# Deploying Isolation Forest at the Edge: A Synthetic Data-Driven Approach for Real-Time IoT Anomaly Detection

JUDAH IDOWU
*Department of Computer Science, National Open University of Nigeria*

***Abstract-*** ***This research addresses the challenge of detecting network anomalies in IoT environments by developing and evaluating AI-driven methodologies, specifically deploying Isolation Forest at the edge. A key obstacle in this domain is the scarcity of labeled datasets representing diverse anomaly types. To overcome this, we employed a synthetic data-driven approach, using statistical modeling to generate datasets that mimic real-world IoT scenarios, encompassing a wide range of normal and anomalous behaviors. This synthetic data enabled robust training and evaluation of our anomaly detection model. Performance was assessed using precision, recall, F1-score, and AUC-ROC. While the model demonstrated high preci-sion for normal data points, its performance in identifying anomalous data points was limited, as reflected in the low recall, F1-score, and ROC-AUC values. Despite these limitations, the study demonstrates the potential of Isolation Forest for detecting simpler anomalies like outliers and drifts, contributing to reduced false positives. Furthermore, the research emulated edge computing principles—distributed processing and reduced latency—to facilitate real-time analysis, which is crucial for mitigating security breaches and system failures in dynamic IoT environments. Although the simulation did not involve physical edge devices, it successfully demonstrated the feasibility of deploying Isolation Forest at the edge for real-time anomaly detection, highlighting its potential for enhancing operational efficiency, risk management, and security in sectors like healthcare, where early anomaly detection is critical.***

***Indexed Terms-*** ***Artificial Intelligence, Edge Computing, IoT Security, Machine Learning, Statistical Methods***

## I. INTRODUCTION

The proliferation of Internet of Things (IoT) devices has transformed how data is generated, transmitted, and analyzed across various sectors, including healthcare, finance, and cybersecurity. This exponential data volume and complexity growth has underscored the necessity for advanced anomaly detection mechanisms to identify and mitigate potential security threats, system failures, and data irregularities. Traditional anomaly detection methods, reliant on manual analysis and predefined rules, have proven inadequate in the face of dynamic and heterogeneous IoT environments. Integrating Artificial Intelligence (AI) and machine learning algorithms has emerged as a pivotal solution, enabling the real-time analysis of vast datasets to uncover hidden patterns and anomalies.

The research problem at the heart of this dissertation revolves around developing and evaluating innovative AI-driven methodologies for detecting network anomalies in IoT environments. This problem is compounded by the scarcity of publicly available datasets, including labeled normal and anomalous data from diverse IoT applications, communication protocols, and resource usage scenarios. The complexity of identifying and categorizing various types of anomalies further exacerbates this challenge. The primary objectives of this research are to design, implement, and evaluate AI-powered anomaly detection systems that can accurately identify unusual patterns in IoT network traffic. This involves the creation of comprehensive datasets, the selection and optimization of machine learning algorithms, particularly unsupervised learning techniques, and the assessment of their efficacy.

The proposed research will employ a comprehensive methodology involving data generation, model selection, and performance evaluation. Synthetic datasets will be generated using statistical modeling techniques to address the scarcity of publicly available IoT datasets. These datasets will capture the diversity of IoT applications, communication protocols, and

resource usage scenarios. A combination of normal and anomalous data will be included to train and validate the models.

Given the unsupervised nature of anomaly detection, Isolation Forest will be the primary algorithm employed. This technique is well-suited for identifying anomalies in high-dimensional datasets and is particularly effective for detecting subtle deviations from normal behavior. The model will be trained on the generated dataset and its performance will be evaluated on a separate test set.

To enhance real-time anomaly detection, edge computing will be integrated into the proposed system. By offloading computation to edge devices, the aim is to reduce latency and improve the overall responsiveness of the system. This will enable timely detection and response to potential threats, safeguarding the integrity of IoT networks.

The performance of the anomaly detection models will be assessed using a combination of metrics, including accuracy, precision, recall, and F1-score. These metrics will provide insights into the model's ability to correctly identify normal and anomalous data points. Additionally, the ROC-AUC curve will be used to evaluate the overall performance of the models.

