Spot the Glitch: Anomaly Detection in IoT Networks

Ashmit Pareek, Ravjot Singh, Sheik Mohammed Azaruddin, Srushtee Upashi, Srushti Doshi
San José State University
Department of Applied Data Science

***Abstract*—The rapid expansion of Internet of Things (IoT) devices has transformed various industries by enabling enhanced connectivity and data-driven insights. However, this growth has also made IoT networks vulnerable to numerous security threats and anomalies. This project addresses the crucial need for effective anomaly detection in IoT networks to maintain their security and reliability. We propose a robust machine learning-based approach to identify anomalies in IoT network traffic. Our methodology involves comprehensive data preprocessing, feature engineering, and the deployment of multiple machine learning models, including K-Nearest Neighbors (KNN), Random Forest, and Decision Tree classifiers. The models are trained and evaluated using the IoT-23 dataset, a real-world dataset specifically designed for IoT security research. Evaluation metrics such as accuracy, precision, recall, and log loss are employed to assess model performance. The results demonstrate that our machine learning models can effectively distinguish between normal and anomalous network behavior, highlighting the KNN model's superior performance in terms of accuracy and recall. This study underscores the potential of machine learning techniques in enhancing IoT network security by providing accurate and timely detection of anomalies. The findings contribute to the development of more secure and resilient IoT systems, offering valuable insights for future research and practical applications in IoT environments. Our work emphasizes the importance of continuous monitoring and advanced analytics in safeguarding IoT networks against evolving threats.**

***Index Terms*— Internet of Things, Security Threats, K-Nearest Neighbours, Decision Tree, Random Forest, Performance Metrics.**

I. Motivation

The integration of Internet of Things (IoT) technology into our daily lives and industries has significantly increased the complexity and volume of network traffic, resulting in a parallel rise in the variety and frequency of network security threats. IoT devices, often being resource-constrained and intermittently managed for security, present unique challenges and vulnerabilities. These vulnerabilities make them attractive targets for various cyberattacks, which can compromise personal privacy, corporate security, and even national safety.[1]

Given this context, there is a critical need for robust security measures that can proactively detect and mitigate such threats before they can cause harm. Anomaly detection in IoT networks stands out as a crucial defensive strategy because it allows for the identification of unusual patterns that may signify a security breach. Traditional security measures often fail to keep pace with the sophistication of modern cyberattacks, making the case for advanced solutions like machine learning-based anomaly detection even stronger.[2]

By focusing on machine learning, this project aims to leverage computational efficiency and adaptive learning capabilities to recognize and respond to anomalous activities in real time, thereby enhancing the resilience of IoT networks against a spectrum of cyber threats.

II. Background

IoT security is a multi-faceted challenge that includes vulnerabilities at the device, network, and data levels. Devices are often shipped with outdated software, have weak authentication protocols, or lack the capability to be patched against security vulnerabilities.[3] At the network level, the sheer volume and variety of IoT traffic can obscure malicious activities, making it difficult for traditional security tools to detect anomalies effectively.

In this challenging landscape, the IoT-23 dataset serves as a pivotal resource for the research and development of anomaly detection models. This dataset, compiled with a focus on IoT security,

categorizes network behaviors into several types of attacks along with benign traffic. The specific attacks recorded include:

- *Benign*: Regular and non-malicious network activities that should not trigger security alarms.
- *PartOfAHorizontalPortScan*: Attempts to discover open ports across multiple devices, often a precursor to more serious attacks.
- *C&C (Command and Control)*: Traffic indicative of control commands being sent to compromised devices, usually part of a botnet.
- *Attack*: General malicious activities aimed at disruption or unauthorized access.
- *Okiru*: A malware strain targeting IoT devices, part of the Mirai botnet family.
- *DDoS (Distributed Denial of Service)*: Overwhelming a network or service with high volumes of data traffic to render it non-functional.

