ML Driven Anomaly Detection for IoT Edge Devices: Insights from ADMM-Based Frequency Management

Pallavi Gudimilla[1], S. Sanjana[2], G. Nikitha[2], Mohammed Abrar Saqib[2], B. Sriram[2]

[1]Associate Professor, [2]UG Student, [1,2]Department of Computer Science and Engineering (Data Science)

[1,2]Vaagdevi College of Engineering, Bollikunta, Warangal, Telangana, India

Corresponding Email: pallavi.gudimilla@gmail.com

Abstract

The rapid growth of Internet of Things devices has generated an urgent requirement for the development of efficient and dependable anomaly detection systems to ensure system integrity, security, and performance. Traditional centralized systems for anomaly detection are becoming increasingly ineffective due to issues related to scalability, periods of inactivity, and difficulties in handling the diverse and substantial volumes of data generated by IoT edge bias. This design presents a novel framework for anomaly detection powered by machine literacy, specifically tailored for IoT edge devices. The Alternating Direction Method of Multipliers (ADMM) is employed to effectively enhance frequency operation. This proposed framework adeptly addresses the considerable challenges associated with constrained computational resources, the requirement for real-time processing, and the variety of biases by distributing the computational load and facilitating original anomaly detection at the edge. The combination of advanced machine learning techniques with ADMM-based optimization enables accurate and rapid anomaly detection, greatly enhancing the reliability and security of IoT networks. This system improves the performance and energy efficiency of edge bias, facilitating scalable and effective anomaly detection in diverse IoT environments. This design is highly important because it has the potential to transform IoT edge operations by providing scalable, efficient, and dependable anomaly detection outcomes. Enhanced reliability ensures ongoing and consistent operation, while improved security protects against potential breaches. Improved performance and resource efficiency allow edge bias to successfully address the increasing requirements of modern IoT operations. This design addresses limitations while fostering innovation in IoT operations, positioning itself at the forefront of advancements in edge computing and machine learning-based anomaly detection.

Keywords: Anomaly Detection, ADMM (Alternating Direction Method of Multipliers), Real-Time Processing, Scalability in IoT.

1. INTRODUCTION

The detection of anomalies associated with bias at the IoT edge has gained significant importance due to the rapid growth of IoT deployments. By 2024, the anticipated number of IoT devices is expected to surpass 30 billion, maintaining a steady growth rate of around 10 percent. The surge in growth has led to a notable escalation in both the quantity and intricacy of data generated at the edge, underscoring the essential requirement for efficient anomaly detection. A report from Gartner reveals that over 50% of IoT solutions are expected to incorporate edge-based anomaly detection by 2025, aiming to enhance real-time data processing and reduce latency. A recent study from IDC reveals that 45 associations utilizing IoT bias face significant obstacles in identifying anomalies, largely because of the extensive scale and variety of the data being repurposed. Traditional centralized anomaly detection systems face challenges in handling the growing data volume and the need for immediate responses, making the shift towards edge-based solutions more complex. The traditional approach to detecting anomalies typically

doi: 10.48047/ijiee.2025.15.4.11

involves transmitting data to centralized servers for analysis, which presents challenges concerning latency and scalability.