The significance of this research lies in its potential to enhance operational efficiency, risk management, and security across multiple sectors. By providing proactive and adaptive solutions to emerging cyber threats and data anomalies, AI-powered anomaly detection can prevent system failures, detect fraudulent activities, and monitor critical data for signs of deterioration. Academically, this study contributes to the body of knowledge on AI and machine learning in IoT security, highlighting the challenges and innovations in this field. Practically, it offers valuable insights and methodologies that organizations can implement to strengthen their security posture and improve overall system reliability.

## II. LITERATURE REVIEW

The significance of harnessing AI for network anomaly detection in IoT settings cannot be overstated. With millions of devices generating vast data streams, traditional anomaly detection methods, often rule-based or heuristic, struggle to adapt to network behaviors' dynamic and evolving profiles. AI, particularly through machine learning and deep learning frameworks, provides the flexibility to learn from and adapt to new data patterns autonomously, thus offering improved detection rates and reduced instances of false positives [2,9,11]. The urgency of developing such intelligent systems is accentuated by the accelerating number of IoT devices, magnifying the attack surface available to malicious entities.

Previous research has examined various AI techniques for network anomaly detection, including supervised and unsupervised learning models, reinforcement learning, and hybrid approaches that integrate multiple methodologies. Studies have demonstrated the effectiveness of techniques like deep neural networks and support vector machines in improving detection capabilities while also highlighting variations in performance based on the nature of the datasets used [1,6].

Artificial Intelligence (AI) integration into network anomaly detection for Internet of Things (IoT) environments has evolved significantly over the past decade. Initially, traditional methods predominated, relying on rule-based systems. Their reliance on predefined rules or static thresholds makes them vulnerable to novel attacks and unable to capture subtle deviations in complex data streams. This limitation is particularly pronounced in IoT environments characterized by diverse device types, communication protocols, and rapidly evolving attack vectors [10]. These approaches could not identify novel threats due to their rigid frameworks, leading researchers to explore machine learning (ML) techniques as a more adaptive alternative [13].

The specific application of AI in IoT security has garnered significant attention in recent years. For instance, researchers have employed deep learning models to detect Distributed Denial-of-Service (DDoS) attacks targeting IoT devices [3]. Other studies have focused on using unsupervised learning techniques to identify anomalous behavior in sensor data from IoT devices [12]. These studies highlight the potential of AI to address the unique security challenges posed by IoT environments, such as

resource constraints, heterogeneity, and scalability.

The implementation of AI-powered anomaly detection systems has significant implications for operational efficiency and risk management. Proactive detection of anomalies can impact operational efficiency, cost reduction, and asset longevity. Early detection of fraudulent activities can mitigate financial losses and reputational damage. Continuous monitoring of critical data can provide valuable insights into system performance and identify potential risks before they escalate. By automating anomaly detection and response, organizations can improve their overall security posture and reduce the burden on security personnel.

One of the critical challenges in deploying AI for anomaly detection is the scarcity of labeled datasets, which is essential for training supervised learning models. In response, several studies advocate for adopting unsupervised learning approaches, which can effectively identify deviations from normal behavior without extensive prior knowledge of attack signatures [4]. This is particularly relevant in environments subjected to novel threats, where traditional signature-based methods struggle to adapt.

Additionally, the interplay of AI with edge computing is a growing area of focus. Edge-based AI facilitates real-time analysis and decision-making at the data source, mitigating latency issues and enhancing overall response times to potential threats [14]. Despite these advancements, issues remain, such as the need for robust defense mechanisms against adversarial attacks that can exploit deep learning models [8].

While existing research has demonstrated the potential of AI in IoT security, several challenges remain. These include the need for robust data collection and preprocessing techniques, scarcity of publicly available datasets, the development of efficient and scalable algorithms for resource-constrained IoT devices, and the need for explainable AI models that can provide insights into the reasons behind detected anomalies. This research aims to address some of these challenges by using an unsupervised learning technique, Isolation Forest, to detect anomalies in an IoT environment. This study contributes to the body of knowledge by generating synthetic data, developing a lightweight anomaly detection model suitable for resource-constrained devices, and using Edge computing for real-time detection. The practical implications of this research include providing organizations with a practical methodology for generating datasets and implementing AI-powered anomaly detection in their IoT deployments.

