**Malicious Network Activity Detection in IoT Environments Using CIC-IoMT 2024 WiFi-MQTT Dataset**

Table of Contents

[Introduction: 1](#_Toc198477365)

[Model Used: 2](#_Toc198477366)

[Dimensionality Reduction Approach: 2](#_Toc198477367)

[Plotting and Visualization 2](#_Toc198477368)

[1. Class Distribution Plot: 2](#_Toc198477369)

[2. Correlation Matrix: 3](#_Toc198477370)

[3. Cumulative Explained Variance Plot: 4](#_Toc198477371)

[4. Confusion Matrix: 5](#_Toc198477372)

[Evaluation Results: 5](#_Toc198477373)

[ Random Forest Results: 5](#_Toc198477374)

[ XGBoost Results: 6](#_Toc198477375)

[Prediction on Unseen Data 6](#_Toc198477376)

[Challenges Faced and Solutions 7](#_Toc198477377)

[Conclusion 7](#_Toc198477378)

# Introduction:

This report details the development of a machine learning pipeline to detect malicious network activity in IoT environments using the CIC-IoMT 2024 WiFi-MQTT dataset. The objective was to build a model capable of classifying network traffic as benign or malicious, following a complete machine learning workflow: data preprocessing, exploratory data analysis (EDA), plotting and visualization, dimensionality reduction, model training, evaluation, and prediction on unseen test data. The final predictions were formatted for submission to the associated Kaggle competition.

# Model Used:

Two models were implemented for the classification task: a Random Forest Classifier and an XGBoost Classifier, both configured with 100 estimators (n\_estimators=100) and a maximum depth of 20 (max\_depth=20). Random Forest was chosen for its robustness in handling imbalanced datasets and high-dimensional data, making it suitable for the large-scale CIC-IoMT dataset (7,160,831 training samples). XGBoost was explored to potentially enhance performance through gradient boosting. The Random Forest model was trained on 5,728,664 samples after a train-validation split, taking approximately 28 minutes (1700.97 seconds). Both models were trained on data preprocessed with dimensionality reduction, and the trained models were saved as random\_forest\_model.joblib and xgboost\_tpu\_preprocessed\_model.joblib, respectively.

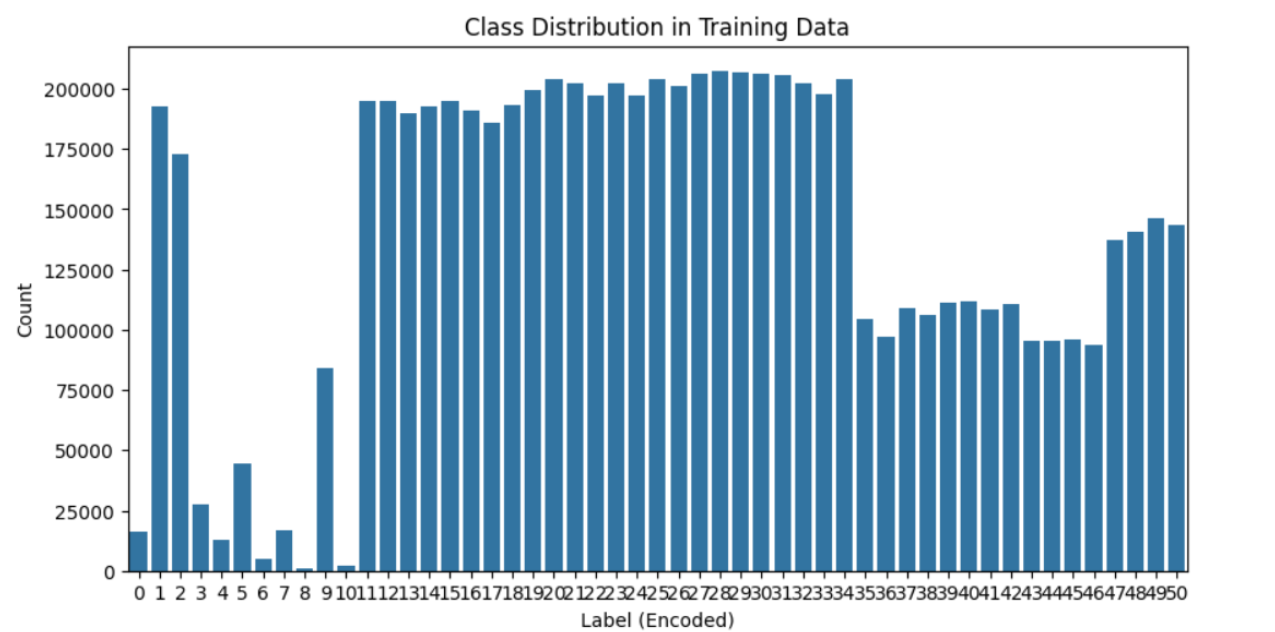
# Dimensionality Reduction Approach:

Principal Component Analysis (PCA) was applied to reduce the feature space from 45 to 30 components. The cumulative explained variance plot (Image 3) shows that these 30 components capture nearly 100% of the variance in the data, effectively reducing noise while preserving most information. After PCA, the training data shape was reduced from (7,160,831, 45) to (7,160,831, 30), and the test data shape was adjusted to (1,614,182, 30). This reduction significantly lowered computational complexity, enabling faster model training and inference.