These classifications in the IoT-23 dataset not only help in training machine learning models to detect and differentiate between normal and malicious activities but also provide a real-world context that enhances the practical value of research outcomes.[4] By applying advanced analytical models to such detailed and specific data, this project aims to contribute significantly to the ongoing efforts in securing IoT environments against emerging and evolving cyber threats.

III. Literature Survey

The recent surge in research on IoT anomaly detection has focused on various machine learning techniques and architectures to enhance the security and reliability of IoT systems. The application of deep learning, particularly using autoencoders, has been significant due to their ability to learn optimal representations of normal behavior, thus identifying anomalies when deviations occur. The versatility of deep learning models, especially in unsupervised settings where labels might not be available, makes them particularly suitable for the dynamic and diverse data generated by IoT devices. [5]

Active learning has also been highlighted as a powerful approach in this domain. By selectively querying labels for the most informative data points, active learning models can efficiently adapt to new and evolving types of attacks, significantly reducing the need for exhaustive manual labeling. This adaptability is crucial for IoT environments where new devices and behaviors are constantly introduced. [6]

Furthermore, there has been a notable shift towards integrating machine learning with blockchain technologies to bolster IoT security. This combination promises enhanced security and data integrity by decentralizing the anomaly detection process, thereby making it more difficult for attackers to manipulate or disrupt the system. [7]

Another critical advancement is in network-based anomaly detection, where machine learning models are applied to network traffic data to identify potentially malicious activities. This approach leverages the growing amount of traffic data and sophisticated pattern recognition capabilities of modern algorithms to improve detection rates and response times to threats. [3]

Overall, the literature emphasizes a trend towards models that not only detect known types of attacks but are also capable of adapting to detect new, previously unseen types of anomalies. This adaptability is achieved through advanced machine learning techniques that learn from the data itself, without heavily relying on predefined rules or signatures.

These advancements represent a substantial progression in the field of IoT security, suggesting a move towards more autonomous, robust, and adaptive security mechanisms that are better suited to the complex and ever-evolving landscape of the Internet of Things.

IV. Methodology

The methodology for this research incorporates comprehensive techniques for data collection, preprocessing, modeling, and training. This section elaborates on the models employed to address the challenge of categorizing types of attacks based on details of sensors.

A. Data Collection

The dataset utilized in this study originates from the IoT23 dataset, a recent compilation published in

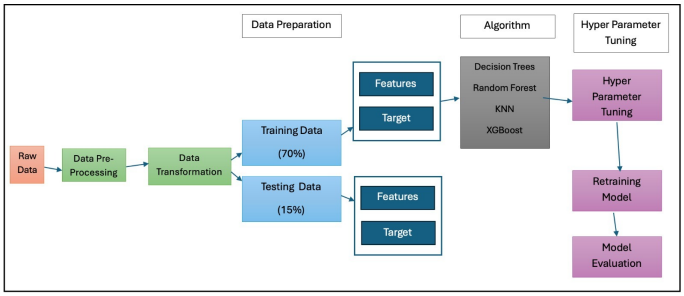

Fig. 1. Methodology

January 2020. This dataset encompasses network traffic data from three distinct smart home IoT devices: Amazon Echo, Philips HUE, and Somfy Door Lock. It represents a substantial collection of real and labeled IoT malware infections and benign traffic, specifically curated for the development of machine learning algorithms. Comprising 23 captures (also referred to as scenarios), the dataset includes 20 malicious captures and 3 benign captures. Each capture from infected devices may bear the name of the potential malware sample executed within that scenario.

The malware labels within the IoT-23 dataset encompass a variety of categories, including Attack, C&C, FileDownload, HeartBeatAttack, Mirai, Torii, DDoS, FileDownload, Okiru, Okiru-Attack, and PartOfAHorizontalPortScan. Furthermore, Zeek software is employed for network analysis. The IoT-23 dataset is provided in the format of conn.log.labeled, derived from the Zeek conn.log file generated by the Zeek network analyzer using the original pcap file. The variable types and definitions for the IoT-23 dataset are detailed in figure 2.