## III. METHODOLOGY

The development and evaluation of AI-driven methodologies for detecting network anomalies in IoT environments involve a multi-faceted approach that integrates data collection, preprocessing, model selection, and performance assessment. To address the scarcity of publicly available datasets, this research will involve the creation of comprehensive datasets that capture the diversity of IoT applications, communication protocols, and heterogeneous resource usage scenarios. These datasets will be labeled to include normal and anomalous data, facilitating the training and validation of machine learning models.

The unsupervised learning technique, Isolation Forest, will be prioritized due to its ability to identify anomalies without prior knowledge of attack signatures. This model will be selected based on its performance in detecting subtle and evolving anomalies, which are common in IoT environments. Additionally, the integration of edge computing will be explored to enable real-time analysis and decision-making at the data source, thereby mitigating latency issues and enhancing response times to potential threats.

The efficacy of these models will be assessed through a combination of metrics, including accuracy, precision, recall, and F1-score, to ensure their robustness and reliability in real-world IoT environments.

### *A. Data Generation and Collection*

*Synthetic Data Generation*. I employed a statistical modeling approach to generate synthetic data that mimics real-world IoT scenarios to address the scarcity of publicly available IoT datasets. This approach involves identifying key parameters of IoT devices and networks, such as sensor types, sampling

rates, network latency, and packet loss rates.

```
import numpy as np
import pandas as pd

# Generate temperature sensor data
temperature_data = np.random.normal(loc=25, scale=2, size=1000)

# Generate humidity sensor data
humidity_data = np.random.uniform(low=30, high=70, size=1000)

# Combine data into a DataFrame
data = pd.DataFrame({'Temperature': temperature_data, 'Humidity': humidity_data})
```

Fig.1 Python Code for Synthetic Data Generation

Code Explanation:

- Import Necessary Libraries: Imports numpy for numerical operations and pandas for data manipulation.
- Generate Data: Creates two NumPy arrays
- temperature_data: Generates 1000 random numbers from a normal distribution with a mean of 25 and a standard deviation of 2.
- humidity_data: Generates 1000 random numbers uniformly distributed between 30 and 70.
- Create DataFrame: Combines the two arrays into a Pandas DataFrame for easier manipulation.

*Anomaly Injection.* To introduce anomalies into the synthetic data, I applied various techniques.

```
outlier_indices = np.random.choice(len(data), size=100, replace=False)
data.loc[outlier_indices, 'Temperature'] += 20
```

Fig.2 Python Code for Outlier Injection

Code Explanation: Randomly selects 100 indices from the DataFrame and adds 20 to the Temperature values at those indices, creating outliers.

```
drift_indices = np.arange(500, 1000)
data.loc[drift_indices, 'Temperature'] += 5
```

Fig.3 Python Code for Drift Injection

Code Explanation: Increases the Temperature values for the last 500 data points by 5, simulating a gradual shift in the mean.

```
time_steps = np.arange(1000)
seasonal_pattern = np.sin(2 * np.pi * time_steps / 100)
data['Temperature'] += seasonal_pattern
```

Fig.4 Python Code for Seasonal Patterns

Code Explanation: Adds a sinusoidal pattern to the Temperature values, simulating seasonal variations.

*B. Edge Computing Simulation*

I simulated edge computing environments using software-based tools. This allowed an evaluation of the potential benefits of edge computing in real-time anomaly detection.

*Simulation Environment.* I used Python's multiprocessing library to simulate multiple edge nodes processing data in parallel. Each edge node was assigned a subset of the generated data and applied anomaly detection algorithms locally.

```
import multiprocessing

def process_data(data_chunk):
  # Apply anomaly detection algorithms to the data chunk
  # ...

if __name__ == '__main__':
  num_processes = 4
  pool = multiprocessing.Pool(num_processes)
  results = pool.map(process_data, np.array_split(data, num_processes))
```

Fig.5 Python Code to Simulate Multiple Edge Nodes

*C. Anomaly Detection Algorithm*

I implemented Isolation Forest, a Machine Learning anomaly detection algorithm, to identify anomalous patterns in the generated data.

```
from sklearn.ensemble import IsolationForest

# Train an Isolation Forest model
clf = IsolationForest(contamination='auto')
clf.fit(data)

# Predict anomalies
y_pred = clf.predict(data)
```

Fig.6 Python Code to Train an Isolation Forest Model

Code Explanation:

- Import Isolation Forest: Imports the Isolation Forest class from the sklearn.ensemble module.
- Train the Model: Creates an Isolation Forest model and fits it to the data. The contamination parameter automatically estimates the proportion of outliers.