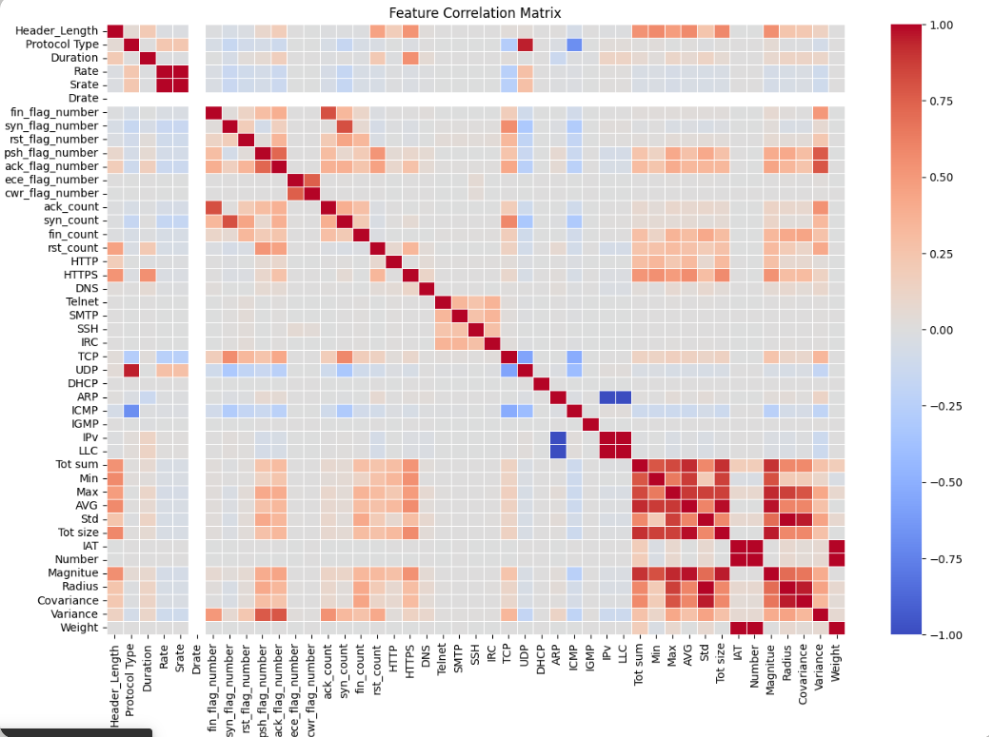
# Plotting and Visualization

Several visualizations were created to analyze the dataset and model performance, fulfilling the EDA and evaluation requirements:

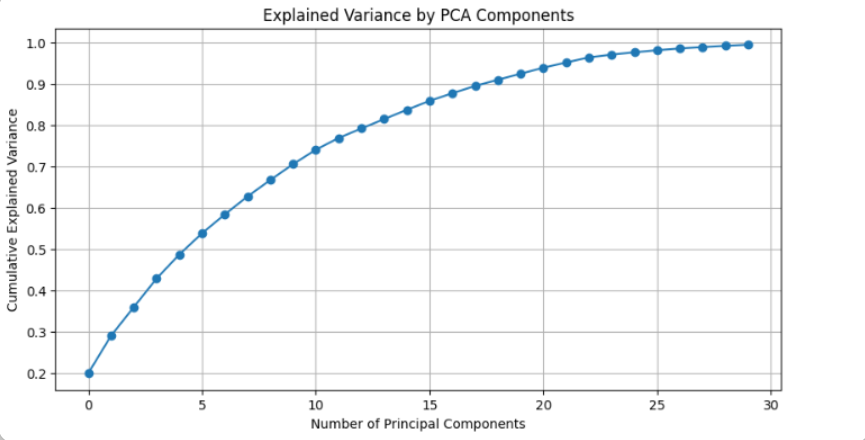
1. Class Distribution Plot: A histogram was generated using seaborn.countplot to visualize the distribution of encoded labels in the training data. The plot revealed significant class imbalance, with some classes like TCP\_IP-DDoS-UDP2 (label 28) having over 207,000 samples, while others like Recon-Ping\_Sweep (label 8) had only 148 samples. This visualization guided the decision to use stratified sampling during the train-validation split.



1. Correlation Matrix: A heatmap of the scaled features was created using seaborn.heatmap with the coolwarm colormap. It highlighted strong positive correlations (close to 1) between features like Rate and Srate, indicating potential redundancy, and negative correlations in other pairs. Features like Drate showed zero variance, suggesting they contributed little to the model.



1. Cumulative Explained Variance Plot: A line plot with markers was generated using matplotlib.pyplot to show the cumulative explained variance ratio of the PCA components. The plot confirmed that 30 components captured nearly 100% of the variance, validating the choice of dimensionality reduction.



