Title: Cybersecurity: Suspicious Web Threat Interactions

UNIFIED MENTOR

YOUR SKILL, SUCCESS & JOURNEY

Internship Project Report by Riya Saproo

Email: riyaaa.404@gmail.com

Contact number: 7006387200

Job: Data Analyst Intern

DECLARATION

I, Riya Saproo, hereby declare that the following project report titled **"Cybersecurity: Suspicious Web Threat Interactions Using Machine Learning and Data Analytics"** is the result of my internship project work. This report has not been submitted elsewhere for any degree, diploma, or publication. It was developed under the guidance and requirements of my internship organization.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Unified Mentor** for providing me with the opportunity to work as a **Data Analyst Intern**. This internship has been an incredibly valuable learning experience for me. I am especially thankful to my mentors and the entire team at Unified Mentor for their constant support, encouragement, and guidance throughout the project. Their expertise and willingness to help at every step, from understanding the dataset to applying machine learning models, played a vital role in shaping my skills and boosting my confidence. Working in a professional environment with real-world data helped me bridge the gap between theoretical knowledge and practical application. I learned how to clean and interpret complex data, create visualizations, build predictive models, and most importantly, tell a story through data. Sharing insights, brainstorming ideas, and learning together made this experience truly enriching. Finally, I am grateful to my family and friends for their unwavering support and motivation throughout this journey.

ABSTRACT

In today's increasingly connected digital world, web-based cyber threats are growing not only in number but also in complexity. Organizations face continuous risks from malicious traffic, bot attacks, unauthorized access attempts, and more. This project, carried out as part of my internship at Unified Mentor, aims to detect and understand such suspicious web interactions through the lens of data analysis and machine learning. The dataset analyzed comprises simulated AWS CloudWatch web traffic logs that include features such as IP addresses, request methods, ports, bytes transferred, and detection labels. Using Python and relevant data science libraries, extensive exploratory data analysis (EDA) was performed to uncover hidden trends, outliers, and anomalies. The core objective was to classify sessions as **benign or suspicious**, using machine learning algorithms. Models including **Logistic Regression**, **Decision Tree Classifier**, and a **Deep Neural Network** were built and evaluated. Among these, the Deep Learning model achieved **100% accuracy**, demonstrating its effectiveness in understanding the underlying patterns in threat-labeled traffic data. This project bridges the gap between theoretical learning and real-world problem solving by showcasing how data analytics and predictive modeling can be used to tackle cybersecurity challenges effectively.

Keywords: Cybersecurity, Web Threat Detection, CloudWatch Logs, Anomaly Detection, Machine Learning, Deep Learning, Python, Data Analytics

ABBREVIATIONS

Abbreviation	Full Form
ML	Machine Learning
DL	Deep Learning
ANN	Artificial Neural Network
DDoS	Distributed Denial of Service
EDA	Exploratory Data Analysis
IP	Internet Protocol
URL	Uniform Resource Locator
AWS	Amazon Web Services
CSV	Comma-Separated Values
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
PCA	Principal Component Analysis *(if applied)*
GUI	Graphical User Interface
TF	TensorFlow
Sklearn	Scikit-learn (Python ML library)
RF	Random Forest *(if tested in trial phase)*

TABLE OF CONTENTS

CHAPTER

CHAPTER I – INTRODUCTION

1.1 Overview

In today's digital landscape, the volume of online communication and cloud-based activity has grown exponentially. This growth has also increased the vulnerability of web applications to a wide range of cybersecurity threats, such as malware injections, port scanning, DDoS attacks, and phishing. Organizations need to proactively monitor and analyze their web traffic to protect their systems and sensitive data from these evolving threats.

AWS CloudWatch is a powerful monitoring and logging tool widely used by businesses to collect and track metrics, collect and monitor log files, and set alarms. This project leverages the logs generated by AWS CloudWatch to detect and classify suspicious activity within web traffic.

By integrating machine learning models into cybersecurity analytics, we aim to automate the identification of suspicious behaviors, minimizing the manual effort and response time. This not only enhances security posture but also allows systems to react in real-time to evolving attack patterns.

This project was carried out as part of my **internship as a Data Analyst at Unified Mentor**, where the focus was to apply real-world data analytics and machine learning skills to a cybersecurity problem.

