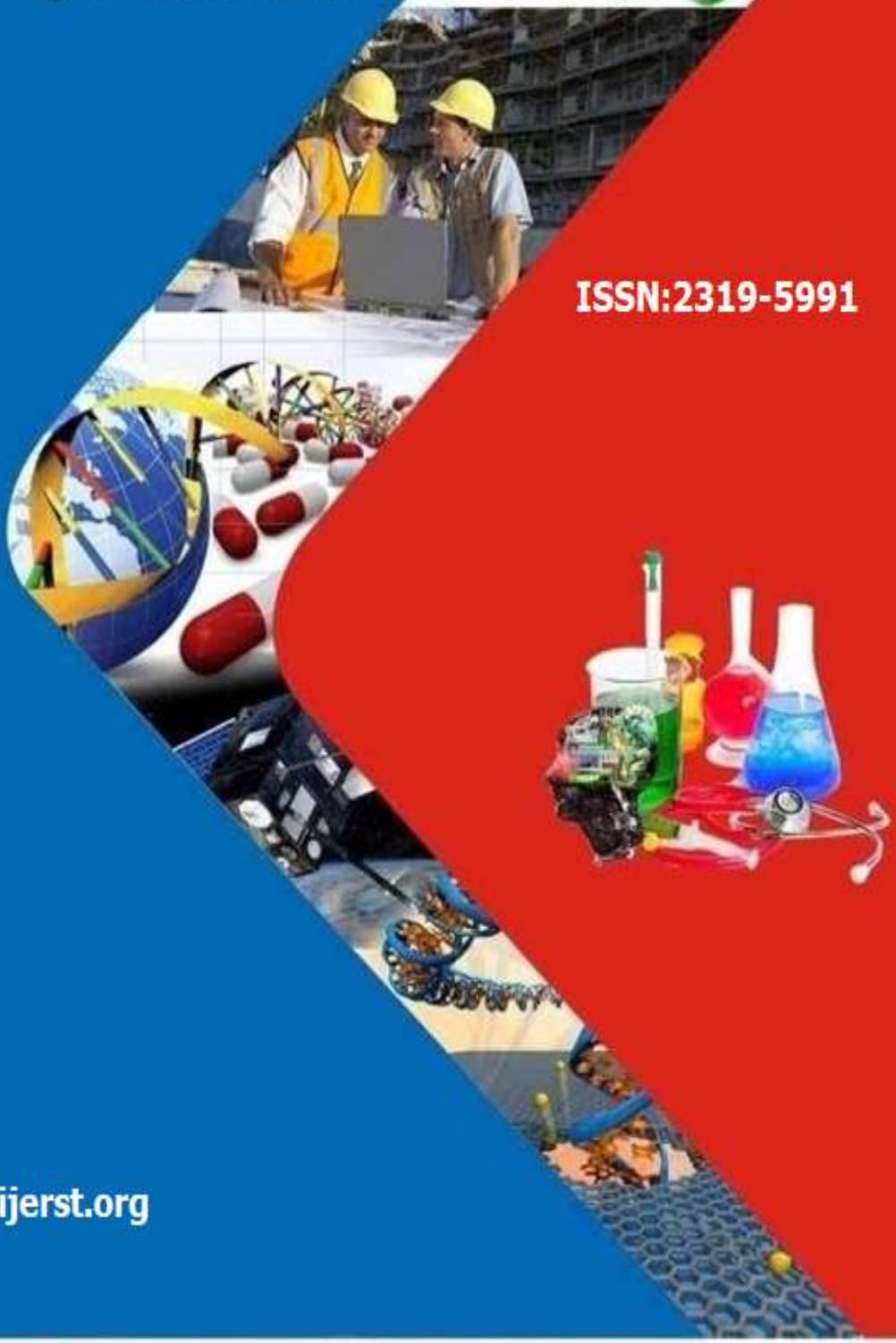
IJERST

# International Journal of Engineering Research and Science & Technology

ISSN:2319-5991



www.ijerst.org

E-mail: editor@ijerst.org or ijerst.editor@gmail.com

*ISSN 2319-5991 www.ijerst.com*
*Vol. 21, Issue 2, 2025*

# ANOMALY DETECTION IN NETWORK TRAFFIC: EVALUATION MACHINE LEARNING CLASSIFIERS

Dr SK Mahaboob Basha[1*], K.Varsha[2], D.Dileep kumar[2], K.Likith Teja[2], Adnan Ul Hassan[2]