Given the substantial size of the dataset, a decision was made to extract a portion of records from each individual dataset and merge them into a new dataset. This approach facilitates the handling of computational workload for the new dataset while retaining the majority of attack types present in the IoT-23 dataset.

B. Data Preprocessing

Effective data preprocessing is essential for enhancing model performance in IoT anomaly

Column Name	Data Type
Unnamed: 0	int64
duration	float64
orig_bytes	int64
resp_bytes	int64
missed_bytes	float64
orig_pkts	float64
orig_ip_bytes	float64
resp_pkts	float64
resp_ip_bytes	float64
label	object
proto_icmp	bool
proto_tcp	bool
proto_udp	bool
conn_state_OTH	bool
conn_state_REJ	bool
conn_state_RSTO	bool
conn_state_RSTOS0	bool
conn_state_RSTR	bool
conn_state_RSTRH	bool
conn_state_S0	bool
conn_state_S1	bool
conn_state_S2	bool
conn_state_S3	bool
conn_state_SF	bool
conn_state_SH	bool
conn_state_SHR	bool

Fig. 2. Data Info

detection. Here are the key steps followed:

- *Cleaning Data*: The dataset was first cleansed of duplicate entries to ensure the uniqueness of the data, which helps in preventing model overfitting and bias. Missing values were addressed by applying appropriate imputation techniques (like replacing with median or mode) or, in cases of extensive missingness, by removing the affected rows or columns to maintain data integrity. Data from various sources often come in different formats. Converting all data into a consistent format (e.g., converting all categorical data to numerical where necessary) ensures compatibility with machine learning algorithms.

- *Data Encoding*: The target variable, representing different traffic types (benign or various attacks), was clearly labeled. This labeling aids in supervised learning, where the model predicts the traffic type based on input features. For models like XGBoost, which prefer numerical input, categorical variables were transformed into numerical codes through

label encoding, enabling the algorithm to process them effectively.

- *Balancing Data*: Given the imbalance typically present in datasets involving security threats, where attacks are less frequent than benign instances, Synthetic Minority Over-sampling Technique (SMOTE) was used. This technique generates synthetic samples from the minority class to balance the dataset, thus preventing the models from being biased toward the majority class. Balancing the classes helps in improving the model's ability to identify less frequent, yet critical, attack instances.

C. Modeling

To effectively detect anomalies in IoT network traffic, we employed several machine learning models, each chosen for its unique strengths and applicability to the problem at hand. Here is a detailed explanation of each algorithm used:

- *K-Nearest Neighbors*: K-Nearest Neighbors is a simple, yet powerful, non-parametric algorithm used in classification tasks. It classifies a new data point based on the majority vote of its 'k' nearest neighbors, where 'k' is a user-defined constant. The nearness of samples is typically determined by a distance metric such as Euclidean distance. KNN is particularly effective for datasets where similar instances generally belong to the same class. However, its performance can degrade with high-dimensional data due to the curse of dimensionality.[4]

- *Decision Tree*: A Decision Tree is a flowchart-like tree structure where each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. This model is highly interpretable, but it can easily overfit, especially with complex trees. Techniques such as pruning are often used to enhance generalization.[8]

- *Random Forest*: Random Forest is an ensemble learning technique that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes of the individual trees. It introduces randomness into the model building process and typically provides a better predictive performance than a single decision tree by reducing over-fitting. Each tree in the forest is built from a sample drawn with replacement from the training set. Additionally, when splitting a node during the construction of the trees, the best split is chosen from a random subset of the features.[9]

- *XGBoost*: XGBoost stands for eXtreme Gradient Boosting and is an implementation of gradient boosted decision trees designed for speed and performance. It is a scalable and highly efficient version of gradient boosting and has proven to be effective across a wide range of data mining tasks. XGBoost improves upon the base gradient boosting framework through system optimization and algorithmic enhancements, such as handling missing data and pruning trees using depth-first approach.[10]

D. Training

The data was divided into a 70:30 train-test split to ensure both sufficient training data and an unbiased evaluation of the model's performance. Hyperparameter tuning is a crucial step in optimizing machine learning models. It involves selecting the optimal set of hyperparameters that minimize a predefined loss function or maximize performance metrics such as accuracy, precision, or recall.