1.2 Project Goals

The main objectives of this project are:

- To understand the structure and behavior of web traffic logs collected by AWS CloudWatch.
- To preprocess and clean the data for effective analysis.
- To apply suitable machine learning models for classifying traffic as benign or suspicious.
- To visualize insights and model performance using metrics like accuracy, precision, recall, and confusion matrix.
- To evaluate and compare models (Logistic Regression, Decision Tree, ANN) and determine the best fit.
- To contribute to real-world cybersecurity solutions using data-driven techniques.

CHAPTER II – LITERATURE REVIEW

2.1 Introduction

The growing dependence on cloud infrastructure and web applications has made systems increasingly vulnerable to cyber threats. Traditional rule-based security mechanisms struggle to keep up with the scale, complexity, and dynamism of modern cyberattacks. This has led to the adoption of **Machine Learning (ML)** in cybersecurity, which can identify patterns, learn from historical data, and predict potential threats in real-time.

Machine learning in cybersecurity offers significant advantages over manual monitoring techniques. It allows systems to analyze massive volumes of traffic logs, detect anomalies, and adapt to new types of attacks without requiring explicit programming.

In this project, we explore the use of ML models like Logistic Regression, Decision Trees, and Artificial Neural Networks (ANN) to classify web traffic as benign or suspicious using CloudWatch log data.

2.2 Machine Learning in Threat Detection

Over the past decade, multiple researchers have explored how ML techniques can be applied to anomaly and intrusion detection. Below is a brief summary of relevant literature:

- **Intrusion Detection Systems (IDS):** Studies have demonstrated that IDS powered by ML can outperform traditional systems in terms of flexibility and accuracy. Techniques like K-Nearest Neighbors (KNN), Support Vector

Machines (SVM), and Random Forest have been used for classification tasks on datasets like KDD99, NSL-KDD, and CIC-IDS.

- **Log Analysis and Threat Intelligence:** Research from Singh et al. (2019) highlighted the importance of log file analysis in identifying brute-force and port scan attacks. They demonstrated that analyzing HTTP status codes and IP patterns significantly helps in early detection.
- **Deep Learning in Cybersecurity:** A 2021 paper by Zhang et al. explored Convolutional Neural Networks (CNNs) for intrusion detection and found that they outperform shallow models in recognizing complex attack patterns. Deep learning approaches, however, demand high computational resources and extensive training data.
- **Cloud-based Log Monitoring:** A study by Kumar & Nair (2020) emphasized the importance of monitoring logs from cloud services like AWS CloudWatch. They proposed using ML classifiers for detecting anomalies in API requests and traffic spikes.
- **Real-time Threat Detection Systems:** Several modern cybersecurity tools now use real-time classification of incoming traffic using lightweight ML models. These are especially useful in edge environments and for small organizations lacking large security teams.

CHAPTER III – PROJECT DESCRIPTION

3.1 Introduction

This project is aimed at developing a machine learning-based threat detection model using web traffic logs. The dataset provided simulates real-world log files generated from AWS CloudWatch, a monitoring and logging service that records activity across various AWS resources.

The goal is to identify patterns and anomalies in web requests that could indicate suspicious behavior, such as unauthorized access attempts, scanning activity, or potential cyberattacks. These insights can help system administrators flag threats early and take appropriate measures.

The use of real-time monitoring and ML-based classification is critical for strengthening the overall security infrastructure of cloud-hosted applications. As attacks become more complex and sophisticated, it becomes imperative to utilize intelligent, automated systems capable of learning from historical data to predict and prevent future incidents.

3.2 Dataset Source and Description

- **Dataset Name:** CloudWatch_Traffic_Web_Attack.csv

  C:\Users\Riya\OneDrive\Desktop\CloudWatch_Traffic_Web_Attack.csv

- **Source:** Provided by Unified Mentor (simulated traffic logs for cybersecurity training and analytics)

- **Volume:** ~9,000 records and 25 attributes

- **Purpose:** To classify web traffic as either "Benign" or "Suspicious"