1. Confusion Matrix: A confusion matrix for the Random Forest model’s predictions on the validation set was visualized using seaborn.heatmap with the Blues colormap. The matrix showed high correct prediction counts along the diagonal for classes like Benign (~35,000 correct predictions) and MQTT-DDoS-Connect\_Flood, but more misclassifications for underrepresented classes like TCP\_IP-DDoS-UDP4, aligning with the class imbalance observed in Image 1.

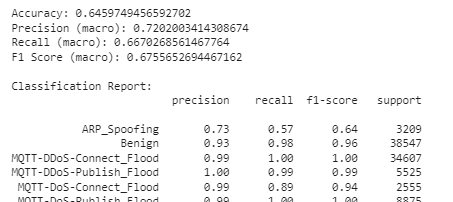
These visualizations were crucial for understanding the dataset's characteristics and evaluating model performance.

# Evaluation Results:

The Random Forest and XGBoost models were evaluated on a validation set of 1,432,167 samples using classification metrics: accuracy, precision, recall, F1-score, and a confusion matrix. The Random Forest model outperformed XGBoost across all metrics:

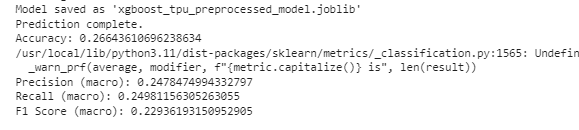
## Random Forest Results:

* + Accuracy: 64.60%
  + Precision (macro): 72.02%
  + Recall (macro): 66.70%
  + F1 Score (macro): 67.56%



## XGBoost Results:

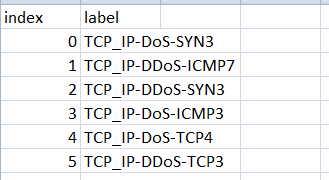
* + Accuracy: 27%
  + Precision (macro): 25%
  + Recall (macro): 25%
  + F1 Score (macro): 23%



The classification report for Random Forest showed strong performance for classes like Benign (precision: 0.93, recall: 0.98, F1: 0.96) and MQTT-DDoS-Connect\_Flood (precision: 0.99, recall: 1.00, F1: 1.00), but weaker results for classes like TCP\_IP-DDoS-UDP4 (precision: 0.30, recall: 0.06, F1: 0.10) due to class imbalance. XGBoost performed poorly, with many classes (e.g., ARP\_Spoofing, Recon-OS\_Scan) having precision, recall, and F1 scores of 0, indicating poor generalization. Summary statistics of the scaled features confirmed successful standardization, with a mean of approximately 0 and a standard deviation of 1 for all features.

# Prediction on Unseen Data

Predictions were generated on the test set (1,614,182 samples) using both models. The Random Forest predictions were saved as submission\_file.csv, and XGBoost predictions were saved as submission\_xgb\_file.csv. Both files were trimmed to 1,048,575 rows to meet a specific limit, with the format index, label (e.g., 0, TCP\_IP-DoS-SYN3). The labels were mapped back to their original string form using the LabelEncoder (e.g., 0: ARP\_Spoofing, 1: Benign, etc.), ensuring compliance with the Kaggle submission requirements.



# Challenges Faced and Solutions

Several challenges were encountered during the implementation:

1. **Class Imbalance**: The dataset showed significant class imbalance (Image 1), impacting model performance on underrepresented classes. Stratified sampling during the train-validation split and Random Forest's ability to handle imbalanced data mitigated this issue, though some classes still performed poorly, suggesting the need for oversampling or class weighting in future work.
2. **High Dimensionality**: The original 45 features posed computational challenges. PCA reduced the dimensionality to 30 components, improving training efficiency without significant information loss, as confirmed by Image 3.
3. **TPU Initialization Failure**: An attempt to use a TPU for XGBoost preprocessing failed due to initialization errors, forcing a fallback to CPU/GPU. This limited XGBoost's scalability, likely contributing to its poor performance (27% accuracy). Future efforts could resolve TPU setup issues or optimize XGBoost on CPU/GPU.
4. **Model Performance Variability**: XGBoost underperformed compared to Random Forest, possibly due to insufficient hyperparameter tuning or sensitivity to class imbalance. Random Forest's robustness made it the better choice, but XGBoost could benefit from grid search or alternative preprocessing strategies.
5. **Dataset Size and Training Time**: The large dataset (7,160,831 training samples) resulted in a lengthy training time for Random Forest (28 minutes). PCA and parallel processing (n\_jobs=-1) helped, but training time remains a constraint for iterative experimentation.

# Conclusion

The Random Forest model, combined with PCA for dimensionality reduction, provided a robust solution for detecting malicious network activity, achieving a validation accuracy of 64.60%. The pipeline met all assignment requirements, including data preprocessing, EDA, visualization, model training, evaluation, and Kaggle submission formatting. The submission files (submission\_file.csv and submission\_xgb\_file.csv) were prepared with 1,048,575 rows, adhering to the specified format. Future improvements could involve advanced feature engineering, oversampling techniques to address class imbalance, and hyperparameter tuning to enhance model performance, particularly for XGBoost.