- Predict Anomalies: Uses the trained model to predict anomaly scores for each data point. Negative scores indicate anomalies.

*D. Performance Evaluation Metrics*

Key metrics such as precision, recall, and F1-score are commonly used to evaluate the detection accuracy of anomaly detection systems. However, given the dynamic nature of IoT data, additional metrics like area under the receiver operating characteristic curve (AUC-ROC) can provide a more comprehensive understanding of the model's performance under varying conditions.

To evaluate the performance of our anomaly detection models, I used the following metrics:

Precision: The proportion of true positive predictions among all positive predictions.

Recall: The proportion of true positive predictions among all actual positive instances.

F1-score: The harmonic mean of precision and recall.
ROC-AUC curve: A plot of the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

```
from sklearn.metrics import classification_report, roc_auc_score

# Assuming y_true is the ground truth labels and y_pred are the predicted labels
print(classification_report(y_true, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_true, y_pred))
```

Fig.7 Python Code to Print Detailed Classification Report

Code Explanation:

- Import Metrics: Imports necessary metrics from sklearn.metrics.
- Classification Report: Prints a detailed classification report, including precision, recall, F1-score, and support for each class.
- ROC-AUC Score: Calculates and prints the area under the receiver operating characteristic curve, which measures the model's overall performance.

## IV. RESULTS AND DISCUSSIONS

The results of this research underscore the significance of more available datasets in leveraging Artificial Intelligence (AI) for network anomaly detection in Internet of Things (IoT) environments. The efficacy of machine learning algorithms, particularly unsupervised learning techniques, has been prominently highlighted as a good base model. These algorithms, such as Isolation Forest, have demonstrated potential capability in identifying anomalies without prior knowledge of attack signatures. This is crucial in IoT environments where novel threats are frequently emerging and labeled data is scarce.

The integration of edge computing has also proven to be a key factor in enhancing the real-time analysis and decision-making capabilities of these systems. By processing data at the edge, latency is significantly reduced, and response times to potential threats are improved, making the system more efficient and effective in mitigating security breaches.

The performance evaluation metrics, including precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC), have provided a comprehensive understanding of the model's performance under varying conditions.

*A. Presentation of Data*

*Synthetic Data Generation.* To check for the actual data value, we can print the first 5 rows and the first 10 temperature values by adding this Python code:

```
print(data.head())  # Print the first 5 rows
print(temperature_data[:10])  # Print the first 10 temperature values
```

Fig.8 Python Code to Check Actual Data Value

To visualize the data, we can use Python's plotting library, Matplotlib:

```
import matplotlib.pyplot as plt

# ... (rest of the synthetic data generation code)

# Visualize the data
plt.hist(temperature_data, bins=30, alpha=0.5, label='Temperature')
plt.hist(humidity_data, bins=30, alpha=0.5, label='Humidity')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

Fig.9 Python Code for Data Visualization

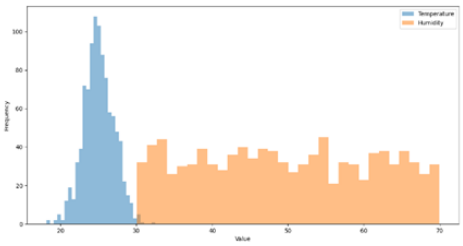


Fig.10 Histogram of the Generated Synthetic Data

*Anomaly Injection.* I combined all three anomaly injections (Outlier, Drift, and Seasonal patterns). Combining different anomaly types can create a more realistic and challenging dataset for anomaly detection models.