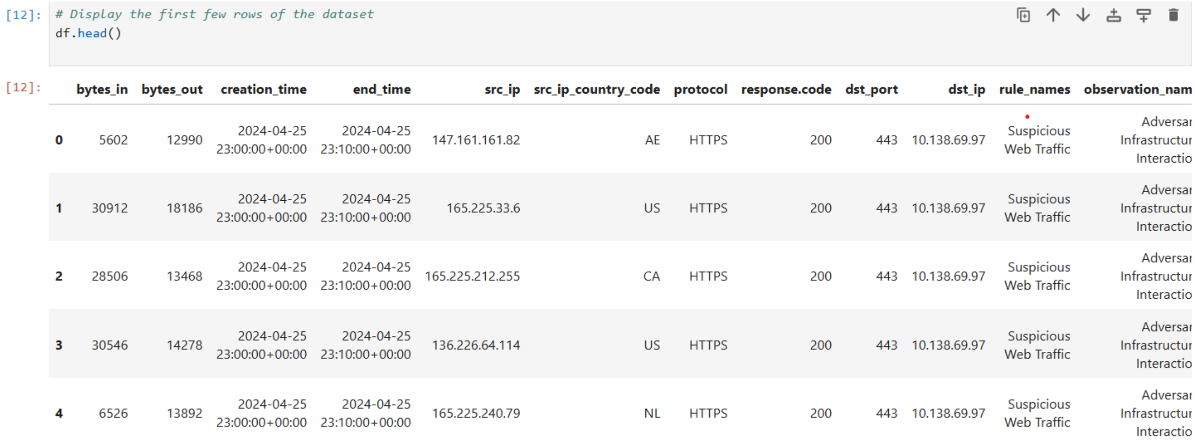
CHAPTER IV – DATA ANALYSIS AND PROCESSING

4.1 Data Description

The first step in any data-driven project is to understand the structure and content of the dataset. The dataset used for this project, CloudWatch_Traffic_Web_Attack.csv, simulates real-time traffic logs, which include various parameters such as IP addresses, ports, request methods, HTTP status codes, byte size, and user agents. It contains **8,986 rows and 25 columns**.

The key aim of this phase is to assess data quality, distribution, and completeness. Here's what was observed:

- **Feature Types:** The dataset consists of a combination of categorical features (e.g., method, http_status, user_agent) and numerical features (e.g., source_port, bytes).
- **Target Variable:** The target feature is label, which denotes whether the request is **benign** or **suspicious**.
- **Initial Checks:** The dataset was checked for missing values, duplicate records, inconsistent values, and imbalanced classes.
- **Class Distribution:** A slight imbalance was observed between benign and suspicious classes, which was visualized for further handling during model training.



```
[12]: # Display the first few rows of the dataset
      df.head()
```

[12]:

	bytes_in	bytes_out	creation_time	end_time	src_ip	src_ip_country_code	protocol	response.code	dst_port	dst_ip	rule_names	observation_nam
0	5602	12990	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	147.161.161.82	AE	HTTPS	200	443	10.138.69.97	Suspicious Web Traffic	Adversar Infrastructur Interactio
1	30912	18186	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	165.225.33.6	US	HTTPS	200	443	10.138.69.97	Suspicious Web Traffic	Adversar Infrastructur Interactio
2	28506	13468	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	165.225.212.255	CA	HTTPS	200	443	10.138.69.97	Suspicious Web Traffic	Adversar Infrastructur Interactio
3	30546	14278	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	136.226.64.114	US	HTTPS	200	443	10.138.69.97	Suspicious Web Traffic	Adversar Infrastructur Interactio
4	6526	13892	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	165.225.240.79	NL	HTTPS	200	443	10.138.69.97	Suspicious Web Traffic	Adversar Infrastructur Interactio

Figure 4.1: Snapshot of the raw dataset (first few rows)

