Network Intrusion Detection using AI and ML Techniques

Supragya Sharma
School of Computer Science Engineering & Technology, Bennett University, Greater Noida - India
rishu.supragya@gmail.com

Sahil Gupta
School of Computer Science Engineering & Technology, Bennett University, Greater Noida - India
sahil.neon09@gmail.com

Sajal Jain
School of Computer Science Engineering & Technology, Bennett University, Greater Noida - India
sajaljain30345@gmail.com

Abstract: The rapid expansion of interconnected systems and the increasing rate of events of cyber threats all around the world have sufficiently urged the need for an extraordinarily strong Network Intrusion Detection System (NIDS). This paper studies the possible application of advanced Artificial Intelligence and Machine Learning techniques in providing a means to develop efficient and highly accurate Network Intrusion Detection System (NIDS). Such high-profile algorithms include Logistic Regression, Random Forest, Feedforward Neural Networks, K-Nearest Neighbour, and Long Short-term Memory. The techniques will be examined based on their ability to very accurately and efficiently detect and classify network intrusions. A comparative study is hence conducted, which evaluates the models based on metrics such as accuracy, recall, precision, and efficiency in processing power, hence their respective advantages and weaknesses. Results intend to assist in the development of fully intelligent next generation NIDS that will indefatigably stand against evolving cyber threats.

Keywords: Artificial Intelligence, machine learning, deep learning, network intrusion detection, dataset, cybersecurity.

I. Introduction

The central focus of concern in this day and age of digital transformation is the security of network systems. Cyberattacks that range from simple malware infection to complex Distributed Denial of Service (DDoS) attacks can almost cripple the entire functioning of any organization or individual. Old traditional intrusion detection methodologies are static-rule or signature-based and hence do not adequately respond to the changing and dynamic spectrum of modern cyber threats. Machine Learning (ML) and Artificial Intelligence (AI) are believed to provide very promising solutions as they enable a certain system to learn different patterns to detect anomalies independently. The present research is concerned with the application of five different techniques of AI and ML to help design a new and more powerful Network Intrusion Detection Systems (NIDS): Logistic Regression, Random Forest, Deep Neural Networks (DNN), K-Nearest Neighbours (KNN), and Long Short-term Memory (LSTM).

We first present the conceptual framework for each of the methodologies pertaining to their principles and how they are well suited to intrusion detection-in, what benefits they would provide for different use cases. Then we evaluate their performance in a benchmark dataset on accuracy, sensitivity, and computational requirements.

The research is mainly targeted at enhancing the already existing knowledge of security against such cyber-attacks by making contributions to the understanding of practical implementations and optimization of AI NIDS, which would eventually improve the defence against ever evolving and changing cyber threats.

II. Intrusion Detection Systems

I. Network Intrusion

A network intrusion is an attempt by an unauthorized actor/entity to breach the security of a network system, thereby compromising the main fundamentals of the security CIA i.e., its confidentiality, integrity, or availability. These intrusions exploit vulnerabilities in network protocols, software configurations, or user behaviours to gain unauthorized access to sensitive data or disrupt normal network operations.

Some of the Network intrusions are as follows:

- **Reconnaissance Attacks**: Scanning or probing a network to identify vulnerabilities, open ports, or exploitable systems.
- **Exploitation Attacks**: Leveraging software bugs, configuration errors, or weaknesses in security protocols to gain access to resources or escalate privileges.
- **Denial of Service (DoS) and Distributed Denial of Service (DDoS)**: Overwhelming network resources with excessive traffic, rendering services unavailable.
- **Malware Deployment**: Introducing malicious software, such as viruses, worms, or ransomware, to compromise systems and data.
- **Man-in-the-Middle (MITM) Attacks**: Intercepting and manipulating communications between two parties without their knowledge.

These intrusions are often motivated by objectives such as data theft, financial fraud, industrial espionage, or simply causing disruption.

Network intrusions are distinguished by patterns of anomalous behaviour in network traffic or user activities. For example:

Sudden spikes in data traffic may indicate DDoS attacks.
Unusual login attempts or geographic anomalies could signal credential theft.
Unauthorized changes in system configurations may reflect insider threats.

II. IDS: Classification and Taxonomy

What are IDS?

An **Intrusion Detection System (IDS)** is a security mechanism designed to monitor network traffic and system activities for signs of malicious behaviour or policy violations. It serves as a critical component of an organization's cybersecurity strategy by identifying unauthorized access attempts, potential threats, and abnormal activities. When suspicious activity is detected, an IDS generates alerts to notify system administrators, enabling them to take appropriate action to mitigate potential risks.