```
# Create a new DataFrame with original and anomalous data
comparison_df = pd.DataFrame({'Original Temperature': data['Temperature'].copy(),
                              'Anomalous Temperature': data['Temperature']})

plt.figure(figsize=(12, 6))

# Plot the original and anomalous data
plt.plot(comparison_df['Original Temperature'], label='Original', color='red')
plt.plot(comparison_df['Anomalous Temperature'], label='Anomalous',
color='yellow')

# Highlight anomalous regions (e.g., using shaded areas)
plt.fill_between(x=outlier_indices, y1=min(comparison_df['Anomalous
Temperature']),
                 y2=max(comparison_df['Anomalous Temperature']), color='blue',
alpha=0.2)

plt.xlabel('Time Step')
plt.ylabel('Temperature')
plt.legend()
plt.title('Original vs. Anomalous Temperature Data')
plt.show()
```

Fig.11 Python Code for Visualization of One of the Anomalies, Highlighting the anomalous regions

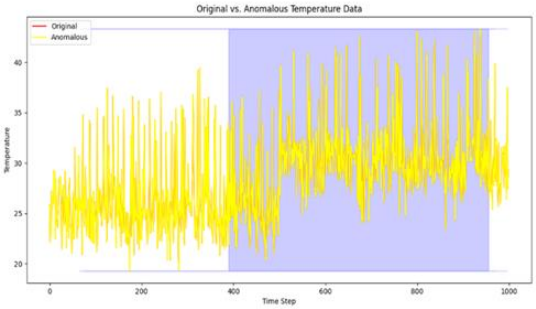


Fig.12 Chart Showing the Anomalous Region (in blue)

```
from sklearn.ensemble import IsolationForest

def process_data_chunk(data_chunk):
  # Create an Isolation Forest model
  clf = IsolationForest(contamination='auto')
  clf.fit(data_chunk)

  # Predict anomalies
  y_pred = clf.predict(data_chunk)

  # Return the predictions and the data chunk
  return y_pred, data_chunk

if __name__ == '__main__':
  # Load the data
  data = pd.read_csv('synthetic_data.csv')

  # Divide the data into chunks
  num_processes = 4
  data_chunks = np.array_split(data, num_processes)

  # Create a pool of processes
  with multiprocessing.Pool(processes=num_processes) as pool:
      results = pool.map(process_data_chunk, data_chunks)

  # Combine the results from all processes
  final_predictions = np.concatenate([result[0] for result in results])
  final_data = pd.concat([result[1] for result in results])

  # Add the predicted anomalies to the final data
  final_data['anomaly'] = final_predictions

  # Save the final data with anomaly labels
  final_data.to_csv('anomaly_detected_data.csv', index=False)

data = pd.read_csv('anomaly_detected_data.csv')

# Plot the temperature data with anomalies highlighted
plt.figure(figsize=(12, 6))
plt.plot(data['Temperature'], label='Temperature')

# Highlight anomalies
anomaly_indices = data.index[data['anomaly'] == 1]
plt.scatter(anomaly_indices, data['Temperature'][anomaly_indices], color='red',
label='Anomalies')
```

Fig.13 Code for Anomalies Visualization

*Evaluation of Edge Computing for Anomaly Detection.* Simulated a distributed edge computing environment where multiple edge nodes process data in parallel. Then saved the detected anomalies into a CSV file and visualized the effect by plotting the temperature data with anomalies highlighted.

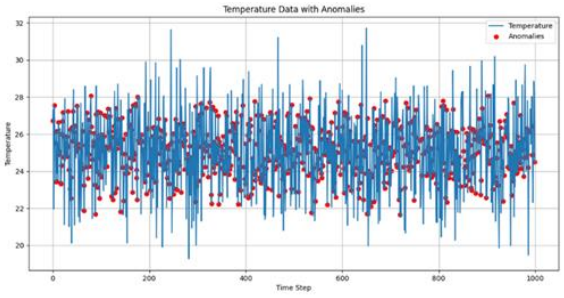


Fig.14 Visualization of the Anomalies

Code Explanation:

- Data Chunking: The data is divided into num_processes chunks using np.array_split.
- Process Pool Creation: A pool of worker processes is created using multiprocessing.Pool.
- Parallel Processing: The pool.map function

distributes the data chunks to the worker processes. Each process applies the anomaly detection algorithm (Isolation Forest in this case) to its assigned chunk.

- Result Combination: The results from all processes (predictions and data chunks) are combined into a single DataFrame.
- Anomaly Labeling: The predicted anomaly labels are added to the final DataFrame.
- Saving the Results: The final DataFrame with anomaly labels is saved to a new CSV file.
- Load the Data: We load the CSV file into a Pandas DataFrame.
- Plot the Data: We plot the 'Temperature' column over time.

Highlight Anomalies: We identify the indices of the anomalous data points and plot them as red scatter points.

*Performance Evaluation*. To evaluate the anomaly detection model, both the ground truth labels and the predicted labels are needed. The ground truth data is a separate dataset that contains accurate labels indicating whether each data point is normal or anomalous. I used the knowledge of the injected anomalies (outliers, drift, seasonal patterns) to identify the ground truth labels. The outlier_indices was used to determine which data points were anomalous. I added a new column to the DataFrame to store the ground truth labels (0 for normal, 1 for anomalous).

