164 Project Report

Ayanna Avalos **Jonathan Garza**

Breast Cancer Wisconsin (Diagnostic) Dataset

The Breast Cancer Wisconsin (Diagnostic) datasets available from the UCI Machine Learning Repository and is used for predicting whether a tumor is malignant or benign. This dataset presents a binary classification task of predicting whether a tumor is malignant or benign. It has 37 citations which deemed it trustworth and usable. Upon closer inspection, the dataset is remarkably clean and has no missing values. This reduces the amount of preprocessing needed which is why this dataset was selected.

The overall structure of the dataset consists of 569 instances and 32 features. The first two features are the *ID* and the *Target Variable (Diagnosis)*. The *Diagnosis* is either an *M* for malignant or a *B* for benign, this column is later encoded so that malignant instances are assigned 1 while benign instances are assigned 0. The rest of the 30 features are continuous values that are extracted from images of cell-nuclei (texture, radis, area, etc.) Therefore, the dataset consists of categorical and continuous (numerical, float64) values.

RT-IoT2022 Dataset

The RT-IoT2022 dataset from the UCI Machine Learning Repository is a dataset consisting of labeled network traffic records collected in a real-time IoT environment. The dataset will be used for classification between malicious and normal activity, which includes the following attacks: ARP poisoning, DDoS, DoS, NMAP scans, SSH brute-force, and MQTT attacks. The classification is based on features such as flow_duration, protocol_type, header_length, rate, and other numerical features for traffic behavior. The targeting variable is the classification of attack or normal behavior. The dataset consists of both numerical and categorical features, with a majority of them being numerical.

For preprocessing the data, we will first take a look at the missing values, which the dataset is fairly free of. For our method of handling missing values, we use a simple imputer. We will then encode our categorical values using the label encoder and then apply a standard scaler. After preparing the data, we will then start training and testing our model using an 80/20 split, 80% training and 20% testing.

Dataset Comparison

The Breast Cancer Wisconsin (Diagnostic) and RT-IoT2022 datasets are both used for classification but differ in size, scope, and domain. The Breast Cancer dataset focuses on classifying tumors as malignant or benign, making it a specific task. It is also a small dataset with 569 instances and 30 features. In contrast, RT-IoT2022 deals with identifying various IoT network intrusions and offers a broader classification task involving multiple attack types. It is much larger, with over 123,000 instances and 83 features. The Breast Cancer dataset is related to medical diagnostics while RT-IoT2022 is focused on cybersecurity.

Breast Cancer Wisconsin (Diagnostic) Dataset

Prior Work - Machine learning in medicine: A practical introduction

This work trained and compared three models: Regularized General Linear Model, Support Vector Machines, and single-layer Artificial Neural Networks. The dataset had 683 samples which were randomly split into evalution (456) and validation (227) samples. They trained each model using the evaluation set, then tested using the validation set. After, they tested ensemble methods (such as averaging and voting) to combine model outputs.

There were three metrics: accuracy, sensitivity, and specificity. All three models achieved high accuracy, ranging from 94 to 96% accuracy. As for sensitivity (recall for malignant cases), the models ranged from 97 to 99%. Lastly, models ranged from 85 to 94% for specificity (true negative rate for benign cases). The best performing model was the Support Vector Machines with a radial basis function kernel. As for the ensemble method results, they were the following: accuracy: 97%, sensitivity: 99%, and specificity: 95%. By combining all the predictions from the three models, performance was improved overall (especially for sensitivity and specificity).

Methods

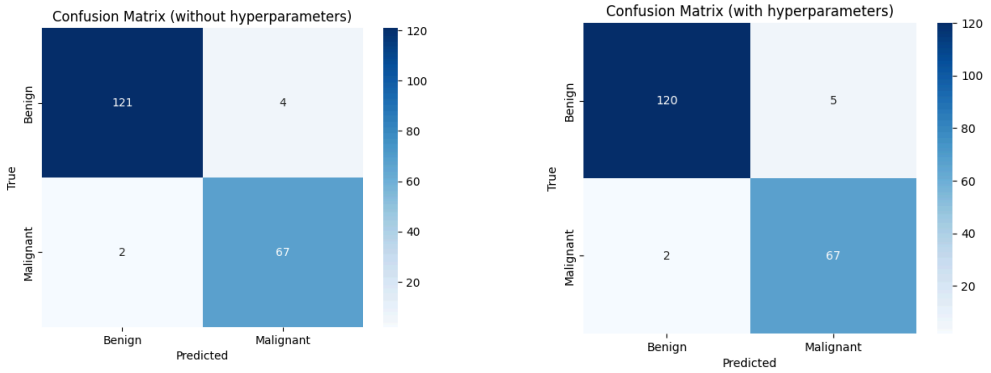
The dataset was split into 66% for training and 34% for testing, totaling 376 samples for training and 193 samples for testing, in order to closely match the prior work's split and allow an accurate comparison between the models. The prior work used a dataset of 683 samples, split into 456 evaluation samples (66.76%) for training and 227 validation samples (33.24%) for testing, so the percentage breakdown is similar despite the difference in total sample size. Although, it is not clear why the dataset has fewer instances now.