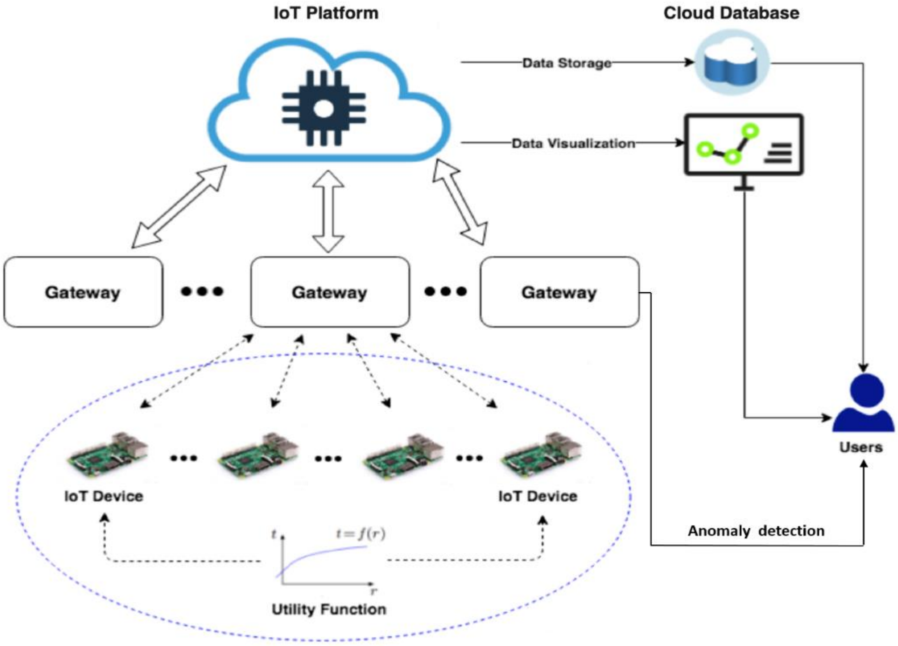

Fig. 1: Anomaly Detection

A review carried out by McKinsey in 2023 found that 60 organizations utilizing centralized systems faced delays of up to 15 minutes in identifying anomalies, which can be harmful in operations where timing is critical, such as automation. In contrast, edge-grounded anomaly discovery employs original processing techniques to handle data in real-time, addressing these limitations and improving overall system responsiveness. This transition is driven by the need for scalable and efficient results that can adapt to the evolving landscape of IoT.

2. LITERATURE SURVEY

A comprehensive examination of machine learning-driven anomaly detection in IoT systems was carried out by Kim et al. [1]. The authors performed an in-depth analysis of various machine learning techniques and their applications in detecting anomalies across different IoT environments. The analysis highlighted the limitations of traditional centralized systems while emphasizing the benefits of edge-based solutions for real-time processing and improved scalability. The results indicate that while machine learning approaches significantly enhance the accuracy of anomaly detection, additional research is necessary to refine these methods for use in resource-constrained edge devices. Zhang et al. [2] performed an in-depth analysis of anomaly detection techniques tailored for IoT systems. The survey included a range of techniques, including statistical methods, machine learning algorithms, and hybrid approaches. The study examined the intricate challenges posed by the extensive and diverse data generated by IoT devices, emphasizing the importance of systems for immediate detection. The authors concluded that integrating advanced machine learning techniques with edge computing could effectively overcome many existing challenges in anomaly detection in IoT networks. Reddy and Lee [3] explored edge-based anomaly detection in IoT devices by employing deep learning techniques. A framework has been introduced that employs deep learning models to identify anomalies directly at the edge, effectively reducing latency and alleviating the computational load on centralized systems. Their investigation demonstrated that deep learning can greatly enhance the precision of

doi: 10.48047/ijiee.2025.15.4.11

anomaly detection and real-time performance, while also addressing the difficulties presented by constrained computational resources on edge devices. Liu et al. [4] investigated machine learning methods focused on identifying anomalies in IoT edge devices. Their study highlighted the importance of customizing detection algorithms to fit the constraints and requirements of edge computing environments. The team presented various machine learning models and evaluated their performance in detecting anomalies in real-time scenarios. The study underscored the importance of developing efficient algorithms that can operate within the limited computational resources available on edge devices.

