GROUP-9

CHI1002-Security and Privacy Policies for Health Care

ANN TO MONITOR HEALTHCARE NETWORK TRAFFIC AND PREVENT CYBER ATTACKS

Team members:
Swapnendu Banik-22BHI10005
Aarushi Kumar-22BHI10025
Vartika Singh -22BHI10041
Neka S -22BHI10061
Keerti Vijay Ananth -22BHI10189
Mane Vinay -22BHI10207

Submitted to:
Dr. Swagat Kumar Samantaray

1. About the Dataset and our Objective:

The dataset we have used is: https://www.kaggle.com/datasets/faisalmalik/iot-healthcare-security-dataset.

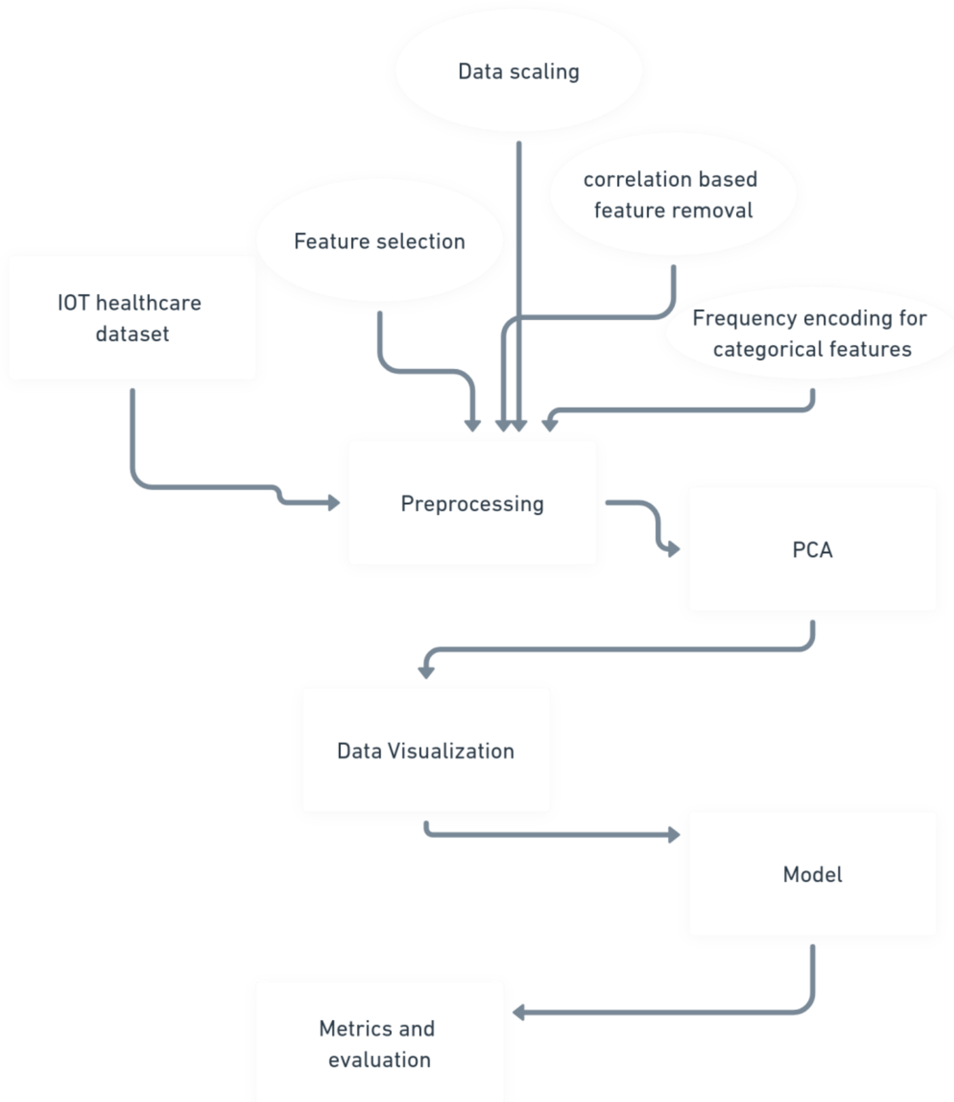
The IoT Healthcare Security Dataset is a collection of network traffic data from IoT medical devices. It includes normal traffic and attack traffic, labeled as 0 and 1, respectively. The dataset provides detailed information about each network packet, such as TCP and MQTT protocol fields, timestamps, and frequency-based features. This dataset can be used to train and evaluate machine learning models for detecting cyberattacks on IoT medical devices.