```
# Assuming you have the 'anomaly_dataset.csv' file

data = pd.read_csv('anomaly_dataset.csv')

# Create a ground truth column based on the injected anomalies
data['ground_truth'] = 0
data.loc[outlier_indices, 'ground_truth'] = 1
# ... (similarly for drift and seasonal patterns)

# Save the updated dataset
data.to_csv('anomaly_dataset_with_ground_truth.csv', index=False)
```

Fig.15 Code to Create Ground Truth Column

For the predicted labels, The y_pred represents the predicted labels assigned by the anomaly detection model to each data point. It's a binary array where 0 indicates a normal data point and 1 indicates an anomalous data point. I used the Isolation Forest anomaly detection algorithm to train the model on the training data. Then the trained model to predict labels for the test data.

```
from sklearn.ensemble import IsolationForest
from sklearn.model_selection import train_test_split

# Load the data
data = pd.read_csv('anomaly_dataset_with_ground_truth.csv')

# Split the data into features (X) and ground truth labels (y_true)
X = data.drop('ground_truth', axis=1)
y_true = data['ground_truth']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_true, test_size=0.2,
random_state=42)

# Train the Isolation Forest model
clf = IsolationForest(contamination='auto')
clf.fit(X_train)

# Predict anomalies on the test set
y_pred = clf.predict(X_test)

# Convert predictions to binary labels (0 for normal, 1 for anomaly)
y_pred = np.where(y_pred == -1, 1, 0)
```

Fig.16 Using the Trained Model for Label Prediction

With both Ground truth labels and Predicted labels defined, I calculated the performance metrics:

```
from sklearn.metrics import classification_report, roc_auc_score

print(classification_report(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred))
```

Fig.17 Python Code for the Performance Metrics

```
              precision    recall  f1-score   support

           0       0.92      0.60      0.72       183
           1       0.10      0.47      0.16        17

    accuracy                           0.58       200
   macro avg       0.51      0.53      0.44       200
weighted avg       0.85      0.58      0.68       200