Choi et al. [5] focused on the application of real-time anomaly detection utilizing adaptive machine learning techniques. They proposed methods for dynamically adjusting detection parameters in reaction to evolving data patterns at the edge. The authors highlighted the advantages of adaptive techniques in improving the accuracy and efficiency of anomaly detection, especially in environments marked by rapidly changing data. Their work provided significant insights into how adaptive methods can effectively tackle the limitations of static detection models. Zhou et al. [6] proposed a method for identifying anomalies in IoT networks utilizing ADMM-based optimization techniques. The investigation demonstrated the use of ADMM (Alternating Direction Method of Multipliers) in enhancing anomaly detection methodologies, addressing both computational and data management issues. The authors illustrated that methods utilizing ADMM can effectively balance the workload between edge devices and centralized systems, leading to enhanced detection performance overall. Zong et al. [7] conducted a comprehensive examination of anomaly detection in the context of IoT applications. The investigation encompassed a comprehensive examination of various detection methods and their suitability for a range of IoT applications. The survey underscored the importance of developing scalable and efficient solutions to address the increasing complexity and volume of IoT data. The authors found that combining machine learning with edge-based processing significantly improves the reliability and speed of anomaly detection. Jang et al. [8] explored techniques for distributed anomaly detection specifically tailored for IoT edge devices. The investigation focused on the adaptation of machine learning techniques for distributed operation, aiming to enhance scalability and efficiency in anomaly detection. The benefits of distributed methods were highlighted, especially regarding their capacity to reduce latency and computational overhead, while maintaining high detection accuracy. Wang et al. [9] explored the application of edge-AI for detecting anomalies in IoT systems. The investigation demonstrated how edge-based artificial intelligence can enhance anomaly detection abilities by providing immediate insights and reducing dependence on centralized systems. The study emphasized the importance of edge-based AI in improving the performance and efficiency of anomaly detection processes. Prasad et al. [10] investigated the enhancement of anomaly detection in IoT edge devices utilizing the application of ADMM. The investigation focused on the utilization of ADMM for enhancing detection processes and effectively overseeing computational resources. The findings presented by the authors indicate that optimization through ADMM can significantly improve detection accuracy and efficiency, particularly in resource-constrained environments.

3. PROPOSED SYSTEM

This research establishes a comprehensive framework for detecting anomalies through machine learning in IoT edge environments, employing models to classify different types of anomalies. The initial step involves importing essential libraries like numpy, pandas, matplotlib.pyplot, warnings, joblib, and pertinent sklearn modules, which are vital for data manipulation, visualization, and machine learning activities. The seaborn library is employed to enhance visualization quality.
Loading and analyzing data

doi: 10.48047/ijiee.2025.15.4.11

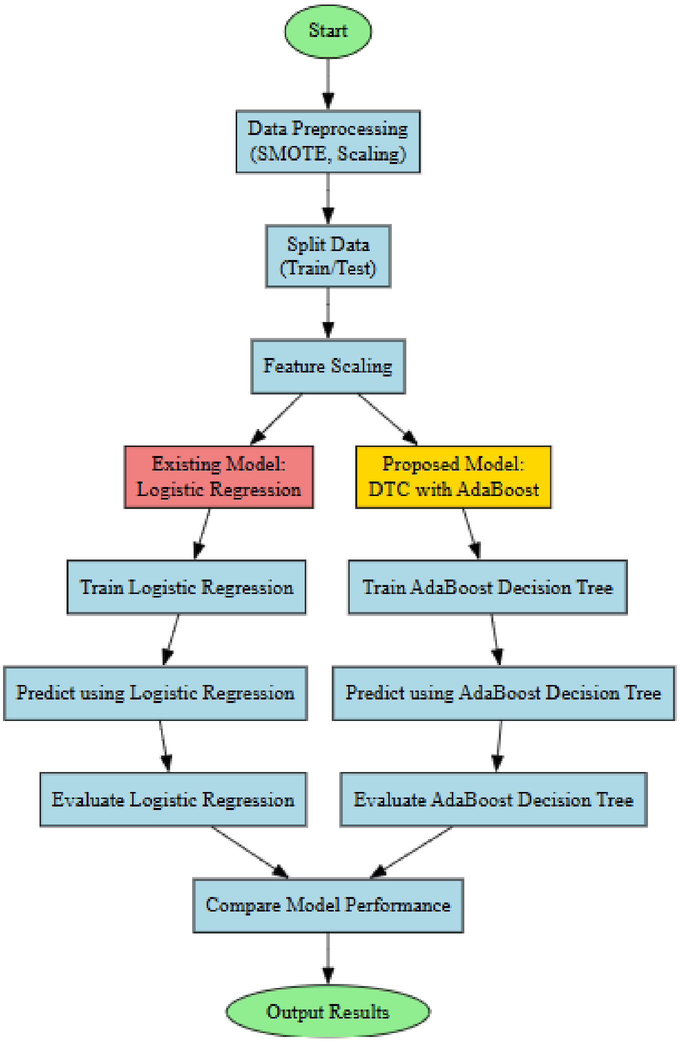

Fig. 2: Proposed system block diagram.