It contains:

- **Normal traffic** Routine operations, such as a doctor accessing patient records or a nurse updating inventory. [flagged as 0]
- **Malicious traffic** Suspicious activities, like unauthorized access or attempts to inject malware. [flagged as 1]

Our Objective:

The aim of this study is to analyze the IoT Healthcare Security Dataset, a collection of network traffic data from IoT medical devices. We will **preprocess** the data to enhance its quality and **reduce its dimensionality**. Subsequently, we will **visualize** the data to gain insights into its patterns and anomalies. Finally, we will employ machine learning techniques(**ANN**) to **classify** network traffic as either **normal** or **malicious**, enabling the detection of potential cyberattacks on these critical devices.



Project Architecture

2. Preprocessing and Dimensionality Reduction(PCA):

a. **Number of Features(Initially):** We had a net of 50 features excluding the target column, to begin with. That huge number of features wasn't deemed necessary and could have hampered model performance.

b. **Feature Selection:** We started by selecting a subset of columns deemed relevant for the analysis. This step focuses on features that are expected to have a significant impact on the target variable ('label' or 'class'). This helps reduce dimensionality and noise in the data. This step was aided by a domain expert; as we didn't have any, we took help from GPT.

c. **Frequency Encoding for Categorical Features:** We identified categorical features and applied frequency encoding. This technique replaces categorical values with their frequency of occurrence in the dataset. This helps represent categorical data numerically without introducing ordinality. **One Hot Encoding** was not chosen because of the sheer number of unique categorical columns possible.

d. **Correlation-Based Feature Removal:** Highly correlated features can introduce redundancy and instability in the model. The code calculates correlations between features and removes those exceeding a specified threshold (0.75).

e. **Data Scaling:** We used `StandardScaler` to scale the numerical features. This ensures that features have zero mean and unit variance, which can improve model performance, especially for algorithms sensitive to feature scales.

f. **PCA:** PCA is used for dimensionality reduction. It transforms the data into a lower-dimensional space while preserving as much variance as possible. In this case, the data is reduced to two principal components.

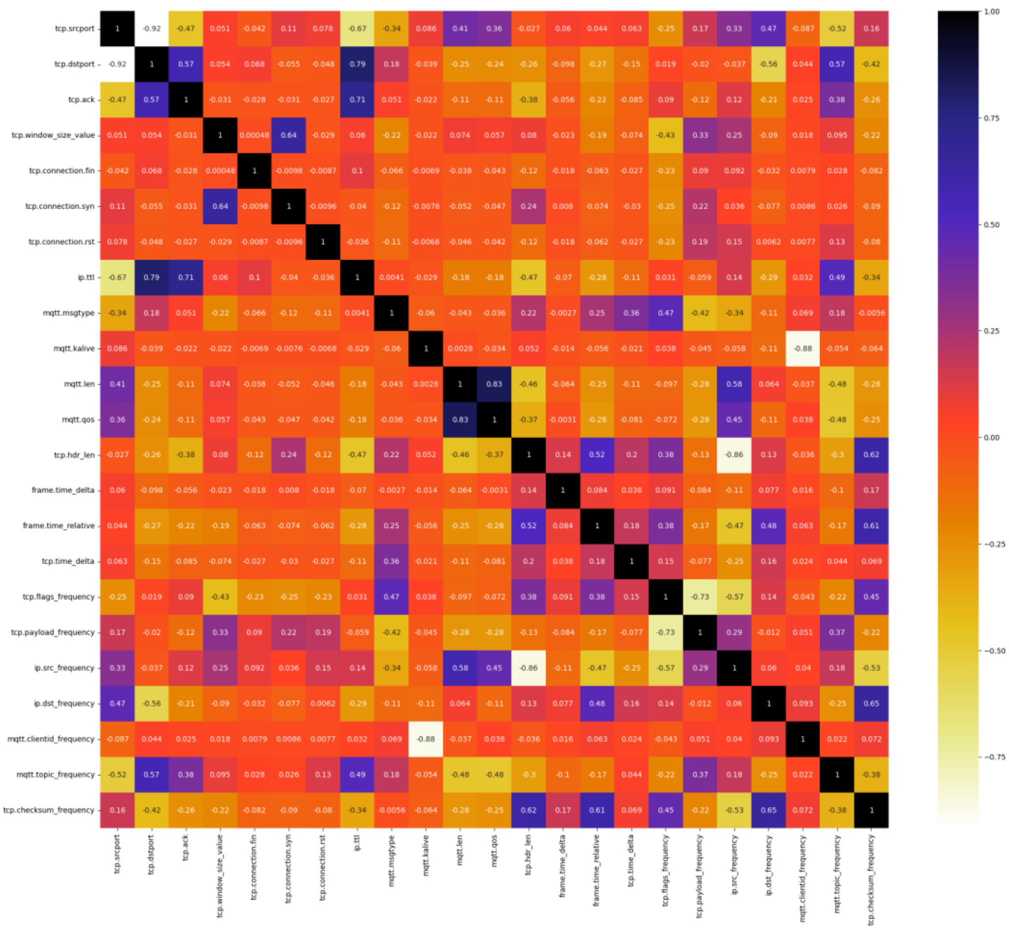
Why PCA:

- **Data Visualization**: PCA is primarily used here for visualizing **high-dimensional data in a lower-dimensional space** (2D in this case). By reducing the data to two principal components, it becomes possible to **plot the data points and visually inspect** their distribution and potential clusters. This is crucial for gaining insights into the data's structure and relationships between features, which might not be apparent in the original high-dimensional space.

- **Distinct Separation**: Observing **distinct separation in the data after PCA** with only two components suggests that the dataset's inherent structure is well-captured by these components. This is a positive indication that the most important **information from the original features is**

retained. This clear separation can be valuable for understanding class distinctions and data patterns.

- **Primary Data for the Model**: As the **PCA-transformed data exhibited clear separation**, it was **chosen as the primary input** for the model. This reduced dimensionality **simplifies the model's complexity** and potentially **improves its performance** by focusing on the most relevant features. Using PCA-transformed data can also reduce the risk of overfitting, especially when dealing with a large number of features in the original data.



Any Columns with beyond 0.75, one of them were dropped, which as visible from the heatmap, were:

```
'Ip.ttl', 'mqtt.qos'
```

3. Data Visualization:

Data visualization is crucial in the early stages of a machine learning project and often more important initially than immediately building a model. Here's what insights we covered--

- **Distinct Class Separation**: The most striking observation is the **clear separation** between the two classes (presumably representing different types of network traffic or attacks) along the first two principal components. This indicates that these **two components capture a significant amount of the variance** that distinguishes the classes.

- **Feature Importance**: The separation along the principal components suggests that the **features contributing most to these components are likely the most important** for distinguishing the classes. Analyzing the loadings of these components can provide further insights into the relative importance of the original features.

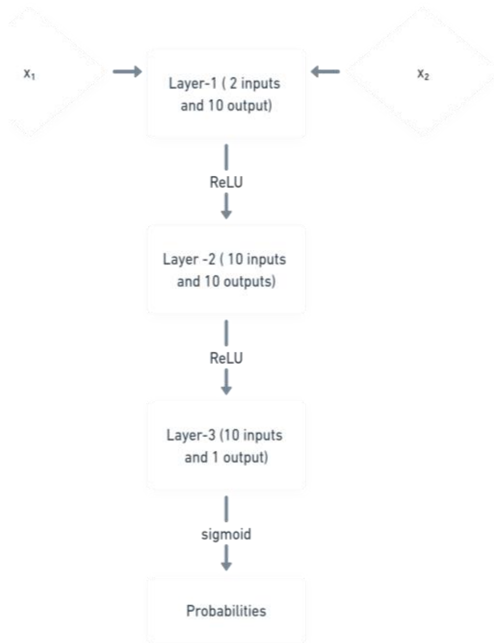
- **Model Suitability**: The clear separation between classes visualized through PCA supports the choice of **using a relatively simple binary classification model**. The data's inherent separability suggests that a model **with a linear decision boundary** (or a **slightly nonlinear one**, as used in the code) could potentially achieve good performance.

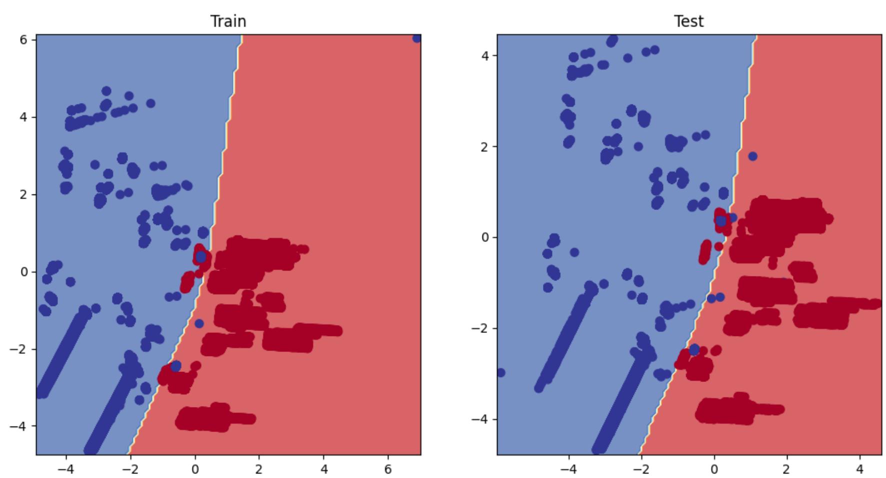
Attack = 1 and Non-Attack = 0

4. Model Architecture:

We developed 2 Models to solve our problem statement, one with lesser complexity and one with more input complexity.