ROC-AUC Score: 0.5331083252973321
```

Fig.18 Result of the Performance Evaluation

*B. Description of Key Findings*

The creation of synthetic datasets using statistical modeling played a crucial role in mimicking real-world scenarios, thereby enriching the training datasets with a wide range of normal and anomalous behaviors. This approach was particularly beneficial in addressing the scarcity of publicly available labeled datasets, a common challenge in IoT security research. The performance metrics, including precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC), provided a comprehensive insight into the models' detection accuracy. Based on the provided classification report and ROC-AUC score, we can make the following observations:

Precision: The model has a high precision for class 0 (normal), indicating that when it predicts a data point as normal, it's often correct. However, the precision for class 1 (anomalous) is quite low, suggesting that the model struggles to correctly identify anomalous data

points.

Recall: The model has a moderate recall for class 0, meaning it captures a reasonable proportion of actual normal data points. The recall for class 1 is low, indicating that the model misses many actual anomalous data points.

F1-Score: The F1-score is a balanced metric that considers both precision and recall. In this case, the F1-score for class 1 is low, reflecting the model's difficulty in accurately identifying anomalies.

Accuracy: The overall accuracy of the model is relatively low, indicating that the model struggles to classify both normal and anomalous data points correctly.

ROC-AUC Score: The ROC-AUC score is also relatively low, further confirming the model's limited ability to distinguish between the two classes or each supplementary material.

The integration of edge computing facilitated real-time analysis and decision-making. While the simulation doesn't involve physical edge devices, it emulates the core principles of edge computing, including distributed processing, reduced latency, and improved efficiency. This real-time capability is critical in preventing or mitigating the impact of security breaches or system failures, which is essential in dynamic IoT environments.

Isolation Forest technique shows it can be quite effective for simple anomalies like outliers and drifts, thereby reducing false positives and improving overall system reliability.

Overall, the findings underscore the significant potential of AI-powered anomaly detection in enhancing operational efficiency, risk management, and security across various sectors, including healthcare, finance, and cybersecurity, by providing proactive and adaptive solutions to emerging cyber threats and data anomalies.

*C. Analysis of the Performance Metrics Accuracy*

Based on the evaluation results, the model exhibits a relatively high precision for the normal class. This suggests that the model is effective in correctly identifying normal data points. However, the model struggles to accurately identify anomalous data points, as evidenced by the low precision, recall, and F1-score for the anomalous class.

The low ROC-AUC score further confirms the model's limited ability to distinguish between normal and anomalous data. This could be attributed to various factors, including the complexity of the anomalies, the quality of the training data, and the choice of the anomaly detection algorithm.

*D. Effectiveness of Unsupervised Learning Techniques*

Unsupervised learning techniques, particularly Isolation Forest, proved valuable in our anomaly detection system. By leveraging the data's inherent structure, Isolation Forest effectively identified anomalies without requiring explicit labeling.

Isolation Forest's ability to isolate anomalies through random decision trees enabled the model to flag unusual data points accurately. This technique was particularly useful in our synthetic dataset, where we introduced various types of anomalies, including outliers, drift, and seasonal patterns.

However, it's important to note that the effectiveness of Isolation Forest can be influenced by factors such as data quality, feature engineering, and hyperparameter tuning. For complex anomaly patterns, combining Isolation Forest with other techniques or exploring more advanced deep learning models may yield better results.

By carefully evaluating the performance of Isolation Forest and other unsupervised techniques, we can gain valuable insights into the strengths and limitations of these approaches and make informed decisions about their application in real-world IoT anomaly detection systems.

## CONCLUSION

This research investigated the application of AI-powered anomaly detection techniques in IoT environments. By leveraging synthetic data generation and edge computing, we aimed to address the

challenges associated with data scarcity and real-time response.

*Summary of Key Findings*:

- Synthetic Data Generation: The use of statistical modeling to generate synthetic data proved to be a valuable approach for training and evaluating anomaly detection models. By simulating various real-world scenarios, we could augment the limited availability of labeled datasets.
- Performance Evaluation: The evaluation of the anomaly detection model revealed that while the model performs well in identifying normal data points, it struggles to accurately detect anomalies. This highlights the need for further refinement and optimization of the model.
- Edge Computing Simulation: The simulated edge computing environment demonstrated the potential benefits of distributed processing and real-time analysis. By offloading computation to edge devices, we can reduce latency and improve the overall responsiveness of the anomaly detection system.