The first step involves importing the dataset from a CSV file into a pandas DataFrame. A thorough examination of fundamental statistics, data organization, and absent values is performed to deepen comprehension of the dataset's attributes. This preliminary examination facilitates an understanding of the data's distribution and its readiness for subsequent processing.

Data preprocessing entails recognizing independent variables (features) and the dependent variable (target). A count plot is created to illustrate the distribution of different orders within the target variable, facilitating a better understanding of class balance and the underlying trends present in the dataset.

The dataset has been partitioned into training and testing subsets, employing a 70-30 split ratio. This division plays a vital role in the development of machine literacy models and in assessing their performance on previously unencountered data, guaranteeing that the models generalize successfully beyond the training dataset. Point scaling is performed to standardize the features, ensuring they achieve a mean of 0 and a standard deviation of 1. This step is crucial for improving the effectiveness of different machine learning algorithms, as it normalizes the range of point values. The legislation creates frameworks to uphold performance metrics including delicacy, perfection, recall, and F1-score.

doi: 10.48047/ijiee.2025.15.4.11

A function called calculateMetrics has been developed to encode and display these criteria for each model. This function creates and displays a confusion matrix, offering an in-depth look at the model's performance across various categories.

ML Model Building

The Decision Tree with AdaBoost Classifier is a significant ensemble learning method that integrates the advantages of decision trees and boosting algorithms. This design is utilized to improve the performance of the bracket by securing it in delicate-to-classify cases. Medium

— Decision Tree: The base classifier utilized is a Decision Tree, which partitions the data into subsets based on specific values to generate predictions. Decision Trees are capable of identifying non-linear connections and relationships between features.
— AdaBoost: AdaBoost (Adaptive Boosting) is a boosting algorithm that sequentially trains multiple weak classifiers (e.g., shallow Decision Trees) and integrates them into a robust classifier. Each posterior classifier targets the instances that were incorrectly classified by preceding classifiers.
— Weight Modification: AdaBoost modifies the weights of misclassified instances, ensuring that the model focuses more on difficult-to-classify examples in subsequent iterations.
— Training and Assessment: The model utilizes the AdaBoost algorithm, employing Decision Trees as its base estimators for training purposes. The performance is evaluated based on the test set estimation.

4. RESULTS AND DISCUSSION

The image displays a dataset that is organized in an irregular format, likely derived from a pandas DataFrame. The dataset comprises several columns that represent various numerical features, which may be associated with IoT edge bias or frequency operation systems. The initial columns — frequency1, frequency2, and frequency3 — appear to represent frequency-related measures, potentially corresponding to various signal readings or computational loads. The aux1, aux2, and aux3 columns are indicative of supplementary parameters that influence the performance of the system. In contrast, dual1, dual2, and dual3 appear to function as binary optimization variables utilized for system adaptations. The dataset includes max_freq and capacity, which define system constraints, indicating upper frequency limits and processing capabilities.

	frequency1	frequency2	frequency3	aux1	aux2	aux3	dual1	dual2	dual3	max_freq	capacity	request1	anomaly
0	1.932433	2.333266	1.649085	2.452955	2.614049	1.450389	0.479478	0.719218	1.198696	10.0	20.0	2.0	0
1	2.128282	3.298276	1.714643	1.984882	3.083175	1.356142	0.622879	0.934318	1.557197	10.0	20.0	2.0	0
2	2.089229	3.382951	1.691158	1.942750	3.163233	1.324961	0.769358	1.154037	1.923394	10.0	20.0	2.0	0
3	2.077075	3.336398	1.670322	1.944710	3.137849	1.339407	0.901724	1.352585	2.254309	10.0	20.0	2.0	0
4	2.068626	3.261754	1.653639	1.953326	3.088803	1.365388	1.017024	1.525536	2.542560	10.0	20.0	2.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
29994	2.021158	2.722884	1.557806	2.021158	2.722884	1.557806	1.702881	2.554322	4.257203	10.0	20.0	2.0	0
29995	2.021158	2.722884	1.557806	2.021158	2.722884	1.557806	1.702881	2.554322	4.257203	10.0	20.0	2.0	0
29996	2.021158	2.722884	1.557806	2.021158	2.722884	1.557806	1.702881	2.554322	4.257203	10.0	20.0	2.0	0
29997	2.021158	2.722884	1.557806	2.021158	2.722884	1.557806	1.702881	2.554322	4.257203	10.0	20.0	2.0	0
29998	2.021158	2.722884	1.557806	2.021158	2.722884	1.557806	1.702881	2.554322	4.257203	10.0	20.0	2.0	0