Two machine learning algorithms were chosen, the first was logistic regression because it is well-suited for binary classification. This aligns with the task of distinguishing between malignant and benign diagnoses. Its simplicity and efficiency make it a good choice for this dataset. Two logistic regression models were trained: one using the default settings (without hyperparameter tuning) and the other with hyperparameter optimization through GridSearchCV. The baseline model was trained and evaluated using performance metrics such as accuracy, sensitivity, specificity, precision, recall, F1-score, and ROC-AUC, with its confusion matrix visualized. The GridSearchCV method was then applied to find the best combination of hyperparameters for the Logistic Regression model, including regularization strength (C), solver type, and other relevant parameters.

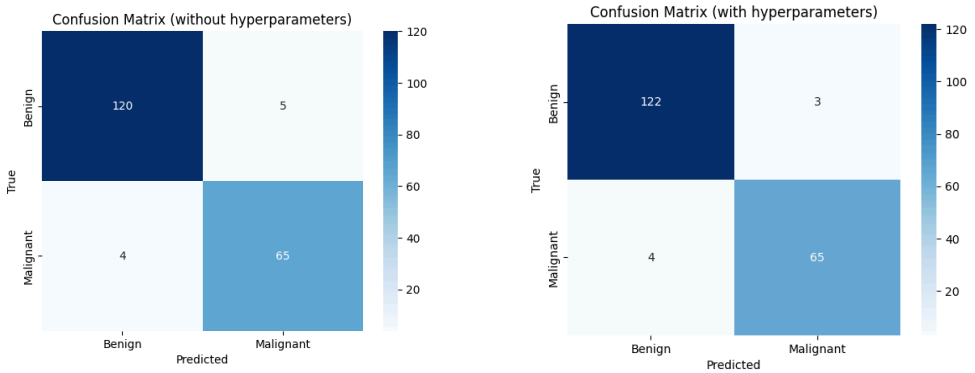
The second machine learning algorithm was k-nearest neighbors (KNN). K-nearest neighbors (KNN) is a good model for this dataset because it is simple, handles non-linear relationships well, and effectively classifies data based on similarity. Like the previous algorithm, two KNN models were trained: one using the default settings and one used hyperparameter optimization through GridSearchCV.

Results

The results compare two logistic regression models: one without hyperparameter tuning and one with tuning through GridSearchCV. The baseline model achieved 96.91% accuracy, 97.10% sensitivity, and 96.80% specificity, with strong metrics like 0.9437 precision, 0.9571 F1-score, and 0.9695 ROC-AUC. The tuned model, with hyperparameters C: 100, max_iter: 100, solver: 'liblinear', and tol: 0.0001, showed slightly lower performance. It had 96.39% accuracy, 97.10% sensitivity, and 96.00% specificity, with 0.9306 precision and 0.9504 F1-score. Despite the slight decrease, the hyperparameter-tuned model still performed well but did not surpass the baseline model. There was one instance that was incorrectly classified by the tuned model that the baseline model correctly classified. This explains the very slight discrepancy between the results.



The K-Nearest Neighbors (KNN) model without hyperparameter tuning achieved an accuracy of 95.36%, with a sensitivity of 94.20% and specificity of 96.00%. Precision was 92.86%, and the F1-score was 93.53%, reflecting a good balance. After hyperparameter tuning, accuracy increased to 96.39%, with specificity improving to 97.60%. Precision rose to 95.59%, and the F1-score increased to 94.89%. The ROC-AUC score also improved from 95.10% to 95.90%, highlighting a better ability to distinguish between classes. Hyperparameter tuning led to overall performance improvements, especially in specificity and precision. The confusion matrices below show that after tuning, the model correctly classifies two additional instances compared to the previous model, which accounts for the difference in the results.