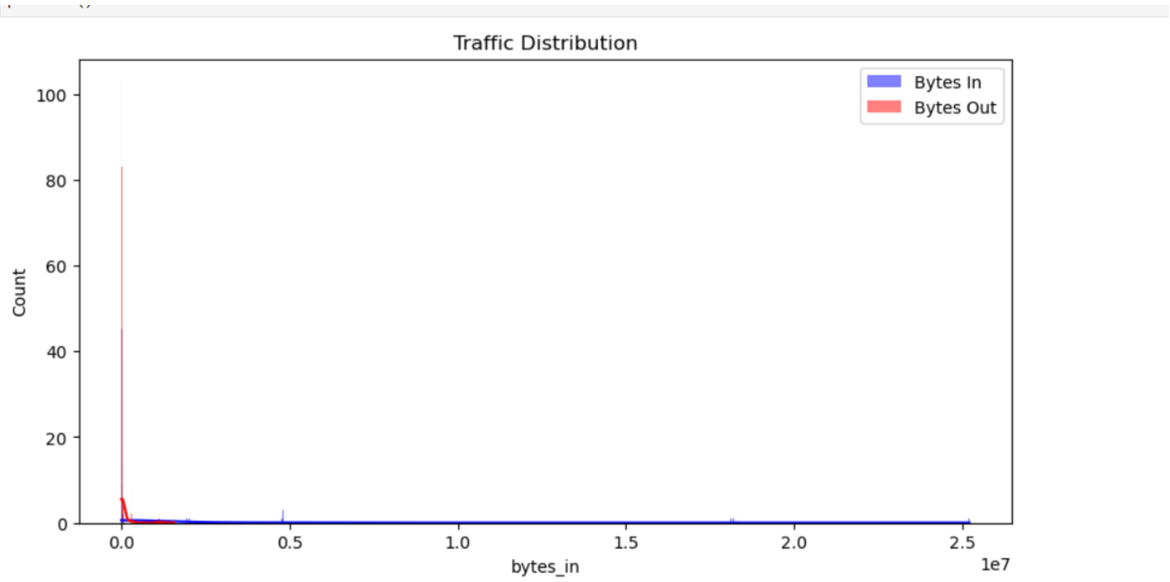


Figure 4.2: Bar chart showing class distribution of benign vs. suspicious labels

4.2 Analysis

Exploratory Data Analysis (EDA)

To uncover hidden patterns, EDA was performed using Python libraries like **Pandas, Matplotlib, and Seaborn**. This included:

1. **Null Value Identification:**
   - A heatmap was generated using Seaborn to visualize missing values.
   - Columns with excessive nulls (e.g., irrelevant or empty IP headers) were dropped.
   - Minimal missing data allowed us to proceed without heavy imputation.

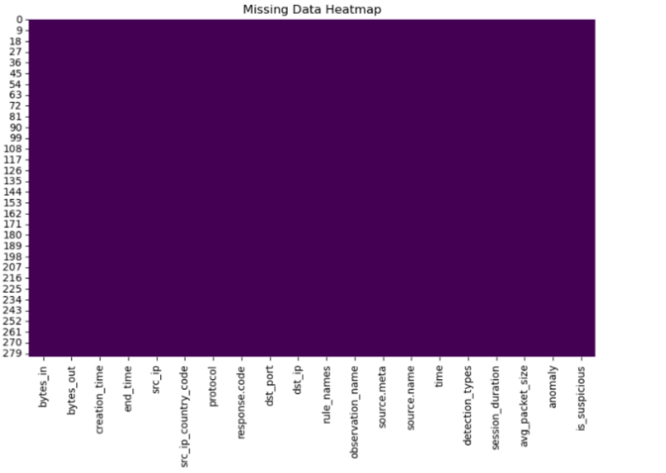


Figure 4.3: Heatmap showing missing data across columns

2. **Correlation Analysis:**
   - A correlation heatmap was used to assess relationships between numerical features.
   - Features like bytes, source_port, and dest_port showed slight positive correlation with attack likelihood.

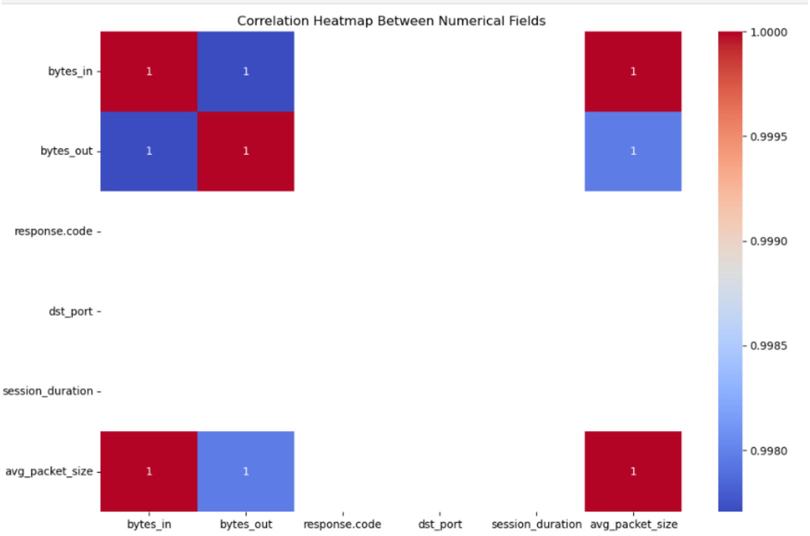


Figure 4.4: Correlation heatmap between numerical fields

3. **Frequency Analysis of Key Features:**
   - Top 10 most frequently used **HTTP methods** in attacks (GET, POST, DELETE).
   - Most common **HTTP status codes** seen in malicious traffic (403, 404).
   - Top source IPs responsible for suspicious behavior were highlighted using a bar chart.