- *K-Nearest Neighbors*: The main hyperparameter is 'k', the number of neighbors. A grid search was used to experiment with different values of 'k' to find the optimal setting that minimizes the error rate.

- *Random Forest*: Parameters like the number of trees, maximum depth of the trees, and the maximum number of features used for the best split were tuned. This was achieved through randomized search and cross-validation, optimizing for both performance and computational efficiency.

E. Testing and Evaluation

Machine learning models, including Decision Trees, K Nearest Neighbors, Random Forest, and XGBoost, in both their base and hyperparameter-tuned configurations were rigorously analyzed with a focus on performance on a few key parameters: accuracy, precision, recall, and weighted F1 scores. These metrics are a key criterion for evaluating the effectiveness of each model in solving the classification task, where both balanced classes and correct predictions are important.

- *K-Nearest Neighbors*: As we can see in figure 3 and figure 4 the K Nearest Neighbors model displayed superior performance, particularly in its tuned configuration. The base model already performed well with an accuracy of 91.54% and an F1-score of 0.92, highlighting its effectiveness in properly partitioning samples and balancing between accuracy and recall.

KNN model trained in 0.10 seconds.
Accuracy of KNN Model: 0.9154441727791361
Classification Report KNN Model:

	precision	recall	f1-score	support
Attack	1.00	1.00	1.00	4846
Benign	0.80	0.68	0.73	4980
C&C	0.74	0.84	0.79	4822
DDoS	1.00	1.00	1.00	4970
Okiru	1.00	1.00	1.00	4957
PartOfAHorizontalPortScan	0.97	0.98	0.97	4873
accuracy			0.92	29448
macro avg	0.92	0.92	0.91	29448
weighted avg	0.92	0.92	0.91	29448

Fig. 3. KNN performance metrics

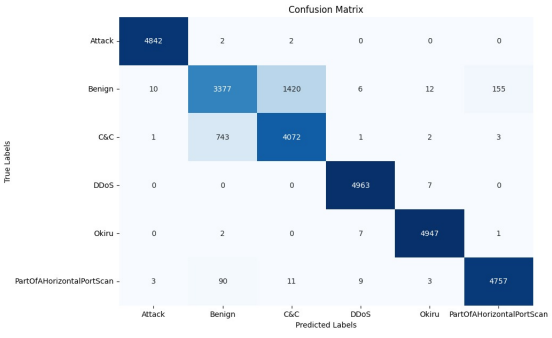

Fig. 4. KNN confusion matrix

Tuning further improved these metrics, and increased the accuracy to 92.02% ,F1- The score was maintained at 0.92 as shown in figure 5 and figure 6

Best K value: 9
Accuracy of KNN Model: 0.9202322738386308
Classification Report KNN Model:

	precision	recall	f1-score	support
Attack	1.00	1.00	1.00	4846
Benign	0.83	0.68	0.75	4980
C&C	0.75	0.87	0.80	4822
DDoS	0.99	1.00	1.00	4970
Okiru	0.99	1.00	1.00	4957
PartOfAHorizontalPortScan	0.97	0.98	0.97	4873
accuracy			0.92	29448
macro avg	0.92	0.92	0.92	29448
weighted avg	0.92	0.92	0.92	29448

Fig. 5. KNN performance metrics (hyperparameter tuning k=9)

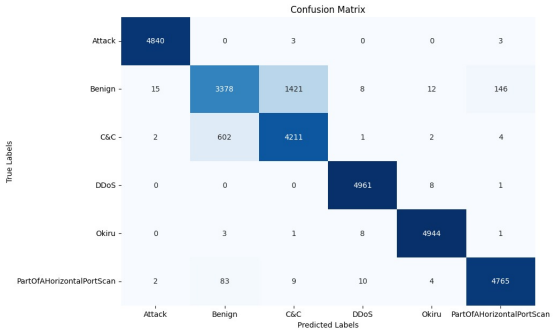

Fig. 6. KNN confusion matrix (hyperparameter tuning k=9)