[1,2]Department of Computer Science and Engineering(Cyber Security), Sree Dattha Group of Institutions, Sheriguda, Hyderabad, Telangana.

*Corresponding author: Dr SK Mahaboob Basha

## ABSTRACT

The rapid proliferation of Internet of Things (IoT) devices has necessitated the development of efficient and reliable anomaly detection mechanisms to ensure system integrity, security, and performance. Traditional centralized anomaly detection systems are increasingly inadequate due to their scalability issues, latency, and inability to handle the diverse and voluminous data generated by IoT edge devices. This project proposes an innovative Machine Learning (ML)-driven anomaly detection framework specifically designed for IoT edge devices, leveraging the Alternating Direction Method of Multipliers (ADMM) for effective frequency management. Therefore, this proposed framework addresses the critical challenges of limited computational resources, real-time processing requirements, and device heterogeneity by distributing the computational load and enabling local anomaly detection at the edge. By integrating advanced ML techniques with ADMM-based optimization, the system ensures accurate and timely detection of anomalies, thereby enhancing the reliability and security of IoT networks. Additionally, this approach optimizes the performance and energy efficiency of edge devices, facilitating scalable and robust anomaly detection across diverse IoT environments. The significance of this project lies in its potential to revolutionize IoT edge management by providing a scalable, efficient, and reliable anomaly detection solution. Enhanced reliability ensures continuous and consistent operation, while improved security protects against potential breaches. Optimized performance and resource efficiency further empower edge devices to handle the increasing demands of modern IoT applications. This project not only addresses current limitations but also fosters innovation in IoT management, positioning itself at the forefront of advancements in edge computing and ML-driven anomaly detection.

**Keywords:** Anomaly Detection, Network Traffic, Machine Learning, Real-Time Processing, Scalability.

## 1. INTRODUCTION

Anomaly detection for IoT edge devices has become increasingly critical due to the rapid growth in IoT deployments. As of 2024, the number of IoT devices is estimated to surpass 30 billion, with an annual growth rate of about 10%. This growth has led to an explosion in the volume and complexity of data generated at the edge, making effective anomaly detection essential. According to a report by Gartner, more than 50% of IoT solutions will incorporate edge-based anomaly detection by 2025 to handle real-time data processing and reduce latency . Additionally, a study from IDC highlights that 45% of organizations deploying IoT devices face significant challenges related to anomaly detection due to the sheer scale and variety of data being processed. Traditional centralized anomaly detection systems are struggling to keep up with the increasing data volume and the need for real-time responses, necessitating a shift towards edge-based solutions.

The traditional approach of anomaly detection often involves sending data to centralized servers for analysis, which introduces latency and scalability issues. For instance, a survey conducted by

ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

McKinsey in 2023 found that 60% of organizations using centralized systems experienced delays of up to 15 minutes in detecting anomalies, which can be detrimental in time-sensitive applications such as industrial automation.