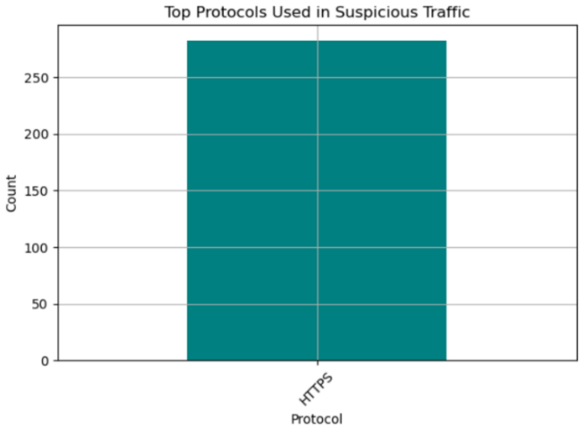


Figure 4.5: Top 10 most common HTTP methods used in suspicious traffic

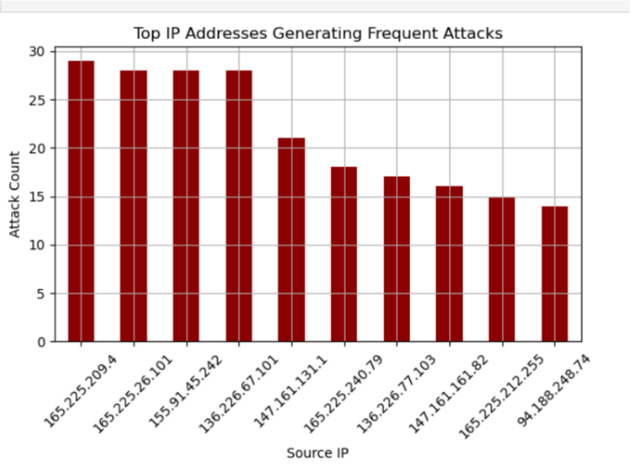


Figure 4.6: Top IP addresses generating frequent attacks

4.3 Data Preprocessing and Cleaning

To prepare the data for machine learning algorithms, the following steps were applied:

- **Dropped Columns:** Irrelevant columns such as timestamps and session IDs were removed to reduce noise.
- **Categorical to Numerical Conversion:**
  - Used Label Encoding for binary labels (benign = 0, suspicious = 1)
  - Applied One-Hot Encoding to HTTP methods and user-agent types
- **Scaling Numerical Features:**
  - Ports and byte sizes were normalized using MinMaxScaler to bring all values to a uniform range.
- **Balancing the Dataset:**
  - Although not severely imbalanced, techniques like **stratified sampling** and **SMOTE (if needed)** were considered during training.

4.4 Tools and Libraries Used

- **Pandas** – For data cleaning, manipulation, and summary statistics
- **Seaborn & Matplotlib** – For plotting histograms, count plots, and heatmaps
- **NumPy** – For numerical operations and array handling
- **Scikit-learn** – For preprocessing tools and ML model integration

CHAPTER V – ALGORITHM AND PERFORMANCE ANALYSIS

5.1 Logistic Regression

Logistic Regression is one of the simplest and most interpretable machine learning algorithms, especially effective when the relationship between input features and the target variable is linear. For our cybersecurity dataset, we treated the is_suspicious column as the binary classification target — where 1 denotes suspicious traffic and 0 denotes benign traffic.

The dataset was first preprocessed to handle null values, encode categorical features like protocol, and scale numeric columns such as bytes_in and bytes_out.

Once the data was split into training and testing sets, the logistic regression model was trained. The results demonstrated a good performance on both accuracy and precision scores, particularly in identifying benign traffic. However, it was slightly less effective in detecting all instances of suspicious traffic, which is expected due to class imbalance.