- *Decision Tree*: The Decision Trees model demonstrated commendable performance in its base configuration, achieving an accuracy of 88.23%. This high level of accuracy indicates the model's capability to correctly classify a large proportion of instances. However, its weighted average F1-score of 0.88, while robust, suggests room for improvement in balancing precision and recall, especially in datasets with uneven class distributions. We can see this in figure 7 and figure 8

Accuracy of DT Model: 0.8823010051616409
Classification Report DT Model:

	precision	recall	f1-score	support
Attack	1.00	1.00	1.00	4846
Benign	0.66	0.65	0.65	4980
C&C	0.68	0.69	0.69	4822
DDoS	1.00	1.00	1.00	4970
Okiru	1.00	1.00	1.00	4957
PartOfAHorizontalPortScan	0.96	0.95	0.96	4873
accuracy			0.88	29448
macro avg	0.88	0.88	0.88	29448
weighted avg	0.88	0.88	0.88	29448

Fig. 7. Decision Tree performance metrics

- *Random Forest*: The Random Forest model,

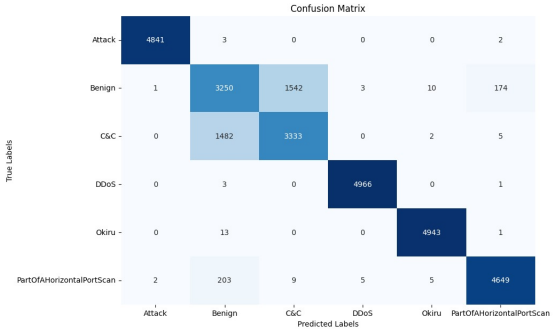

Fig. 8. Decision Tree confusion matrix

known for its robustness due to the ensemble approach, recorded lower metrics compared to the KNN model. The base RF model's accuracy stood at 85.18% with a weighted average F1-score of 0.85 as we can see in figure 9 and figure 10. After tuning, slight improvements

```
RF model trained in 4.26 seconds.
Accuracy of RF Model: 0.8518065743004618
Classification Report RF Model:
                        precision    recall  f1-score   support

                Attack       1.00      0.99      0.99      4846
                Benign       0.98      0.22      0.36      4980
                   C&C       0.57      0.98      0.72      4822
                  DDoS       1.00      0.99      1.00      4970
                 Okiru       0.90      0.99      0.94      4957
PartOfAHorizontalPortScan     0.95      0.95      0.95      4873

              accuracy                           0.85     29448
             macro avg       0.90      0.85      0.83     29448
          weighted avg       0.90      0.85      0.83     29448
```

Fig. 9. Decision Tree performance metrics

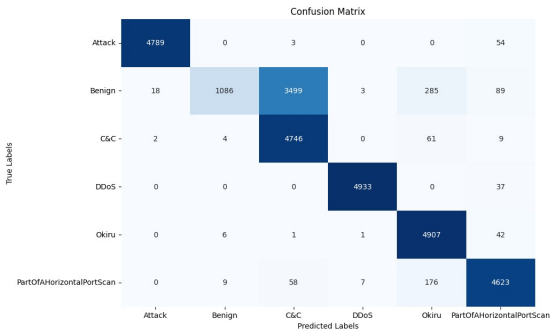

Fig. 10. Decision Tree confusion matrix

were noted, bringing the accuracy to 85.63% and the F1-score to 0.86 as we can see in figure 11 and figure 12. These figures reflect the model's general reliability but also indicate that it may not be as responsive to hyperparameter tuning as the KNN model, particularly in this dataset.

```
RF model trained in 7.05 seconds.
Accuracy of RF Model: 0.8563569682151589
Classification Report RF Model:
                        precision    recall  f1-score   support

                Attack       1.00      0.99      0.99      4846
                Benign       0.97      0.23      0.37      4980
                   C&C       0.58      0.98      0.73      4822
                  DDoS       1.00      1.00      1.00      4970
                 Okiru       0.90      1.00      0.95      4957
PartOfAHorizontalPortScan     0.97      0.96      0.96      4873

              accuracy                           0.86     29448
             macro avg       0.90      0.86      0.83     29448
          weighted avg       0.90      0.86      0.83     29448
```