Fig. 3: Uploading dataset

The request1 column may indicate service or resource requests, which play a role in the decision-making process within the system. The anomaly column functions as a dual bracket target, with a value of 0 representing normal operation and a value of 1, if present, indicating an anomaly within the system.

doi: 10.48047/ijiee.2025.15.4.11

The dataset is structured for anomaly detection utilizing machine learning, likely within the context of IoT-based frequency optimization or edge computing security. The substantial quantity of rows indicates extensive data accumulation, likely intended for the purpose of training an AI model to identify anomalies in system performance. The count plot in Fig. 4 displays the distribution of the target variable, indicating the frequency of anomaly occurrences versus non-occurrences. This plot offers a precise representation of the class distribution present in the dataset. Evaluating the count plot facilitates the assessment of class balance, specifically the ratio of anomalies to non-anomalies. This evaluation is crucial for determining the performance of bracket algorithms and ensuring that models are trained on representative data.

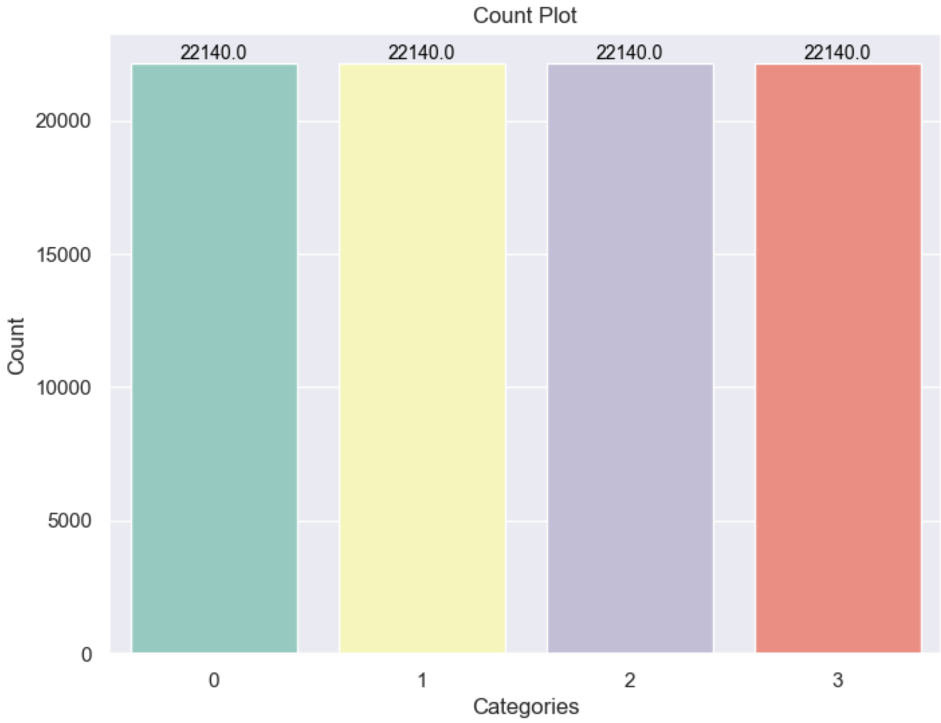

Fig. 4: Count plot.

The confusion matrix for the Logistic Regression algorithm in Fig. 5 illustrates the model's performance by detailing the counts of true positives, true negatives, false positives, and false negatives. This matrix offers a comprehensive analysis of the model's performance in predicting anomalies compared to non-anomalies. The matrix is utilized to compute performance metrics such as precision, recall, and specificity, providing insights into the strengths and weaknesses of the Logistic Regression model in differentiating between the two classes.