Comparison to Prior Work

Metric	Accuracy	Sensitivity (Recall)	Specificity
Model with Hyperparameter Tuning	96.39%	97.10%	96.00%
Model without Hyperparameter Tuning	96.91%	97.10%	96.80%
Range from Previous Studies	94% - 96%	97% - 99%	85% - 94%

The logistic regression models, with and without hyperparameter tuning, show strong performance compared to previous studies. The accuracy of the models (96.39% with tuning and 96.91% without tuning) is within the 94% to 96% range reported in prior works. The models also demonstrate excellent sensitivity (recall) of 97.10%, consistent with the 97% to 99% range found in earlier studies. Specificity is high as well, with values of 96.00% and 96.80%, which are higher than the 85% to 94% range seen in prior research.

Metric	Accuracy	Sensitivity (Recall)	Specificity
Model with Hyperparameter Tuning	96.39%	94.20%	97.60%
Model without Hyperparameter Tuning	95.36%	94.20%	96.00%
Range from Previous Studies	94% - 96%	97% - 99%	85% - 94%

The K-Nearest Neighbors (KNN) models perform similarly to the prior work in terms of accuracy, sensitivity, and specificity. The accuracy of both models (95.36% without tuning and 96.39% with tuning) falls within the 94% to 96% range seen in previous studies. Sensitivity is consistent at 94.20%, matching the 97% to 99% range from prior work. Specificity at 96% and improves with tuning, reaching 97.60%, which is higher than the 85% to 94% range from previous models. Overall, the KNN models are competitive, with the tuned model showing better specificity.

In conclusion, the logistic regression model without hyperparameter tuning performed the best overall, with an accuracy of 96.91%, sensitivity of 97.10%, and specificity of 96.80%. The tuned logistic regression model showed slightly lower performance, and both K-Nearest Neighbors (KNN) models had similar results, with the tuned KNN model improving in specificity and precision. However, the baseline logistic regression model was the best performer across key metrics.

RT-IoT2022 Dataset

Related Works

In the report by B S Sharmila and Rohini Nagapadma, they also use the dataset to analyze a model fit for a Raspberry Pi. They had other constraints they had to consider for the model, such as processing and memory capacity, due to the hardware of the Raspberry Pi. They used an Autoencoder, which is an unsupervised neural network, and quantized autoencoders for the hardware limitations. Their performance for their methods is listed in a table below. As you can

see, their models performed well with this classification task, with an accuracy of 96% or more and an F-1 score of 97.24% or more.

Model	Accuracy	Precision	Recall	F-1 Score
Autoencoder	98.40%	98.39%	98.40%	98.39%
QAE-f16	97.25%	97.24%	97.25%	97.24%
QAE-u8	96.35%	96.35%	96.36%	98.10%

To achieve their high performance, they performed some optimization and preprocessing techniques. For example, they did pruning, which removes neurons (similar to dropout for MLP), weight clustering, and quantization to reduce memory size. The quantization was crucial for their ability to get this model working on a Raspberry Pi. Additionally, they used standardization and feature selection to reduce overfitting and focus the model on the most relevant data. Their results show a strong balance between efficiency and detection accuracy, making their approach well-suited for real-time IoT intrusion detection.

Model

For our first model, we decided to use the Logistic Regression model from scikit-learn to classify network traffic in the RT-IoT2022 dataset. Our Logistic Regression model will go through 1000 iterations with a random state of 42. The model was trained on the preprocessed feature set and evaluated on a test set. After training, predictions were made using the predict method, and performance was evaluated using the classification report, which provides key metrics such as precision, recall, F1-score, and support for each class. This Logistic Regression model serves as a strong baseline for comparison due to its simplicity, speed, and well-understood behavior in multi-class classification tasks.

For our second model, we decided to use a Multilayer Perceptron (MLP) using scikit-learn's MLPClassifier to evaluate the performance of a supervised neural network on the RT-IoT2022 dataset. Due to the feedforward and non-linear decision boundary characteristics of MLPs should theoretically perform well for this multi-class classification dataset. Our model was configured with a single hidden layer containing 100 neurons, a maximum of 300 training iterations, and a random state of 42. The MLP is also trained and tested on the same split as the Logistic Regression model to ensure we can effectively compare the two models. Performance metrics such as precision, recall, and F1-score were calculated to assess the classifier's ability to detect various traffic types effectively.