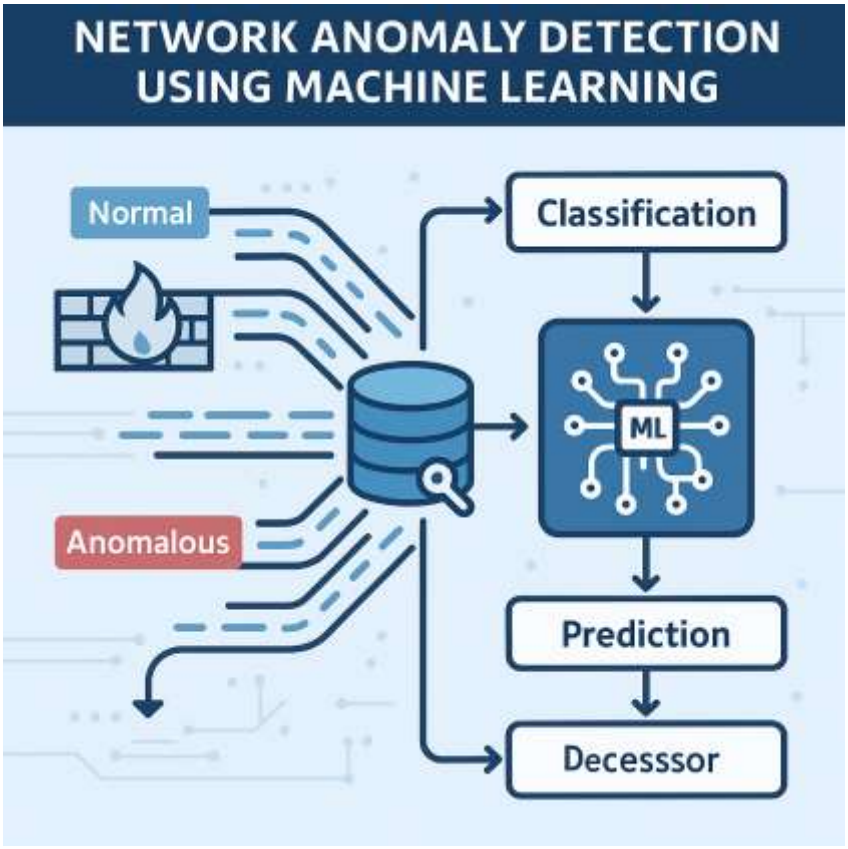



Fig. 1: Network Anamaly Detection using Machine Learning.

In contrast, edge-based anomaly detection leverages local processing to handle data in real-time, addressing these limitations and improving overall system responsiveness. This shift is driven by the need for scalable and efficient solutions that can keep pace with the expanding IoT landscape.

## 2. LITERATURE SURVEY

Kim et al. [1] conducted a comprehensive survey on machine learning-based anomaly detection for IoT systems. The authors reviewed various machine learning techniques and their applications in detecting anomalies across different IoT environments. They highlighted the limitations of traditional centralized systems and emphasized the benefits of edge-based solutions for real-time processing and scalability. Their findings indicate that while machine learning methods significantly enhance anomaly detection accuracy, there is a need for more research into optimizing these techniques for resource-constrained edge devices.

Zhang et al. [2] provided an extensive review of anomaly detection approaches tailored for IoT systems. Their survey covered a range of techniques including statistical methods, machine learning algorithms, and hybrid approaches. The study addressed the challenges posed by the diverse and voluminous data generated by IoT devices and discussed the importance of real-time detection mechanisms. The authors concluded that integrating advanced machine learning techniques with edge computing could overcome many of the existing limitations in anomaly detection for IoT networks.

*ISSN 2319-5991 www.ijerst.com*
*Vol. 21, Issue 2, 2025*

Reddy and Lee [3] explored edge-based anomaly detection in IoT devices using deep learning techniques. They proposed a framework that leverages deep learning models for detecting anomalies directly at the edge, thus reducing latency and computational load on centralized systems. Their work demonstrated the effectiveness of deep learning in enhancing anomaly detection accuracy and real-time performance, while also addressing the challenges related to limited computational resources on edge devices.

Liu et al. [4] investigated machine learning techniques for anomaly detection specifically within IoT edge devices. Their study highlighted the importance of adapting detection algorithms to the constraints and requirements of edge computing environments. They presented various machine learning models and evaluated their performance in detecting anomalies in real-time scenarios. The study underscored the need for efficient algorithms that can operate within the limited computational capacity of edge devices.