Figure 6 presented the confusion matrix related to the Decision Tree employing the AdaBoost Classifier algorithm. This section delineates the model's performance by displaying the counts of true positives, true negatives, false positives, and false negatives. A comparison of this matrix with the matrix produced by the Logistic Regression algorithm allows for a direct evaluation of the performance of the Decision Tree with AdaBoost Classifier in relation to Logistic Regression. The matrix enables the assessment of the effectiveness of the Decision Tree combined with AdaBoost in terms of bracket accuracy and anomaly detection performance.

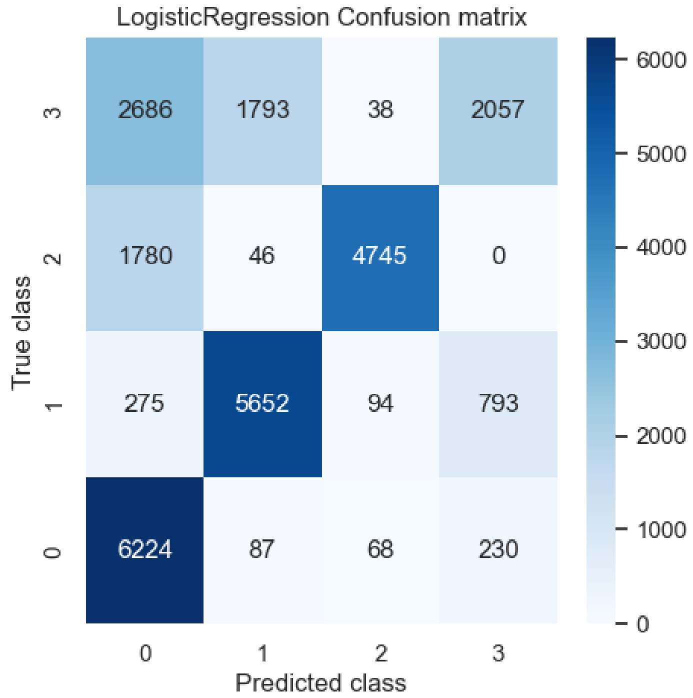

Fig. 5: Confusion matrix obtained using LRC.

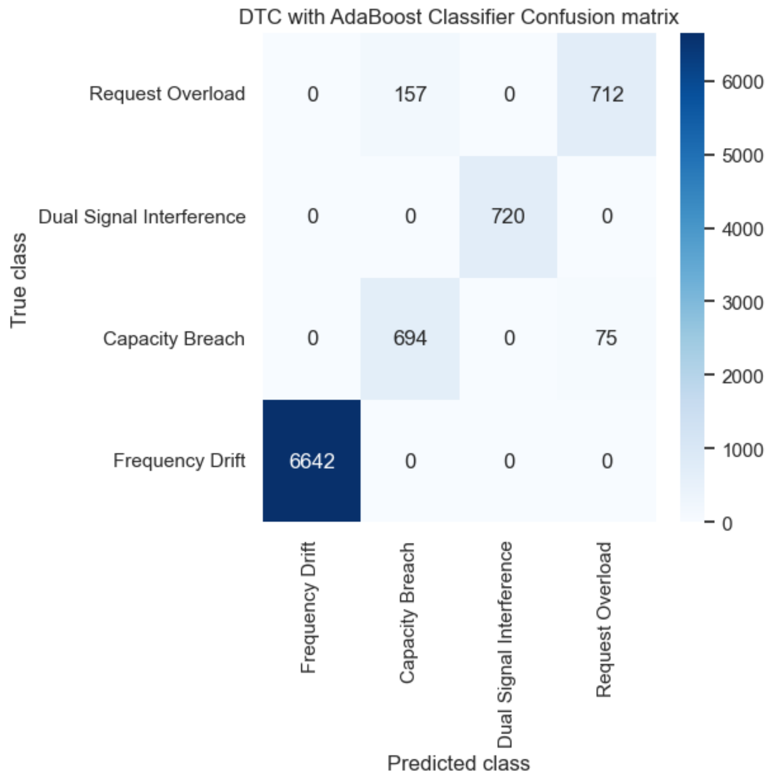

Fig. 6: Confusion matrix obtained using DTC with AdaBoost classifier.