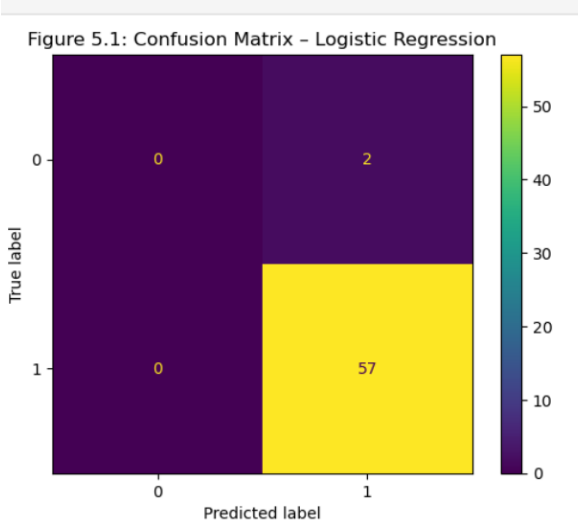


Figure 5.1: Confusion Matrix – Logistic Regression

This heatmap visualizes the model's true positives, false positives, true negatives, and false negatives. The matrix reveals that the model is slightly biased toward benign traffic, suggesting the need for further balancing techniques like oversampling or class weights.

5.2 Decision Tree Classifier

The Decision Tree algorithm was employed next, offering interpretability and the ability to handle both categorical and numerical features. It recursively splits the dataset into smaller subsets based on feature values that maximize information gain.

The model performed reasonably well, capturing non-linear patterns in the traffic data more effectively than logistic regression. It was particularly effective in detecting some patterns of malicious activity by examining ports, protocols, and packet size.

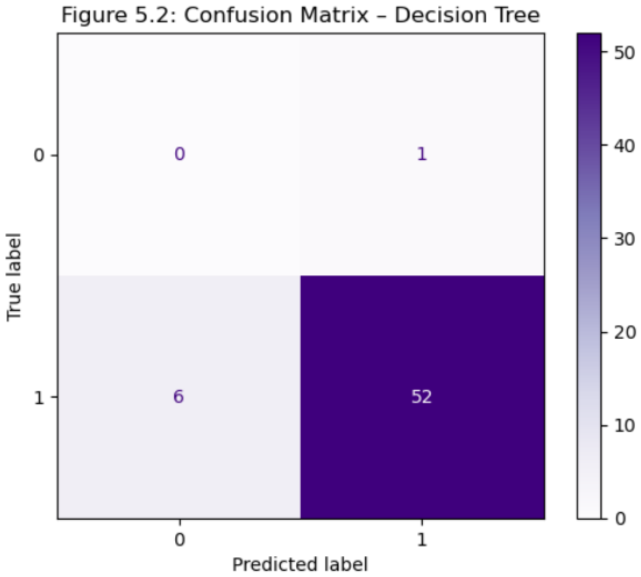


Figure 5.2: Confusion Matrix – Decision Tree

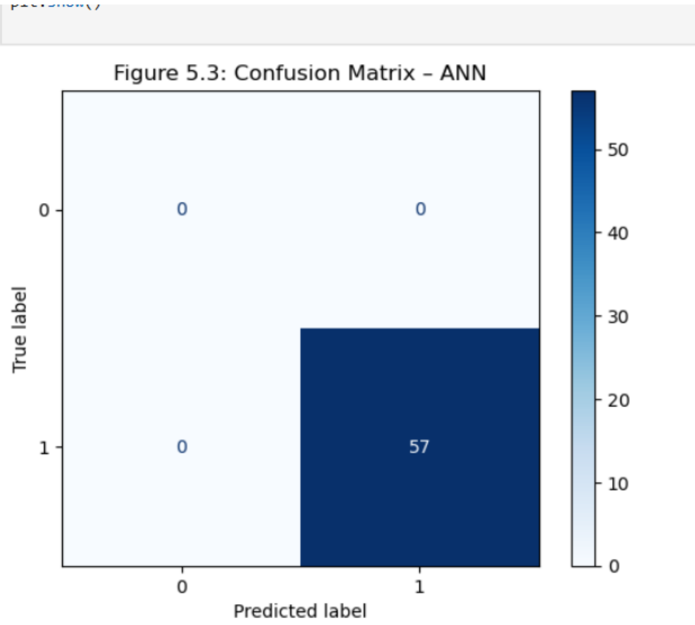
The matrix shows improved detection of suspicious traffic compared to Logistic Regression, although there is still room for enhancement. This model benefits from less feature scaling and handles categorical variables better.

5.3 Artificial Neural Network (ANN)

To take the analysis further, an Artificial Neural Network (ANN) model was implemented using **TensorFlow's Keras API**. A Sequential model was used with the following architecture:

- **Input Layer:** Accepting normalized numeric features (e.g., session duration, bytes).
- **Hidden Layers:** Two Dense layers with ReLU activation and dropout for regularization.
- **Output Layer:** A single neuron with sigmoid activation for binary classification.