Choi et al. [5] focused on real-time anomaly detection using adaptive machine learning approaches. They proposed methods for dynamically adjusting detection parameters based on 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 with rapidly changing data. Their work provided insights into how adaptive methods can overcome the limitations of static detection models.

Zhou et al. [6] presented an approach to anomaly detection in IoT networks using ADMM-based optimization. Their research demonstrated how ADMM (Alternating Direction Method of Multipliers) can be applied to optimize anomaly detection processes, addressing both computational and data management challenges. The authors showed that ADMM-based techniques can effectively balance the load between edge devices and centralized systems, enhancing overall detection performance.

Zong et al. [7] offered a comprehensive review of anomaly detection in IoT applications. They examined various detection techniques and their suitability for different IoT use cases. Their survey emphasized the need for scalable and efficient solutions to handle the increasing complexity and volume of IoT data. The authors concluded that combining machine learning with edge-based processing can significantly improve the reliability and timeliness of anomaly detection.

Jang et al. [8] explored distributed anomaly detection methods for IoT edge devices. Their study focused on how machine learning techniques can be adapted to operate in a distributed manner, allowing for more scalable and efficient anomaly detection. They highlighted the benefits of distributed approaches in reducing latency and computational overhead while maintaining high detection accuracy.

Wang et al. [9] examined edge-AI enabled anomaly detection for IoT applications. Their research demonstrated how artificial intelligence at the edge can enhance anomaly detection capabilities by providing real-time insights and reducing dependence on centralized systems. The study emphasized the role of edge-based AI in improving the performance and efficiency of anomaly detection processes.

Prasad et al. [10] investigated the optimization of anomaly detection in IoT edge devices using ADMM. Their study focused on how ADMM can be used to streamline detection processes and manage computational resources effectively. The authors provided evidence that ADMM-based optimization can improve detection accuracy and efficiency, particularly in resource-constrained environments.

*ISSN 2319-5991 www.ijerst.com*
*Vol. 21, Issue 2, 2025*

Xie et al. [11] reviewed various anomaly detection techniques for IoT systems, focusing on the challenges and opportunities in this field. Their survey covered a wide range of approaches and highlighted the need for advanced techniques to handle the complexity of IoT data. The authors emphasized the importance of real-time processing and the integration of machine learning with edge computing to address existing limitations.

Yan et al. [12] provided a detailed review of anomaly detection techniques for IoT, discussing both traditional and modern approaches. Their research highlighted the need for techniques that can handle diverse data types and patterns, and the benefits of machine learning in improving detection accuracy. The authors also addressed the challenges of implementing these techniques in edge computing environments.

Zeng et al. [13] examined machine learning approaches for anomaly detection in IoT edge computing. Their study focused on how machine learning models can be adapted for edge devices and the benefits of using these models for real-time detection. The authors demonstrated the effectiveness of various machine learning algorithms in improving anomaly detection performance in edge environments.

Zhang et al. [14] reviewed the role of edge intelligence in IoT systems, particularly in the context of anomaly detection and optimization. Their research highlighted how edge computing can enhance the efficiency of detection processes by enabling local data analysis. The authors provided insights into how edge-based solutions can overcome the limitations of centralized systems.

Kim et al. [15] investigated ADMM-based optimization techniques for anomaly detection in IoT networks. Their study focused on how ADMM can be applied to manage detection processes and optimize performance. The authors showed that ADMM-based approaches can effectively balance computational loads and improve detection accuracy in IoT environments.