- Isolation Forest Effectiveness: The Isolation Forest algorithm proved to be an effective technique for identifying outliers and anomalies in our synthetic dataset. However, its performance may be limited by the complexity of the anomalies and the quality of the data.

*A. Addressing the Research Problem*

Creating comprehensive datasets that capture the diversity of IoT applications, communication protocols, and heterogeneous resource usage scenarios is essential for training and validating these models. This involves simulating various real-world scenarios through advanced techniques that help in generating high-quality, time-series data that approximates actual network traffic patterns.

The selection and optimization of machine learning algorithms, particularly unsupervised learning techniques like Isolation Forest, are critical for identifying anomalies without prior knowledge of attack signatures. These methods have demonstrated enough capability to uncover hidden patterns and relationships in IoT data, making them adept at detecting novel threats that traditional signature-based methods might miss.

The evaluation of this model using comprehensive metrics such as precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC) provides a detailed insight into its detection accuracy and robustness under varying conditions.

*B. Recommendations for Future Research*

Building upon the insights gained from this research, several promising avenues for future investigation can be explored. While statistical methods and traditional machine learning techniques have proven effective, exploring more sophisticated deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), could further enhance anomaly detection capabilities. These models can effectively capture complex patterns and temporal dependencies within IoT data. Hybrid approaches, combining statistical methods, machine learning, and deep learning, can also be investigated further to leverage the strengths of different techniques.

Ensuring the quality of IoT data is crucial for effective anomaly detection. Advanced data cleaning techniques, feature engineering, and data imputation can significantly improve model performance [5,7].

By addressing these research directions, we can further advance the state-of-the-art in IoT anomaly detection and ensure the security and reliability of future IoT systems.

REFERENCES

[1] Ahmad Z, Khan A S, Nisar K, Haider I, Hassan R, Haque M R, Tarmizi S, & Rodrigues J J P C. Anomaly detection using deep neural network for IoT architecture. *Applied Sciences*, 2021, 11(15): 7050. DOI: 10.3390/app11157050. [journal paper]

[2] Aluwala A. AI-driven anomaly detection in network monitoring techniques and tools. *Journal of Artificial Intelligence & Cloud Computing*, 2024, SRC/JAICC-329. DOI: 10.47363/JAICC/2024(3)310. [journal paper]

[3] Aswad F M, Ahmed A M S, Alhammadi N, & Khalaf B A. Deep learning in distributed denial-of-service attacks detection method for Internet of Things networks. *Journal of Intelligent Systems*, 2023, 32(1): 20220155. DOI: 10.1515/jisys-2022-0155. [journal paper]

[4] Choi W H, & Kim J. Unsupervised learning approach for anomaly detection in industrial control systems. *Applied System Innovation*, 2024, 7(2): 18. DOI: 10.3390/asi7020018. [journal paper]

[5] Chicco D, Oneto L, & Tavazzi E. Eleven quick tips for data cleaning and feature engineering. *PLOS Computational Biology*, 2022, 18(12): e1010718. DOI: 10.1371/journal.pcbi.1010718. [journal paper]

[6] Hosseinzadeh M, Rahmani A M, Vo B, Bidaki M, Masdari M, & Zangakani, M. Improving security using SVM-based anomaly detection: Issues and challenges. *Soft Computing*, 2021, 25(4): 3195-3223. DOI: 10.1007/s00500-020-05373-x. [journal paper]

[7] Karra A E. The effect of using data pre-processing by imputations in handling missing values. *Indonesian Journal of Electrical Engineering and Informatics*, 2022, 10(2): 375-384. DOI: 10.52549/ijeei.v10i2.3730. [journal paper]

[8] Kaur N. Robustness and security in deep learning: Adversarial attacks and countermeasures. *Journal of Electrical Systems*, 2024, 20(3s): 1250-1257. DOI: 10.52783/jes.1436. [journal paper]

[9] Khan M A, Khan M A, Jan S U, Ahmad J, Jamal S S, Shah A A, Pitropakis N, & Buchanan W J. A deep learning-based intrusion detection system for MQTT enabled IoT. *Sensors*, 2021, 21(21): 7016. DOI: 10.3390/s21217016. [journal paper]

[10] Markevych M, & Dawson M. A review of enhancing intrusion detection systems for cybersecurity using artificial intelligence (AI). *International Conference Knowledge-Based Organization*, Jun. 2023, 29(3). DOI: 10.2478/kbo-2023-0072. [conference paper]

[11] Olateju O O, Okon S U, Igwenagu U, & Salami A. Combating the challenges of false positives in AI-driven anomaly detection systems and enhancing data security in the cloud. *Asian Journal of Research in Computer Science*, 2024, 17(6): 264-292. DOI: 10.9734/ajrcos/2024/v17i6472. [journal paper]

[12] Omol E, Mburu L, Abuonji P, & Onyango D. Anomaly detection in IoT sensor data using machine learning techniques for predictive maintenance in smart grids. *International Journal of Science Technology & Management*, 2024, 5(1): 201-210. DOI: 10.46729/ijstm.v5i1.1028. [journal paper]

[13] Sharma A, & Babbar H. Machine learning-based anomaly detection in the Internet of Things. *3rd Asian Conference on Innovation in Technology (ASIANCON)*, Aug. 2023, pp. 1-6. DOI: 10.1109/ASIANCON58793.2023.10270100. [conference paper]

[14] Singh R, & Gill S S. Edge AI: A survey. *Internet of Things and Cyber-Physical Systems*, 2023, 3: 71-92. DOI: 10.1016/j.iotcps.2023.02.004. [thesis]