fault bearing classification using sensors and ConvNet-based transfer learning approach," *IEEE Sensors J.*, vol. 20, no. 3, pp. 1433–1444, Feb. 2020.

[52] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. Hershey, PA, USA: IGI Global, 2010, pp. 242–264.

[53] M. Abdallah, R. Rossi, K. Mahadik, S. Kim, H. Zhao, and S. Bagchi, "AutoForecast: Automatic time-series forecasting model selection," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.* NY, USA: Association for Computing Machinery, Oct. 2022, pp. 5–14, doi: 10.1145/3511808.3557241.

[54] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," 2020, *arXiv:2006.05822*.

[55] R.-J. Hsieh, J. Chou, and C.-H. Ho, "Unsupervised online anomaly detection on multivariate sensing time series data for smart manufacturing," in *Proc. IEEE 12th Conf. Service-Oriented Comput. Appl. (SOCA)*, Nov. 2019, pp. 90–97.

[56] Y. Fathy, M. Jaber, and A. Brintrup, "Learning with imbalanced data in smart manufacturing: A comparative analysis," *IEEE Access*, vol. 9, pp. 2734–2757, 2021.

[57] L. Calderoni, A. Magnani, and D. Maio, "IoT manager: An open-source IoT framework for smart cities," *J. Syst. Archit.*, vol. 98, pp. 413–423, Sep. 2019.

[58] D. Han, Z. Wang, W. Chen, Y. Zhong, S. Wang, H. Zhang, J. Yang, X. Shi, and X. Yin, "DeepAID: Interpreting and improving deep learning-based anomaly detection in security applications," 2021, *arXiv:2109.11495*.

[59] K. A. Jackson, D. H. DuBois, and C. A. Stallings, "An expert system application for network intrusion detection," Los Alamos Nat. Lab. (LANL), Los Alamos, NM, USA, Tech. Rep. LA-UR-91-558; CONF-911059-1; ON: DE91008590, 1991.

[60] C. Wu, A. Qian, X. Dong, and Y. Zhang, "Feature-oriented design of visual analytics system for interpretable deep learning based intrusion detection," in *Proc. Int. Symp. Theor. Aspects Softw. Eng. (TASE)*, Dec. 2020, pp. 73–80.

[61] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018, *arXiv:1802.09089*.

[62] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1–13, Dec. 2020.

[63] A. Gulli and S. Pal, *Deep Learning With Keras*. Berlin, Germany: Packt, 2017.

[64] C. A. Stewart, V. Welch, B. Plale, G. C. Fox, M. Pierce, and T. Sterling, "Indiana University Pervasive Technology Institute," Indiana Univ., Blomingtoon, IN, USA, Tech. Rep., 2017.

[65] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study," *Data Mining Knowl. Discovery*, vol. 30, no. 4, pp. 891–927, Jul. 2016.

[66] C. Mougan, L. State, A. Ferrara, S. Ruggieri, and S. Staab, "Beyond demographic parity: Redefining equal treatment," 2023, *arXiv:2303.08040*.
[67] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.
[68] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Proc. 8th Int. Conf. Database Theory*, London, U.K. Cham, Switzerland: Springer, Jan. 2001, pp. 420–434.
[69] B. Ingre, A. Yadav, and A. K. Soni, "Decision tree based intrusion detection system for NSL-KDD dataset," in *Proc. Inf. Commun. Technol. Intell. Syst. (ICTIS)*, vol. 2. Berlin, Germany: Springer, 2017, pp. 207–218.
[70] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "RDTIDS: Rules and decision tree-based intrusion detection system for Internet-of-Things networks," *Future Internet*, vol. 12, no. 3, p. 44, Mar. 2020.
[71] M. Al-Omari, M. Rawashdeh, F. Qutaishat, M. Alshira'H, and N. Ababneh, "An intelligent tree-based intrusion detection model for cyber security," *J. Netw. Syst. Manage.*, vol. 29, no. 2, pp. 1–18, Apr. 2021.
[72] W. J. Lee, G. P. Mendis, and J. W. Sutherland, "Development of an intelligent tool condition monitoring system to identify manufacturing tradeoffs and optimal machining conditions," in *Proc. 16th Global Conf. Sustain. Manuf.*, vol. 33, 2019, pp. 256–263.
[73] M. C. Garcia, M. A. Sanz-Bobi, and J. Del Pico, "SIMAP: Intelligent system for predictive maintenance: Application to the health condition monitoring of a windturbine gearbox," *Comput. Ind.*, vol. 57, no. 6, pp. 552–568, 2006.
[74] M. De Benedetti, F. Leonardi, F. Messina, C. Santoro, and A. Vasilakos, "Anomaly detection and predictive maintenance for photovoltaic systems," *Neurocomputing*, vol. 310, pp. 59–68, Oct. 2018.
[75] A. L. Alfeo, M. G. C. A. Cimino, G. Manco, E. Ritacco, and G. Vaglini, "Using an autoencoder in the design of an anomaly detector for smart manufacturing," *Pattern Recognit. Lett.*, vol. 136, pp. 272–278, Aug. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865520302269
[76] S. Wang and N. Xi, "Calibration of haptic sensors using transfer learning," *IEEE Sensors J.*, vol. 21, no. 2, pp. 2003–2012, Jan. 2021.
[77] K. I. Wang, X. Zhou, W. Liang, Z. Yan, and J. She, "Federated transfer learning based cross-domain prediction for smart manufacturing," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 4088–4096, Jun. 2022.

ANNA NAMRITA GUMMADI received the bachelor's degree in computer science engineering from Jawaharlal Nehru Technological University (JNTU) Hyderabad, India. She is currently pursuing the master's degree in cybersecurity and trusted systems with the Purdue School of Engineering and Technology, Indiana University-Purdue University Indianapolis (IUPUI). She is a Research Assistant focusing on cybersecurity and explainable AI. Having spent almost three years as an Associate Software Engineer, she has hands-on experience with technology's practical applications. She aspires to leverage her expertise in creating transparent and ethically-driven technological advancements.

JERRY C. NAPIER is currently pursuing the bachelor's degree in computer information technology (CIT) with the Purdue School of Engineering and Technology, Indiana University-Purdue University Indianapolis (IUPUI). He is a current participant in IUPUI's 1st Year Research Immersion Program, where he has the opportunity to gain important research, problem-solving, and critical thinking skills. He plans on continuing his studies in CIT and hopes to gain valuable experience during his time at IUPUI. His research interests include network design, information security, and cybersecurity.

MUSTAFA ABDALLAH (Member, IEEE) received the M.Sc. degree in engineering mathematics from the Faculty of Engineering, Cairo University, and the Ph.D. degree from the Elmore Family School of Electrical and Computer Engineering, Purdue University. He is currently an Assistant Professor with the Purdue School of Engineering and Technology, Indiana University-Purdue University Indianapolis (IUPUI). He has also several industrial research experiences, including internships with the Adobe Research, and a Principal, and a five-year machine learning research experience with RDI (a leading machine learning company in Egypt). His research interests include game theory, human decision-making, explainable AI, and machine learning with applications, including network security and autonomous driving systems. His research contribution is recognized by receiving the Purdue Bilsland Dissertation Fellowship and having many publications in top IEEE/ACM journals and conferences. He was a recipient of the M.Sc. Fellowship from the Faculty of Engineering, Cairo University, in 2013.