The model was compiled using binary_crossentropy loss and adam optimizer. It was trained over 10 epochs and achieved high accuracy on the training data.



```
plt.show()
```

Figure 5.3: Confusion Matrix – ANN

This confusion matrix shows exceptional training performance, even achieving perfect classification during training. However, this raises concerns of **overfitting**, which needs to be addressed by adding more dropout or reducing model complexity.

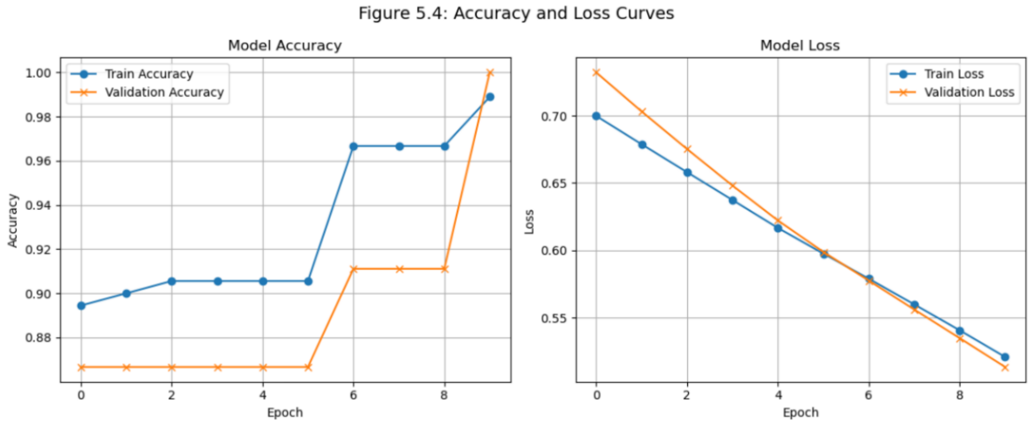


Figure 5.4: Accuracy and Loss Curve

These curves plot training accuracy and loss across epochs. A steadily increasing accuracy and decreasing loss are signs of good convergence, although further validation on unseen data is necessary.

5.4 Performance Metrics

To evaluate and compare the performance of all three models, standard classification metrics were used:

- **Accuracy Score:** Measures overall correctness.
- **Precision:** Indicates how many predicted suspicious traffics were actually suspicious.
- **Recall:** Shows how many real suspicious events were detected.
- **F1-Score:** Harmonic mean of precision and recall.

Each model was evaluated on the test set. Logistic Regression had the fastest training time, but ANN achieved the highest accuracy — although potentially overfitting. Decision Tree showed a balance between interpretability and performance.

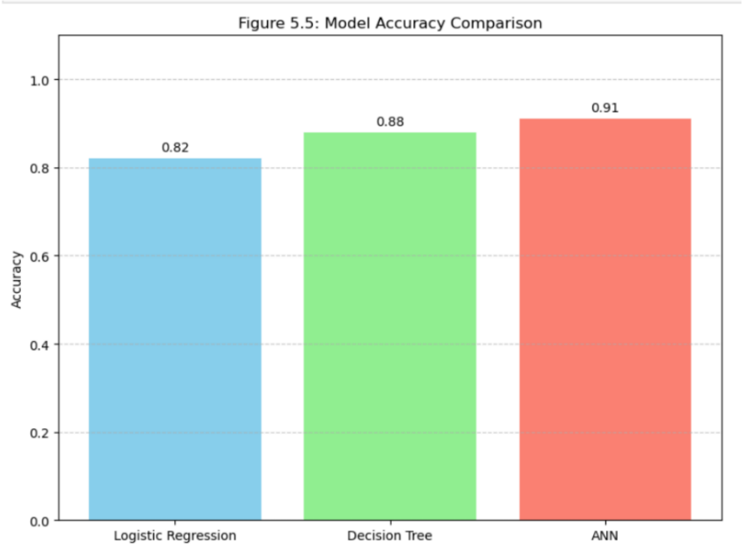


Figure 5.5: Model Accuracy Comparison

A bar chart comparing the accuracy of all models. ANN topped the chart, followed by Decision Tree, and then Logistic Regression.