Tuning

We tuned the Logistic Regression model using Grid Search. For our Grid Search parameters, we decided to use our scaled dataset for X, our normal y, 5 folds, and using the

accuracy scoring parameter. We received very similar results to our model without the Grid Search parameters, but it was a lot more computationally expensive. Our Grid Search model took about 7-8 minutes to complete, while our basic logistic regression model takes about 2-3 minutes. I decreased the number of folds to 3, and it decreased the computation time, but the accuracy was about 2% less at a 97% accuracy score. The only difference in accuracy was <1% of an increase. Because of this, it would be inefficient to stick with the Grid Search model and instead just use the Logistic Regression model.

Our MLP before tuning already achieved an accuracy score of 99.61%, with some classifications being 100%. After altering the layers and adding optimizers such as Adam, the performance only changed by about 0.2%, but it was more computationally expensive. Both cases, Logistic Regression and MLP, didn't see much improvement based on tuning. Both models already performed well without tuning and at lower computational costs.

Performance

Both our Logistic Regression and Multilayer Perceptron models performed exceptionally well for this task. Our Logistic Regression model had an average accuracy of 98.80% but some attack labels were not as accurate. For example, for attacks with not many instances in the data set, such as Metasploit_Brute_Force_SSH and NMAP_FIN_SCAN, we had precision scores of 86% and 71%, respectively. The two attack types had about 20 combined instances in the total dataset. For our attacks that had more than 75 total instances, we saw precision scores of 90% or more. Our attack that had the most mislabeled instances was the Wipro_bulb attack, with scores of 55% for recall and 65% for F1-score. This attack seemed to be constantly mislabeled with ARP_Poisoning, which isn't for no reason, as they are similar, since the Wipro attack uses ARP Poisoning or a very similar method.

The MLP performed better than the Logistic Regression model and tended to make up for where the Logistic Regression model lacked. The MLP had an average accuracy of 99.61% and had increased performance for attacks like Metasploit_Brute_Force_SSH and NMAP_FIN_SCAN. These attacks had precision scores of 86% and 100%, respectively, and the Wipro_bulb attack had an 84% for recall and 90% for F1-score. Although both models performed exceptionally good, the MLP did outperform the Logistic Regression model and made up for where it lacked. I will attach the graphs of all metrics for both below.