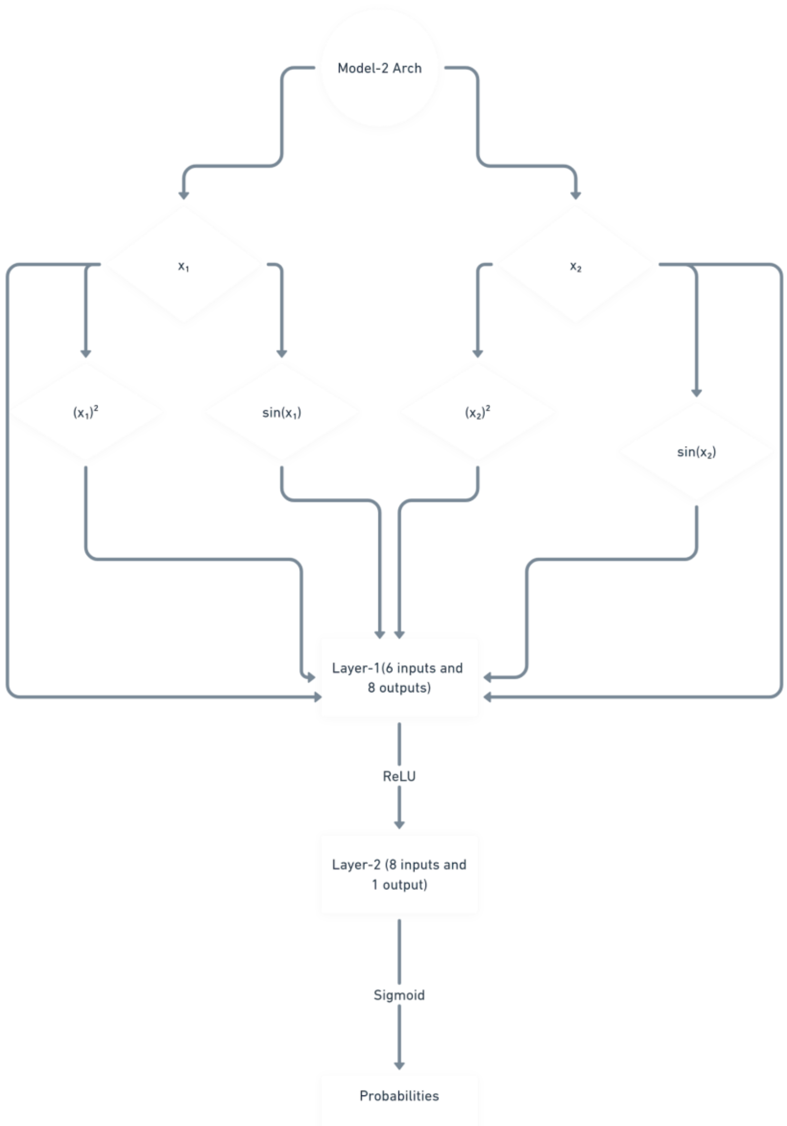
a. **Model 1:** A simple feedforward neural network with **three fully connected layers**. It takes two input features (presumably the principal components from PCA) and passes them through the first layer with 10 neurons, followed by a ReLU activation function. The output is then fed to the second layer, also with 10 neurons and a ReLU activation. Finally, the output of the second layer is passed to the third layer with a single output neuron, producing a raw output (logit) that can be converted to a probability using the sigmoid function. This probability is then rounded to predict the binary class label (0 or 1).