Fig. 11. Random Forest performance metrics (hyperparameter tuning)

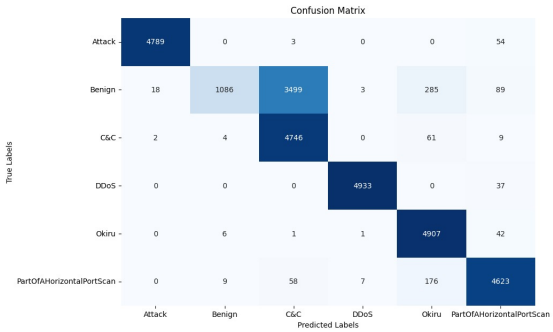

Fig. 12. Random Forest confusion matrix (hyperparameter tuning)

- *XGBoost*: The XGBoost model, often praised for its performance in classification tasks, showed a solid base model accuracy of 87.65% and a weighted average F1-score of 0.87 which we can see in figure 13 and 14.

```
XGBoost model trained in 1.20 seconds.
Accuracy of XGBoost Model: 0.8814860092366205
Classification Report XGBoost Model:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4846
           1       0.68      0.58      0.63      4980
           2       0.65      0.73      0.69      4822
           3       1.00      1.00      1.00      4970
           4       1.00      1.00      1.00      4957
           5       0.97      0.98      0.97      4873

    accuracy                           0.88     29448
   macro avg       0.88      0.88      0.88     29448
weighted avg       0.88      0.88      0.88     29448
```

Fig. 13. XGBoost performance metrics

Upon tuning, these metrics improved modestly to 88.14% for accuracy and 0.88 for the F1-score. These results affirm the effectiveness of gradient boosting techniques in handling

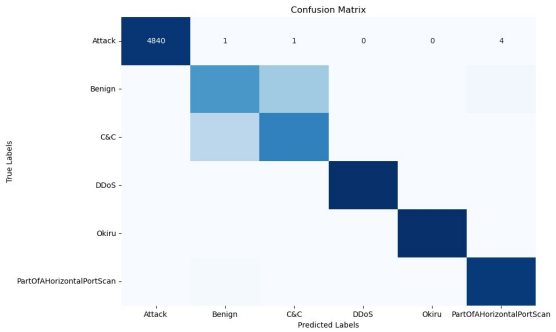

Fig. 14. XGBoost confusion matrix

complex datasets, though the increments post-tuning were not as substantial as those seen in the KNN model.

V. Experiments and Results

In our analysis, we employed several machine learning models—Decision Trees, Random Forest, K Nearest Neighbors (KNN), and XGBoost—to detect anomalies in IoT networks. The models were evaluated based on their accuracy and F1-scores across different classes of network traffic.

Model Accuracy Comparison can be seen using figure 15. The graph below illustrates the accuracy of each model. Here are the key observations:

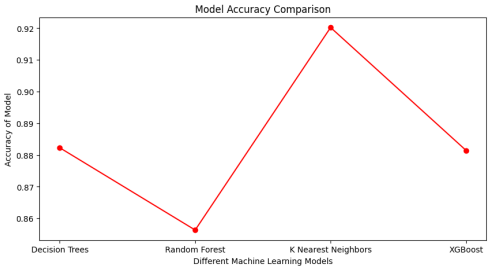

Fig. 15. XGBoost performance metrics

K Nearest Neighbors (KNN) achieved the highest accuracy, peaking at 92.02%, which highlights its effectiveness in classifying the data correctly.
Decision Trees showed a good start with an accuracy of 88.23%, while XGBoost also performed well, although slightly lower than KNN.
Random Forest had the lowest accuracy among the four, indicating that while robust, it might require further tuning to match the performance of the others in this specific dataset.