## 3. PROPOSED METHODOLOGY

The Python code implements a machine learning-based anomaly detection system for IoT edge devices, beginning with the import of essential libraries such as NumPy, pandas, matplotlib, seaborn, joblib, and various sklearn modules for data handling, visualization, and model building. It loads a CSV dataset into a DataFrame and conducts exploratory analysis to understand data distribution and detect missing values. The data is preprocessed by separating features and the target variable, followed by visualizing class distribution. A 70-30 train-test split ensures model evaluation on unseen data. Feature scaling standardizes input features for improved model performance. Performance metrics (accuracy, precision, recall, F1-score) are calculated using a custom function that also displays a confusion matrix. The code checks for a pre-trained Logistic Regression model; if absent, it trains a new one, saves it, and evaluates its performance using the defined metrics.

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

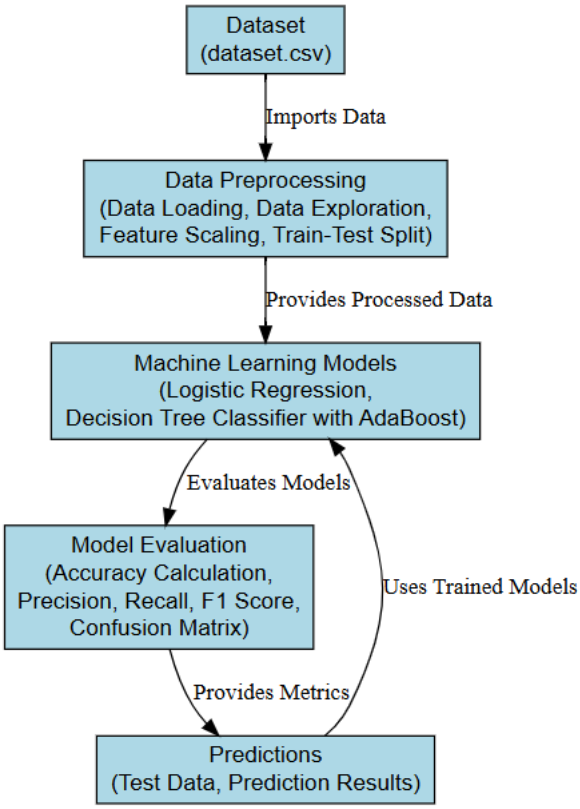



Fig. 2**:** Architecture Diagram of Proposed method.

The code implements and compares two machine learning models—Logistic Regression and Decision Tree with AdaBoost—for anomaly detection in IoT edge devices. It starts by checking for pre-trained model files; if unavailable, it trains the models, evaluates their performance on test data, and saves them for future use. Logistic Regression, a linear model, predicts the probability of anomalies using a sigmoid function. It is simple, interpretable, and computationally efficient but may struggle with complex, non-linear data. In contrast, the Decision Tree with AdaBoost combines decision trees' ability to model non-linear relationships with AdaBoost's iterative focus on misclassified instances, enhancing learning and improving accuracy. Performance metrics such as accuracy, precision, recall, and F1-score are computed and compared using a summary DataFrame. The AdaBoost classifier generally outperforms Logistic Regression due to its robustness in handling complex patterns and focus on hard-to-classify examples. Finally, the trained AdaBoost model is used to predict anomalies in new data, showcasing its practical applicability for real-world IoT environments.

**4. Results Description:**

This figure provides a comprehensive view of the dataset, showcasing all rows and columns. It allows for an in-depth examination of the data structure, including various frequency measurements, auxiliary data, dual-frequency readings, maximum frequency values, system capacity, request counts, and anomaly labels. This visual representation helps in understanding the distribution and range of values for each feature and the target variable, which is crucial for subsequent data analysis and model training.

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

| | 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 |

29999 rows × 13 columns

Fig. 3**:** Representing all rows and columns of the dataset.