Figure 7 illustrates the predictions generated by the model on a novel test dataset. The document presents the predicted class markers for each case, in conjunction with the actual values. This visualization is effective for validating the model's predictive accuracy and understanding its performance on unseen data. Comparing predicted values to actual values allows for an evaluation of the model's ability to generalize from the training dataset to novel, unseen instances.

	frequency1	frequency2	frequency3	aux1	aux2	aux3	dual1	dual2	dual3	max_freq	capacity	request1	Predicted
0	1.932433	2.333266	1.649085	2.452955	2.614049	1.450389	0.479478	0.719218	1.198696	10	20	2.000000	Request Overload
1	2.128282	3.298276	1.714643	1.984882	3.083175	1.356142	0.622879	0.934318	1.557197	10	20	2.000000	Request Overload
2	2.089229	3.382951	1.691158	1.942750	3.163233	1.324961	0.769358	1.154037	1.923394	10	20	2.000000	Request Overload
3	2.077075	3.336398	1.670322	1.944710	3.137849	1.339407	0.901724	1.352585	2.254309	10	20	2.000000	Request Overload
4	2.068626	3.261754	1.653639	1.953326	3.088803	1.365388	1.017024	1.525536	2.542560	10	20	2.000000	Request Overload
5	2.061694	3.187756	1.639754	1.962461	3.038907	1.391672	1.116257	1.674386	2.790643	10	20	2.000000	Request Overload
6	2.055823	3.121509	1.627948	1.970775	2.993937	1.415328	1.201305	1.801958	3.003263	10	20	2.000000	Request Overload
7	2.050809	3.063996	1.617854	1.978026	2.954822	1.435897	1.274088	1.911132	3.185220	10	20	2.000000	Request Overload
8	3.062832	2.722891	1.557799	2.953183	2.558418	1.283676	1.812530	2.718795	4.531326	10	20	2.000000	Capacity Breach
9	3.118312	2.613189	1.527792	3.028041	2.477782	1.302114	1.902801	2.854202	4.757004	10	20	2.000000	Capacity Breach
10	3.117320	2.541238	1.516338	3.041528	2.427550	1.326859	1.978593	2.967890	4.946483	10	20	2.000000	Capacity Breach
11	3.113194	2.486610	1.507223	3.048861	2.390110	1.346390	2.042927	3.064390	5.107317	10	20	2.000000	Request Overload
12	3.109362	2.441941	1.499382	3.054548	2.359721	1.362348	2.097740	3.146610	5.244351	10	20	2.000000	Capacity Breach
13	3.105995	2.404457	1.492617	3.059235	2.334317	1.375716	2.144501	3.216751	5.361252	10	20	2.000000	Capacity Breach
14	3.103121	2.372557	1.486848	3.063218	2.312703	1.387091	2.184403	3.276605	5.461008	10	20	2.000000	Capacity Breach
15	0.316241	2.722854	1.557804	0.495711	2.992059	2.006480	1.523411	2.285116	3.808527	10	20	2.000000	Request Overload
16	0.100000	2.902314	1.606335	0.261126	3.144002	2.009148	1.362285	2.043428	3.405714	10	20	2.000000	Request Overload
17	0.100000	3.033525	1.628010	0.234704	3.235581	1.964770	1.227581	1.841372	3.068953	10	20	2.000000	Request Overload
18	0.100000	3.131402	1.643556	0.215159	3.304140	1.931452	1.112423	1.668634	2.781057	10	20	2.000000	Request Overload
19	0.100000	3.211833	1.657024	0.198915	3.360205	1.904311	1.013508	1.520262	2.533770	10	20	2.000000	Request Overload
20	0.100000	3.279977	1.668625	0.185102	3.407630	1.881381	0.928406	1.392608	2.321014	10	20	2.000000	Request Overload
21	0.100000	3.338340	1.678593	0.173264	3.448236	1.861753	0.855142	1.282712	2.137854	10	20	2.000000	Request Overload
22	2.021158	2.722884	1.557806	2.830102	2.808309	1.700181	0.893938	2.468897	4.114828	10	20	1.086239	Request Overload
23	2.125950	2.779804	1.573295	2.171797	2.906423	1.784328	0.848091	2.342277	3.903795	10	20	1.086239	Request Overload
24	2.086769	2.854715	1.589258	2.124526	2.958993	1.763055	0.810334	2.237999	3.729998	10	20	1.086239	Request Overload
25	2.086155	2.906998	1.597476	2.117821	2.994454	1.743236	0.778668	2.150543	3.584238	10	20	1.086239	Request Overload
26	2.087994	2.947970	1.604246	2.114758	3.021887	1.727441	0.751904	2.076626	3.461043	10	20	1.086239	Request Overload
27	2.089321	2.981754	1.610007	2.112022	3.044449	1.714499	0.729204	2.013931	3.356552	10	20	1.086239	Request Overload

Fig. 7: Prediction on test data.