F1-Score Comparison by Model and Class The F1-score, which balances precision and recall, is crucial for evaluating model performance, especially in datasets with imbalanced classes. The bar chart in the figure 16 compares the F1-scores for each model across different classes:

KNN consistently shows high F1-scores across

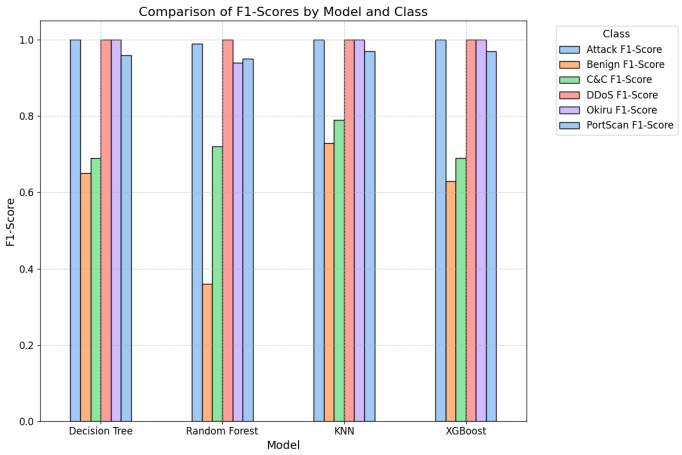

Fig. 16. XGBoost performance metrics

most classes, aligning with its superior accuracy results. Decision Trees and XGBoost display competitive F1-scores, but like their accuracy, are slightly overshadowed by KNN. Random Forest shows variability in its F1-scores among different classes, suggesting that some classes are better predicted than others.
KNN's high performance across both accuracy and F1-scores suggests it is very effective at handling the nuances of our dataset, likely due to its capability to form decision boundaries based on the distribution of all data points. Random Forest and XGBoost, while generally robust and powerful, may need adjustments in their hyperparameter settings to enhance their performance in this specific application. Decision Trees provide a good balance between simplicity and effectiveness, making them a valuable tool for initial assessments and scenarios where model interpretability is crucial. These results underline the importance of selecting the right model and tuning it appropriately for the specific characteristics of the data and the task at hand. Moving forward, continued refinement of model parameters and exploration of feature engineering could yield even better results, particularly for models that showed lower performance in this initial

analysis.

VI. Future Improvements

Looking forward, several strategies could be employed to enhance the performance of these models. First, more sophisticated feature engineering could be explored to extract and select features that capture more nuanced patterns in the data. Additionally, implementing deeper and more complex ensemble methods might further improve model robustness and accuracy. Experimentation with incremental learning techniques would also be advantageous, especially for applications requiring models to adapt to new data dynamically.

Moreover, efforts should be directed towards improving the scalability of these models to handle larger datasets more efficiently. This includes upgrading computing resources to accommodate the increasing volumes and speeds of data in industrial applications and refining data handling and processing techniques Finally, modeling capabilities interpretation is an important area, particularly in areas where understanding model decisions is critical for reliability and compliance

References

[1] I. Butun, P. Österberg, and H. Song, "Security of the internet of things: Vulnerabilities, attacks, and countermeasures," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 616–644, 2020.

[2] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer Networks*, vol. 76, pp. 146–164, 2015.

[3] W. Heyne, "A comprehensive study of anomaly detection schemes in iot networks using machine learning algorithms," *Sensors*, vol. 21, no. 24, p. 8320, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/24/8320

[4] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. Springer, 2013.

[5] X. Zhang and F. Wen, "Anomaly detection in iot: Recent advances, ai and ml perspectives and applications," IntechOpen.

[6] Electronics Editorial Office, "Iot botnet anomaly detection using unsupervised deep learning," *Electronics*, vol. 13, no. 11, p. 1825, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/13/11/1825

[7] Applied Sciences Editorial Office, "Anomaly detection for iot systems using active learning," *Applied Sciences*, vol. 13, p. 4350, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/21/4350

[8] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Wadsworth, 1984.

[9] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 785–794.