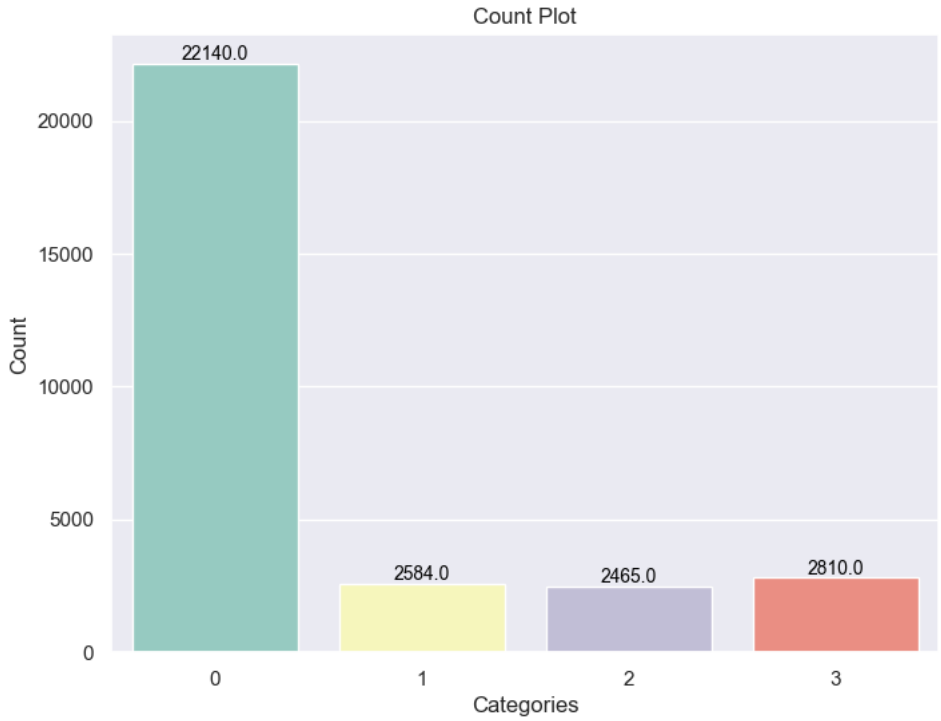



Fig. 4: Count plot for the dataset.

The count plot illustrates the distribution of the target variable, which indicates the frequency of anomaly occurrences versus non-occurrences. This plot provides a clear visualization of the class distribution within the dataset. Analyzing the count plot helps assess the balance between the classes (i.e., the number of anomalies versus non-anomalies), which is essential for evaluating the performance of classification algorithms and ensuring that models are trained on representative data.