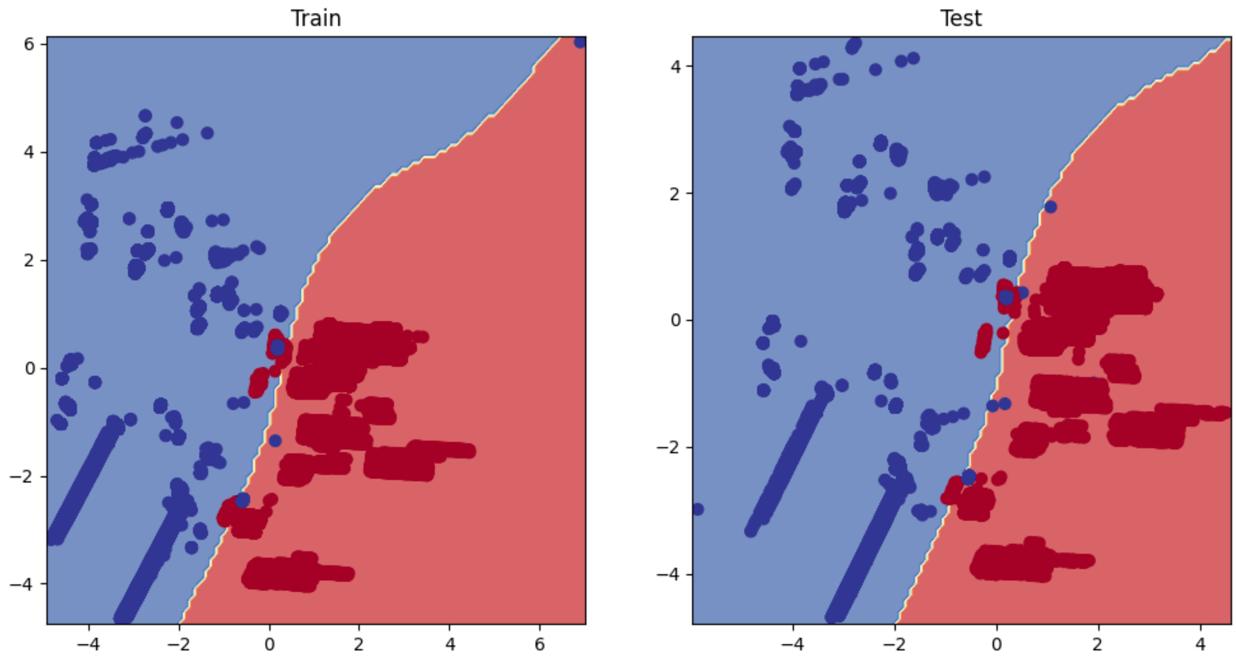




- The **Decision Boundaries of the Initial Model** with respect to Train and Test Data.
- We obtained an **accuracy** of: **99.411%**

b. **Model 2:** A feedforward neural network enhanced with intricate feature engineering. It begins by taking two input features, derived from Principal Component Analysis (PCA), and calculates four additional features: the square of PCA1 (x_1^2), the square of PCA2 (x_2^2), the sine of PCA1 ($sin(x_1)$), and the sine of PCA2 ($sin(x_2)$). These six features (2 original + 4 engineered) are then fed into the first fully connected layer with 6 input features and 8 neurons, followed by a ReLU activation function. The output of this layer is passed to the second fully connected layer with 8 input features and a single output neuron, producing a raw output (logit). This logit is then transformed using the sigmoid function to obtain a probability, which is subsequently rounded to predict the binary class label (0 or 1).





- The **Decision Boundaries of the Advanced Model** with respect to Train and Test Data.
- We obtained an **accuracy** of: **99.417%**

5. Why is accuracy is not the best metric:

- **Imbalanced Datasets:**
  Reasoning: In healthcare security, attack instances (positive cases) are often significantly less frequent than normal traffic (negative cases). This creates an imbalanced dataset.
  Impact: A model can achieve high accuracy by simply predicting the majority class (normal traffic) most of the time. However, it might fail to detect the crucial attack instances.

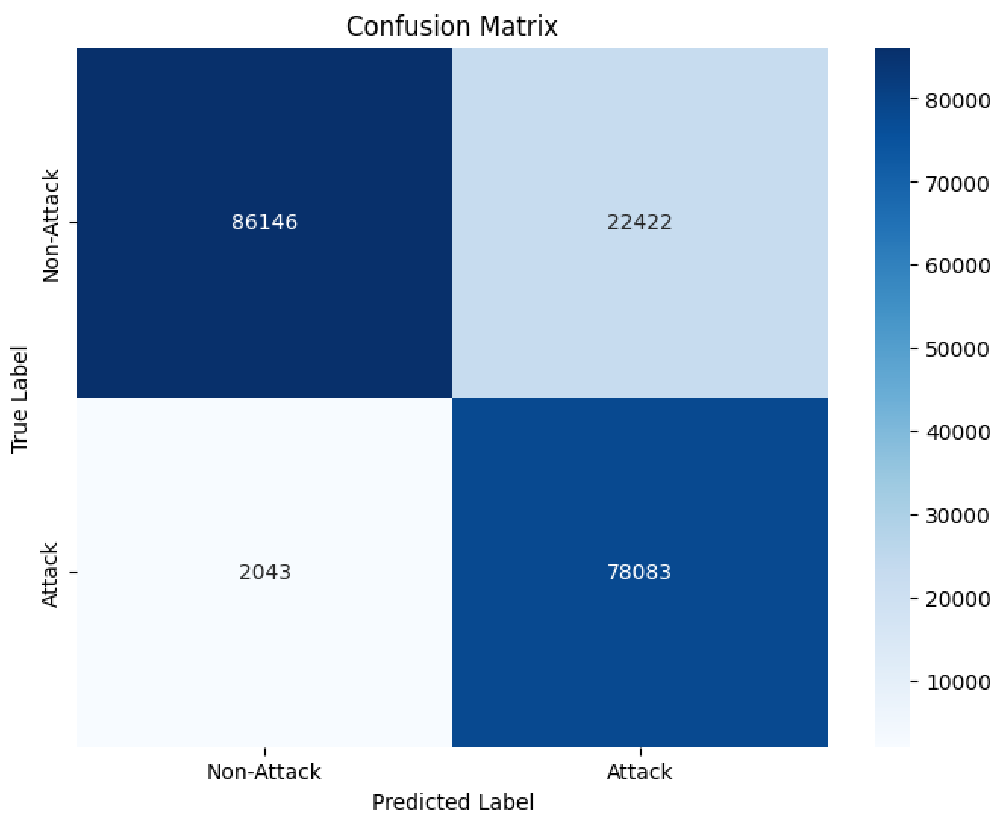
- **Cost of False Negatives:**
  Reasoning: In healthcare security, failing to detect an attack (false negative) can have severe consequences, like data breaches or system disruptions.
  Impact: Accuracy doesn't differentiate between false positives and false negatives. A model with high accuracy might still have a high rate of false negatives, which is unacceptable in this domain.