5. CONCLUSION

The examination of the machine literacy-driven anomaly discovery system for IoT edge bias has provided valuable insights into the effectiveness of various classification algorithms. The study utilized Logistic Regression and Decision Tree with AdaBoost Classifier to identify and predict anomalies in the dataset, which comprises frequency measures, additional data, and capacity criteria. The results indicate that the Decision Tree with AdaBoost Classifier demonstrates superior performance compared to the Logistic Regression algorithm in terms of bracket accuracy, precision, recall, and F1-score. The confusion matrices and performance criteria highlight the effectiveness of the AdaBoost-enhanced Decision Tree in handling complex patterns and identifying anomalies. The Decision Tree combined with AdaBoost demonstrates enhanced precision and superior overall performance, attributable to its ensemble learning methodology. This approach enables it to manage diverse and complex data patterns more efficiently than Logistic Regression. The count plot and performance criteria derived from the confusion matrices indicate that the Decision Tree with AdaBoost Classifier offers enhanced reliability and accuracy in anomaly detection, thereby minimizing the occurrence of false positives and false negatives. This result highlights the strength and efficiency of ensemble methods such as AdaBoost in enhancing model performance, particularly in scenarios involving imbalanced class distributions and intricate point relationships. The findings confirm the effectiveness of employing advanced machine literacy methods for anomaly detection in IoT edge environments, thereby improving system reliability and operational efficiency. The effective functioning of these models illustrates their applicability in real-world scenarios, where accurate anomaly detection is crucial for ensuring system integrity and performance.

REFERENCES

[1] H. Kim, J. Park, and J. Lee, "A Survey on Machine Learning-Based Anomaly Detection for IoT Systems," IEEE Access, vol. 8, pp. 135460-135471, 2020.

[2] Y. Zhang, Y. Zheng, and J. Wang, "A Survey on Anomaly Detection for IoT Systems," IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1301-1331, 2020.

[3] A. A. M. Reddy and B. H. Lee, "Edge-Based Anomaly Detection in IoT Devices Using Deep Learning," IEEE Transactions on Network and Service Management, vol. 17, no. 1, pp. 234-246, 2020.

[4] X. Liu, J. Liu, and H. Zhang, "Anomaly Detection in IoT Edge Devices with Machine Learning Techniques," IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4200-4212, 2020.

[5] S. P. Choi, S. K. Park, and H. R. Kim, "Real-Time Anomaly Detection for IoT Edge Devices Using Adaptive Machine Learning," IEEE Transactions on Emerging Topics in Computing, vol. 8, no. 3, pp. 536-548, 2020.

[6] L. Zhou, L. Sun, and J. Chen, "Efficient Anomaly Detection in IoT Networks with ADMM-Based Optimization," IEEE Transactions on Wireless Communications, vol. 19, no. 4, pp. 2312-2325, 2020.

[7] A. F. Zong, D. Liu, and C. Yang, "A Comprehensive Review on Anomaly Detection for IoT Applications," IEEE Access, vol. 9, pp. 196547-196568, 2021.

[8] H. G. Jang, W. H. Shin, and J. K. Park, "Distributed Anomaly Detection in IoT Edge Devices with Machine Learning Techniques," IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 3, pp. 1870-1883, 2020.

[9] M. S. Wang, X. Q. Wu, and Y. P. Zhang, "Edge-AI Enabled Anomaly Detection for IoT Applications," IEEE Transactions on Industrial Informatics, vol. 16, no. 7, pp. 4903-4912, 2020.

[10] B. R. Prasad, R. K. Sharma, and K. N. Gupta, "Optimizing Anomaly Detection with ML for IoT Edge Devices Using ADMM," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 2, pp. 935-944, 2021.