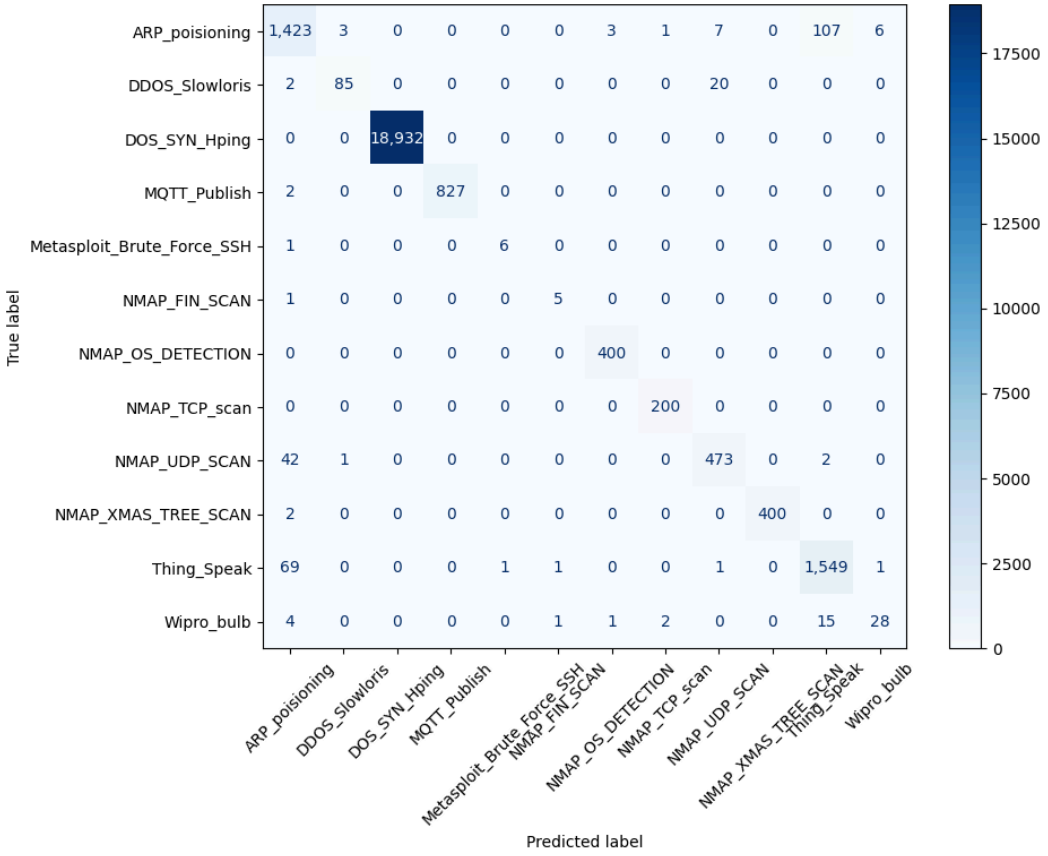
Conclusion

Overall, both our Logistic Regression and MLP models performed exceptionally well on the RT-IoT2022 dataset. Compared to the model used on the Raspberry Pi developed by B. S. Sharmila and Rohini Nagapadma, we achieved accuracies above their 96% and F1-scores of 97% using quantized autoencoders. One main reason for this is that our hardware wasn't as restricted as theirs, as we used a modern computer compared to their Raspberry Pi with very limited resources. Our better-performing model, the MLP, achieved an average accuracy of 99.61% and performed better than the Logistic Regression model for labels that didn't have many instances.

This highlights that while resource optimization is critical for edge devices, models running on more capable hardware can prioritize achieving higher predictive performance without needing as many trade-offs.

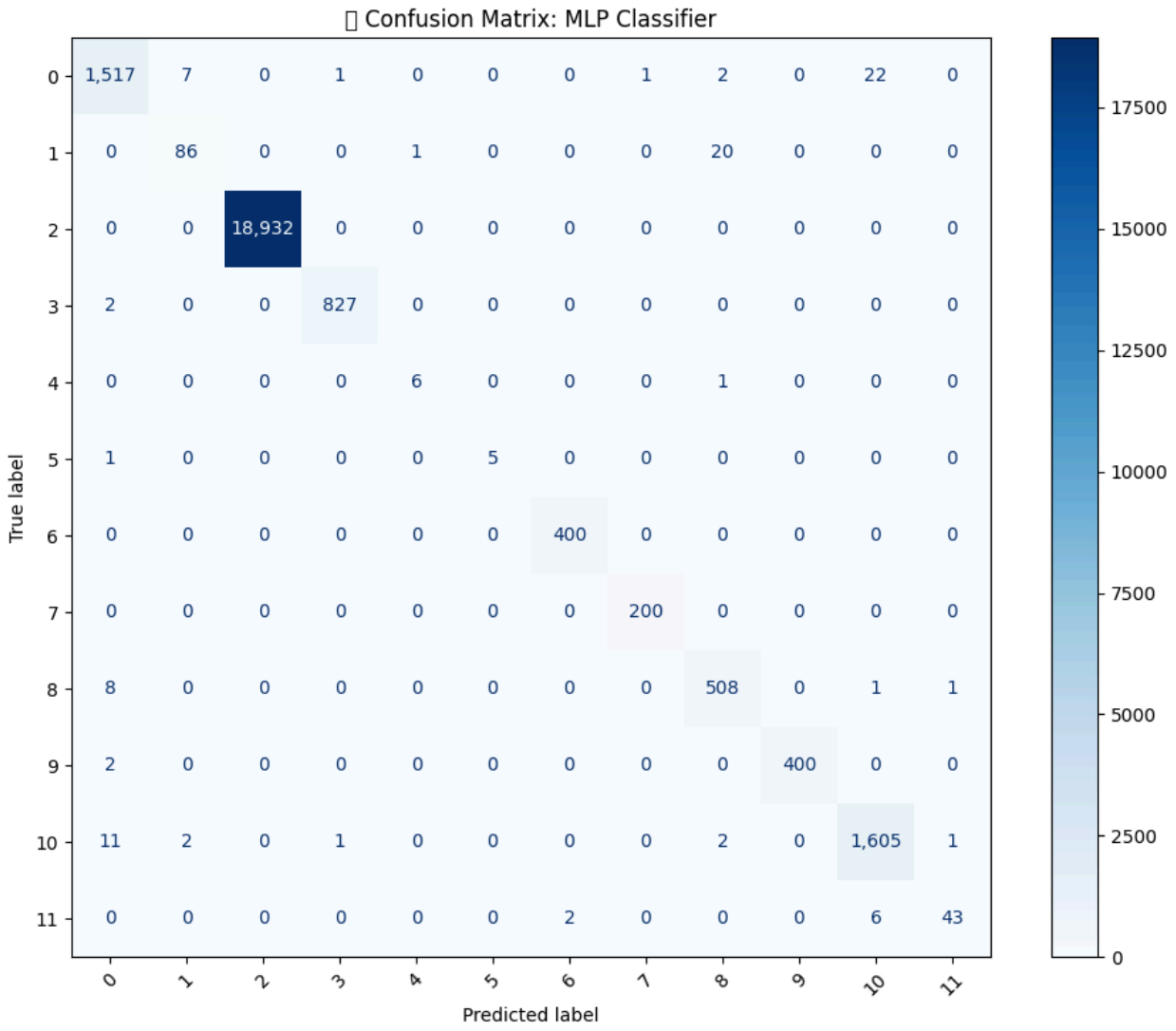
LR MODEL:

Class	Precision	Recall	F1-Score	Instances
ARP_poisioning	0.92	0.92	0.92	1550
DDOS_Slowloris	0.96	0.79	0.87	107
DOS_SYN_Hping	1	1	1	18932
MQTT_Publish	1	1	1	829
Metasploit_Brute_Force_SSH	0.86	0.86	0.86	7
NMAP_FIN_SCAN	0.71	0.83	0.77	6
NMAP_OS_DETECTION	0.99	1	1	400
NMAP_TCP_scan	0.99	1	0.99	200
NMAP_UDP_SCAN	0.94	0.91	0.93	518
NMAP_XMAS_TREE_SCAN	1	1	1	402
Thing_Speak	0.93	0.95	0.94	1622
Wipro_bulb	0.8	0.55	0.65	51
Accuracy		0.99		24624
Macro Avg	0.92	0.9	0.91	24624
Weighted Avg	0.99	0.99	0.99	24624



MLP Model:

Class	Precision	Recall	F1-Score	Support
0	0.98	0.98	0.98	1550
1	0.91	0.8	0.85	107
2	1	1	1	18932
3	1	1	1	829
4	0.86	0.86	0.86	7
5	1	0.83	0.91	6
6	1	1	1	400
7	1	1	1	200
8	0.95	0.98	0.97	518
9	1	1	1	402
10	0.98	0.99	0.99	1622
11	0.96	0.84	0.9	51
Accuracy		1		24624
Macro Avg	0.97	0.94	0.95	24624
Weighted Avg	1	1	1	24624


Confusion Matrix: MLP Classifier
True label
Predicted label
0 1,517 7 0 1 0 0 0 1 2 0 22 0
1 0 86 0 0 1 0 0 0 20 0 0 0
2 0 0 18,932 0 0 0 0 0 0 0 0 0
3 2 0 0 827 0 0 0 0 0 0 0 0
4 0 0 0 0 6 0 0 0 1 0 0 0
5 1 0 0 0 0 5 0 0 0 0 0 0
6 0 0 0 0 0 0 400 0 0 0 0 0
7 0 0 0 0 0 0 0 200 0 0 0 0
8 8 0 0 0 0 0 0 0 508 0 1 1
9 2 0 0 0 0 0 0 0 0 400 0 0
10 11 2 0 1 0 0 0 0 2 0 1,605 1
11 0 0 0 0 0 0 2 0 0 0 6 43
0 1 2 3 4 5 6 7 8 9 10 11
17500
15000
12500
10000
7500
5000
2500
0

References

S., B. and Rohini Nagapadma. "RT-IoT2022 ." UCI Machine Learning Repository, 2023, https://doi.org/10.24432/C5P338.

Sharmila, B S and Rohini Nagapadma. “Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset.” Cybersecurity 6 (2023): 1-15.

Sidey-Gibbons, J.A., & Sidey-Gibbons, C.J. (2019). Machine learning in medicine: a practical introduction. BMC Medical Research Methodology, 19.

Wolberg, W., Mangasarian, O., Street, N., & Street, W. (1993). Breast Cancer Wisconsin (Diagnostic) [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5DW2B.