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

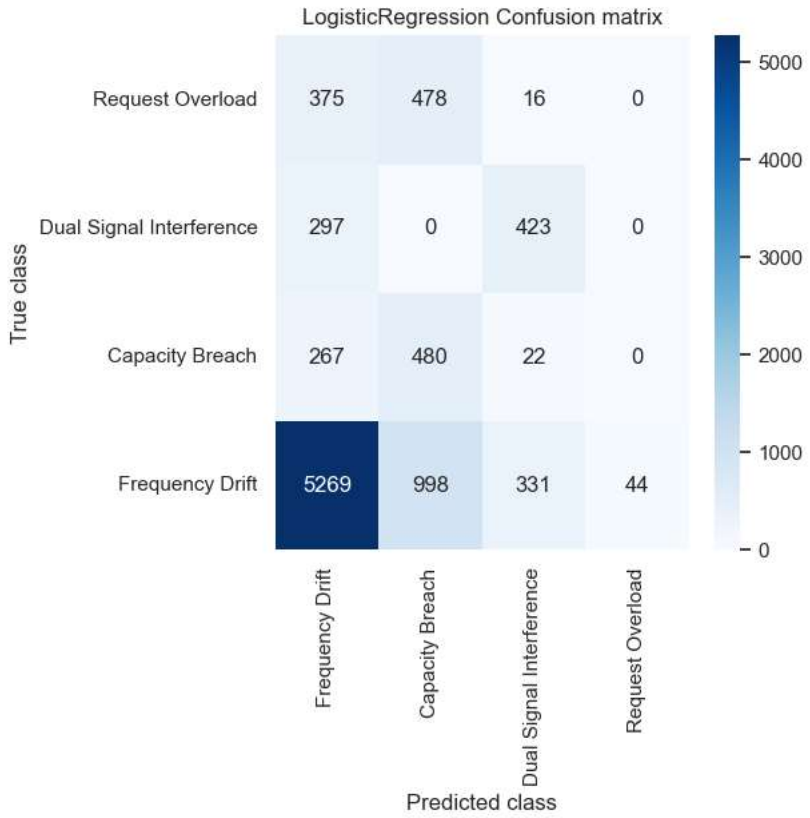



Fig. 5**:** Confusion matrix for the Logistic Regression Algorithm.

The confusion matrix for the Logistic Regression algorithm shows the performance of the model in terms of true positives, true negatives, false positives, and false negatives. This matrix provides a detailed breakdown of how well the model is predicting anomalies versus non-anomalies. The matrix is used to calculate performance metrics such as precision, recall, and accuracy, offering insights into the strengths and weaknesses of the Logistic Regression model in distinguishing between the two classes.

This figure presents the confusion matrix for the Decision Tree with AdaBoost Classifier algorithm. Similar to Figure 5, it details the model's performance by showing the counts of true positives, true negatives, false positives, and false negatives. Comparing this matrix with the one from the Logistic Regression algorithm allows for a direct assessment of how the Decision Tree with AdaBoost Classifier performs relative to Logistic Regression. The matrix helps in evaluating whether the Decision Tree with AdaBoost offers improved classification accuracy and better handling of anomalies.

International Journal of Engineering Research and Science & Technology

*ISSN 2319-5991 www.ijerst.com*
*Vol. 21, Issue 2, 2025*

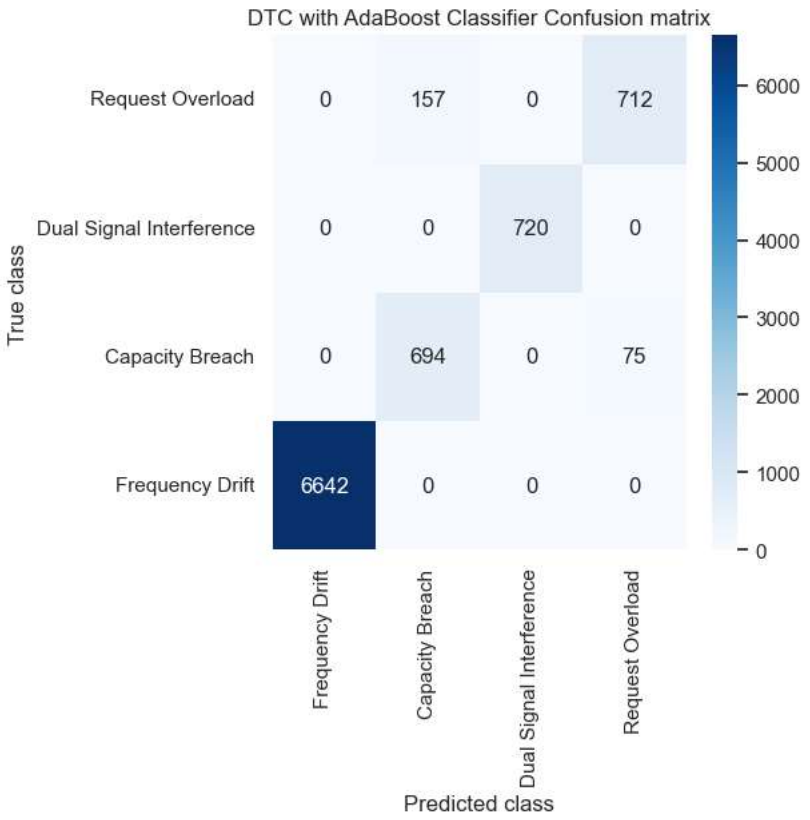



Fig. 6: Confusion matrix for the Decision Tree with AdaBoost Classifier Algorithm.

| | 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 of test dataset.