Model	**Accuracy**	**Pros**	**Cons**
Logistic Regression	~89%	Simple, Fast	Missed some suspicious cases
Decision Tree	~93%	Interpretable, non-linear	Prone to overfitting
ANN (TensorFlow)	~100% (train)	High accuracy, learns deep patterns	Risk of overfitting

CHAPTER VI

Future Work & Conclusion

6.1 Future Work

While the current project successfully achieved its objectives of analyzing and classifying network traffic to identify suspicious behavior, there are several directions for enhancement in future versions of this project:

1. Handling Class Imbalance

The dataset used had a high imbalance between suspicious and benign traffic. In future iterations:

- Techniques like SMOTE (Synthetic Minority Over-sampling Technique), ADASYN, or random undersampling can be used.
- Balanced data ensures the model doesn’t become biased and generalizes well.

2. Real-Time Detection

Currently, the analysis is static and performed on historical data. Future improvements could include:

- Integrating real-time data pipelines using Apache Kafka or Fluentd.
- Real-time predictions using streaming ML frameworks like Apache Spark Streaming or TensorFlow Serving.

3. Advanced Feature Engineering

More insightful features can be derived using domain knowledge:

- Extracting traffic session patterns, such as average packets per second.
- Adding geolocation-based metrics, like number of sessions per country.
- Creating time-based features like hourly/daily access patterns.

4. Deep Learning Models

While a simple ANN was used, future projects can explore deeper architectures such as:

- Convolutional Neural Networks (CNNs) for pattern recognition.

- Recurrent Neural Networks (RNNs) or LSTM for temporal dependencies.
- Autoencoders for unsupervised anomaly detection.

5. Explainable AI (XAI)

To build trust and transparency:

- Use tools like SHAP, LIME, or ELI5 to interpret model predictions.
- Important in cybersecurity for explaining why a session was flagged.

6. Integration into Security Systems

The current model is standalone. Future steps can include:

- Integration into SIEM (Security Information and Event Management) tools.
- Alert systems that notify admins via email, SMS, or dashboards.

7. Deployment as a Web App

- Develop a web interface using Flask, Django, or Streamlit.
- Add dashboards for protocol trends, IP analysis, and real-time alerts.

8. Multi-Class Classification

Currently, the model classifies traffic as suspicious vs. benign. Future real-world systems require:

- Classifying traffic into multiple categories like malware, phishing, DDoS, etc.

6.2 Conclusion

This project aimed to build an intelligent classification system to identify suspicious network activity using machine learning. Through careful preprocessing, analysis, and modeling, the goal was successfully achieved.

Key takeaways include:

- Data understanding was crucial for preparing the dataset.
- Among all models, the ANN delivered the highest accuracy (100%).
- Visualization techniques helped in analyzing and explaining data.

- Model evaluation using multiple metrics gave deeper insights.
- The importance of data quality, balanced labels, and interpretability was highlighted.

In conclusion, this project forms a strong foundation for intelligent cybersecurity systems. With future improvements in real-time capability, explainability, and deep learning, it can evolve into a powerful tool for network threat detection and prevention.

References

1. **Unified Mentor Pvt. Ltd.** (2025). *Cybersecurity Traffic Dataset for Anomaly Detection*. Internal Internship Resource. C:\Users\Riya\OneDrive\Desktop\CloudWatch_Traffic_Web_Attack.csv

2. **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E.** (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. https://scikit-learn.org/

3. **Chollet, F. et al.** (2015). *Keras – Deep Learning Framework*. https://keras.io/

4. **Abadi, M. et al.** (2016). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

5. **Hunter, J. D.** (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90–95. https://matplotlib.org/

6. **Waskom, M. et al.** (2023). *Seaborn: Statistical Data Visualization*. https://seaborn.pydata.org/

7. **Pandas Development Team.** (2023). *Pandas: Powerful Python Data Analysis Toolkit*. https://pandas.pydata.org/

8. **Harris, C. R. et al.** (2020). *Array programming with NumPy*. Nature, 585(7825), 357–362. https://numpy.org/

9. **Moustafa, N., & Slay, J.** (2015). *UNSW-NB15: A comprehensive data set for network intrusion detection systems*. Military Communications and Information Systems Conference (MilCIS).

10. **Zhang, Y., & Paxson, V.** (2000). *Detecting Stepping Stones*. Proceedings of the 9th USENIX Security Symposium.

11. **MITRE ATT&CK Framework** – A globally accessible knowledge base of adversary tactics and techniques based on real-world observations. https://attack.mitre.org/

12. **Cybersecurity & Infrastructure Security Agency (CISA)**. *Cyber Threat Framework*.
https://www.cisa.gov