- **Focus on Attack Detection**:
  Reasoning: The primary goal is to identify attacks accurately, even if it means having some false alarms (false positives).
  Impact: Accuracy treats false positives and false negatives equally, but for healthcare security, false positives are less critical than false negatives.

  Better metrics used:

  - **Precision**: Focuses on minimizing false positives, ensuring that when an attack is predicted, it's likely to be true.
  - **Recall**: Focuses on minimizing false negatives, ensuring that most attacks are identified, even if it leads to some false alarms.
  - **F1-Score**: A balanced measure that considers both precision and recall.
  - **AUC (Area Under the ROC Curve)**: A robust metric for imbalanced datasets, representing the model's ability to distinguish between classes.

6. Metrics and Reports: [Model 2 was chosen for better Decision Boundary]

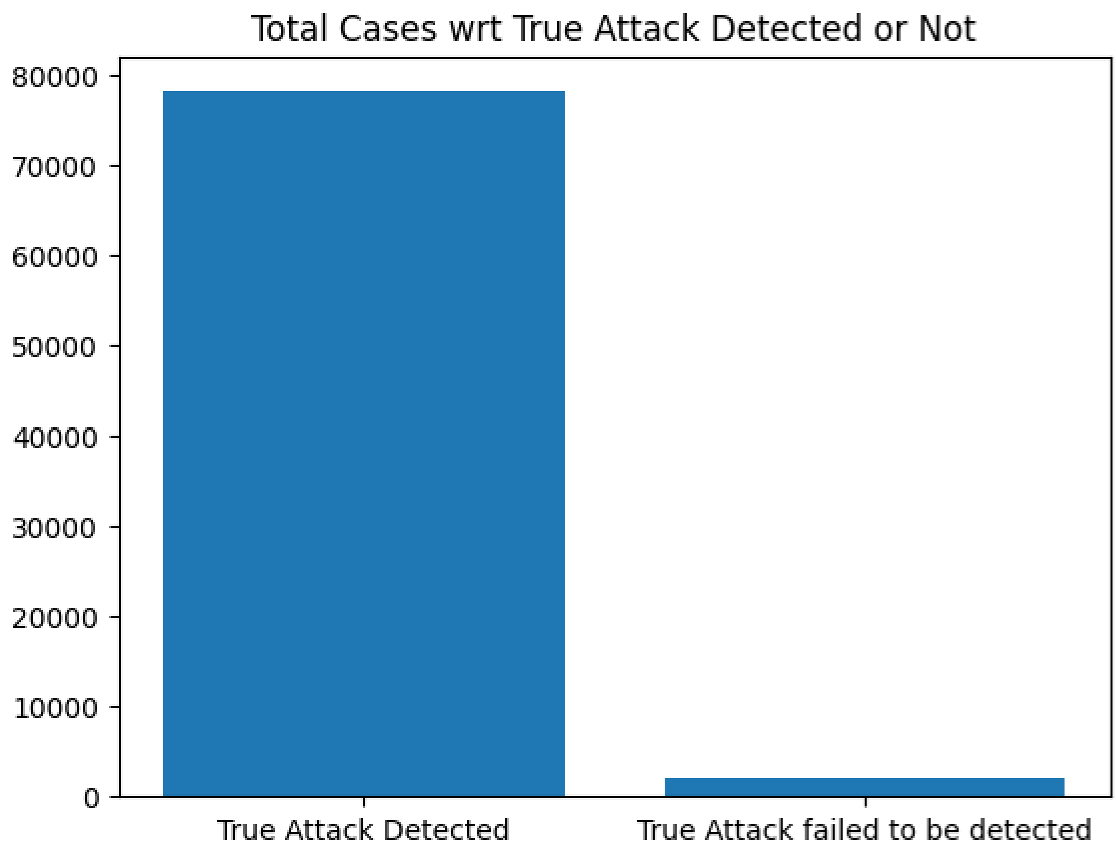


Detailed Metrics:

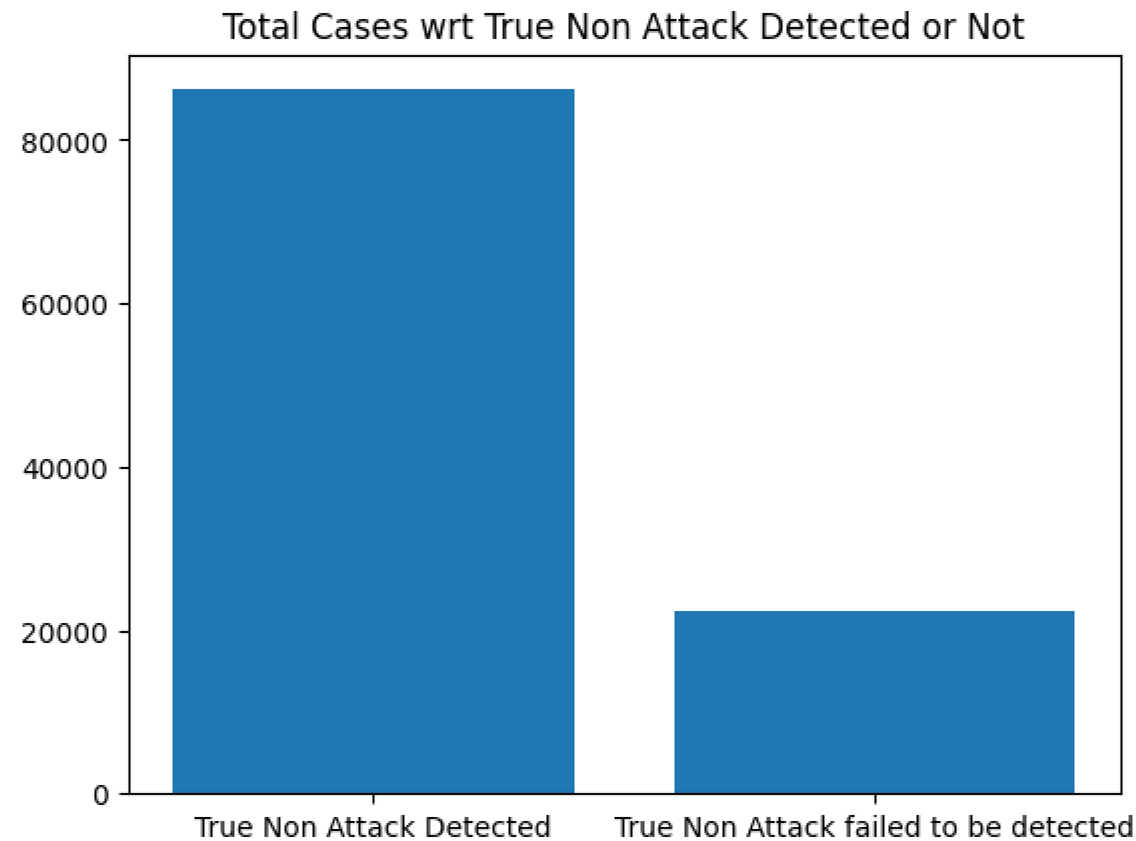
	Precision	Recall	F1-Score	Support
Non-Attack	0.9768	0.7935	0.8757	108568.0
Attack	0.7769	0.9745	0.8646	80126.0

Additional Metrics:

Specificity:	0.7935
False Positive Rate:	0.2065
False Negative Rate:	0.0255

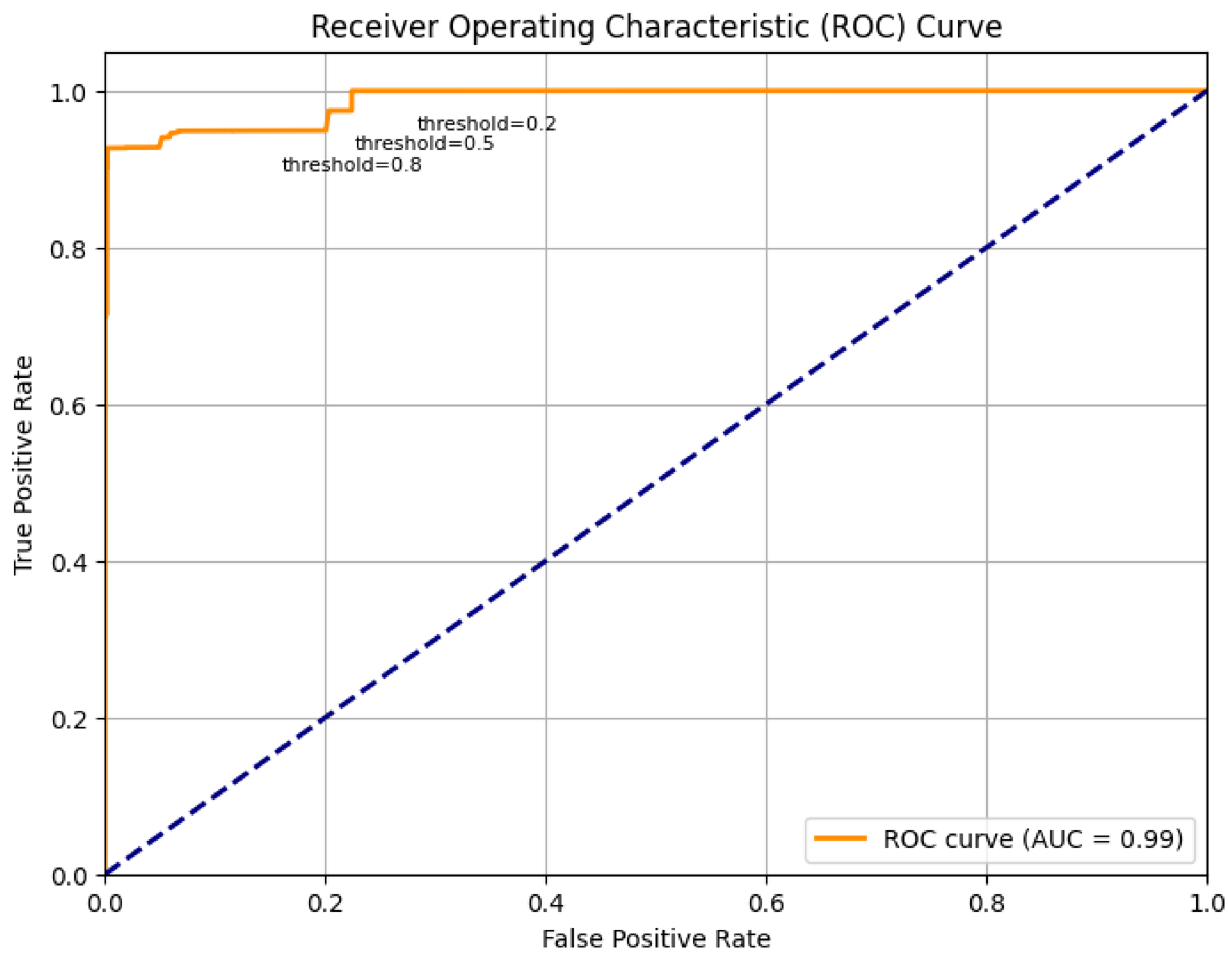


The Model is extremely good at detecting True-Positives aka the True Attack cases, even though the support for True Attack cases is barely 42.4% of the actual dataset.



The Model fails a considerable amount of time to detect False-Positives, even though the dataset is 52.7% of this case. But as Healthcare data, we are generally not concerned about this.

Detecting True-Positives and avoiding Attack Cases takes priority even if that means sacrificing a bit on cases of False-Positives; this happens as in data visualization we saw some overlapping near the decision edge. The model has a False Positive rate of 0.025, meaning it detects a True Attack 97.5% times!



The Model has an AUC Score of 0.99; the model demonstrates excellent discrimination ability, as indicated by the high AUC score. This suggests the model is effectively separating attacks from non-attacks across a range of thresholds.

7. Conclusion:

- The model demonstrates promising performance in detecting true attacks, as evidenced by a high AUC score of [0.99]. This score indicates the model's ability to effectively distinguish between attack and non-attack traffic across various thresholds.

- The model achieves a sensitivity/recall of [0.97], suggesting it correctly identifies [97%] of actual attacks. This highlights the model's capability in minimizing false negatives, which is crucial in security applications where missing an attack could have significant consequences.

- **Overall, this model provides a valuable tool for detecting TCP-based attacks in healthcare security settings. Its high AUC score and sensitivity/recall demonstrate its effectiveness in identifying true attacks, making it a promising solution for mitigating security risks in this critical domain.**