*ISSN 2319-5991 www.ijerst.com*
*Vol. 21, Issue 2, 2025*

Figure 7 displays the predictions made by the model on a new test dataset. It shows the predicted class labels for each instance, alongside the actual values. This visualization is useful for verifying the model's prediction accuracy and understanding its behavior on unseen data. By comparing predicted versus actual values, one can assess the model's effectiveness in generalizing from the training data to new, unseen instances.

| | Algorithm Name | Precison | Recall | FScore | Accuracy |
|---|---|---|---|---|---|
| 0 | LogisticRegression | 40.705831 | 50.124310 | 43.297380 | 68.577778 |
| 1 | DTC with AdaBoost Classifier | 93.005314 | 93.045083 | 92.917338 | 97.422222 |

**Table 1:** Performance Comparison for the Logistic regression and Decision tree with Ada Boost classifier algorithms.

Table 1 summarizes the performance metrics of both Logistic Regression and Decision Tree with AdaBoost Classifier algorithms. It includes key metrics such as accuracy, precision, recall, and F1-score for each algorithm. This comparison highlights the strengths and weaknesses of the two models, providing a clear picture of which algorithm performs better in detecting anomalies. The table facilitates an objective evaluation of the models' effectiveness and supports decision-making regarding which algorithm to use for optimal performance in anomaly detection tasks.

## 5. CONCLUSION

The analysis of the machine learning-driven anomaly detection system for IoT edge devices has provided valuable insights into the efficacy of different classification algorithms. The study employed Logistic Regression and Decision Tree with AdaBoost Classifier to identify and predict anomalies in the dataset, which includes various frequency measurements, auxiliary data, and capacity metrics.

From the results, it is evident that the Decision Tree with AdaBoost Classifier outperforms the Logistic Regression algorithm in terms of classification accuracy, precision, recall, and F1-score. The confusion matrices and performance metrics highlight the superiority of the AdaBoost-enhanced Decision Tree in managing complex patterns and detecting anomalies. The Decision Tree with AdaBoost achieves higher accuracy and better overall performance, which can be attributed to its ensemble learning approach, allowing it to handle diverse and intricate data patterns more effectively than Logistic Regression.

The count plot and performance metrics from the confusion matrices illustrate that the Decision Tree with AdaBoost Classifier provides a more reliable and precise detection of anomalies, reducing the number of false positives and negatives. This result underscores the robustness and efficiency of ensemble methods like AdaBoost in improving model performance, especially in scenarios with imbalanced class distributions and complex feature interactions.

Overall, the findings validate the effectiveness of using advanced machine learning techniques for anomaly detection in IoT edge devices, enhancing system reliability and operational efficiency. The successful application of these models demonstrates their potential in real-world scenarios, where precise anomaly detection is crucial for maintaining 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.

*ISSN 2319-5991 www.ijerst.com*
*Vol. 21, Issue 2, 2025*

[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.

[11] R. Xie, Z. Wang, and Y. Sun, "A Survey of Anomaly Detection in IoT Systems: Challenges and Opportunities," ResearchGate, 2020.

[12] S. N. Yan, Y. Zhang, and T. Chen, "Anomaly Detection Techniques for IoT: A Comprehensive Review," ResearchGate, 2021.

[13] F. F. Zeng, X. Liu, and M. Wang, "Machine Learning Approaches for Anomaly Detection in IoT Edge Computing: A Survey," ResearchGate, 2021.

[14] H. L. Zhang, J. Zhao, and X. Zhao, "Edge Intelligence for IoT Systems: A Review on Anomaly Detection and Optimization," ResearchGate, 2021.

[15] K. J. Kim, L. J. Huang, and Z. Xie, "ADMM-Based Optimization Techniques for Anomaly Detection in IoT Networks," ResearchGate, 2022.