Parameters	IDS (Intrusion Detection System)	IPS (Intrusion Prevention System)
Functionality:	It is a passive system that only serves the purpose of monitoring network traffic as well as alerting administrators to potential threats without taking any action to blocking suspicious activities, relying mostly on human intervention after an alert.	An IPS proactively monitors network traffic in real-time, capable of detecting and preventing threats, automatically blocking or mitigating malicious traffic without needing manual intervention.
Deployment Mode:	It is generally deployed out-of-band meaning it can passively monitor network flow without interference in its performance.	Deployed in-line , IPS sits directly in the path of network traffic allowing it to block malicious packets, although it can increase latency if not configured properly.
Response to Threats:	Requirement of manual intervention whenever a threat is generated making it slow.	Automatically responds to the threats without manual intervention hence improving response time.
Impact on Network Traffic:	Does not impact performance as it monitors passively.	Affects performance owing to its active filtering capabilities.
Complexity and Maintenance:	Easier to deploy and manage	Complex due to its careful configuration to avoid disruptions.

Table 1: IDS vs IPS

IDS can be classified into several types based on their deployment and detection methodologies. The classification based on deployment are as follows:

- **Network-Based Intrusion Detection System (NIDS)**: This type monitors the entire network by analysing traffic flowing to and from all devices. It is typically deployed at strategic points within the network to detect suspicious activity across subnets. NIDS can identify known attack patterns and anomalies in real-time.

- **Host-Based Intrusion Detection System (HIDS)**: HIDS operates on individual hosts or devices within the network. It monitors system logs, file integrity, and user activity to detect unauthorized access or malicious actions specific to that host. This type is particularly useful for protecting critical systems where changes are infrequent.

- **Protocol-Based Intrusion Detection System (PIDS)**: PIDS focuses on monitoring specific protocols used in communication between devices and servers. It analyses the behaviour of these protocols to identify any deviations that may indicate an attack.
- **Application Protocol-Based Intrusion Detection System (APIDS)**: Similar to PIDS, APIDS monitors application-specific protocols (e.g., SQL) to detect anomalies in user interactions with applications, providing insights into potential application-layer attacks.
- **Hybrid Intrusion Detection System**: This system combines features from multiple types of IDS (e.g., both NIDS and HIDS) to provide a more comprehensive security solution. By integrating data from various sources, hybrid systems enhance detection capabilities and reduce blind spots in monitoring.

Detection Methodologies

Intrusion detection methodologies can be broadly categorized into three primary types:

- **Signature-Based Detection**: This method relies on predefined patterns or signatures of known attacks. The IDS compares incoming traffic against a database of signatures to identify threats. While effective for known vulnerabilities, this approach struggles with zero-day attacks—new exploits that do not yet have established signatures.
- **Anomaly-Based Detection**: Anomaly-based IDS establishes a baseline of normal network behaviour by analysing historical data. It detects deviations from this baseline, which may indicate potential intrusions or malicious activities. Although this method can identify previously unknown threats, it is susceptible to false positives due to benign activities being flagged as anomalies.
- **Stateful Protocol Analysis**: This methodology involves monitoring the state of network connections and ensuring that they adhere to predefined protocols. By understanding the expected behaviour of a protocol, stateful analysis can detect deviations that may signify an attack.

Parameters	Signature-based detection	Stateful protocol analysis	Anomaly-based detection
Process	Signatures are based on known attack methods, such as malware fingerprints or exploit patterns, and the IDS triggers an alert when traffic matches a signature.	Compares observed traffic to predefined protocol standards (e.g., HTTP, FTP, DNS) to detect anomalies, focusing on the state and sequence of interactions, like a TCP handshake.	Uses machine learning or statistical models to establish "normal" activity and monitors network traffic or system logs to detect outliers or unusual patterns.
Advantages	Accurate and reliable for detecting known threats, with low false positive rates when the signature database is up to date.	Effective in detecting protocol-specific attacks, such as SQL injection, HTTP flooding, or DNS tunnelling, and can identify misuse of legitimate protocols	Can detect novel or zero-day attacks without predefined signatures and identify abnormal behaviour from insider threats or misconfigurations.
Limitations	Unable to detect zero-day or novel attacks without signatures and requires regular updates to stay effective against evolving threats.	Resource-intensive due to the complexity of analysing protocol states, with potential false positives from legitimate misconfigurations causing protocol deviations.	High false positive rates due to variations in normal behaviour, such as new software or updates, and requires continuous tuning and updating of baselines to remain effective.

Table 2: Different detection methodologies

III. Dataset

A. CIC-IDS- 2017

The dataset used in the training and testing of our results for different models is the CIC-IDS2017 dataset by Canadian Institute of Cybersecurity which is an intrusion detection evaluation dataset from 2017 which captures the network traffic analysis on the particular network within a week i.e. from Monday to Friday during working hours using CICFlowMeter with labelled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols and attack (CSV files). CICIDS2017 dataset contains benign and the most up-to-date common attacks, which resembles the true real-world data (PCAPs). The dataset contains more than 80 network flow features from the generated network traffic using CICFlowMeter and delivered the network flow dataset as a CSV file.

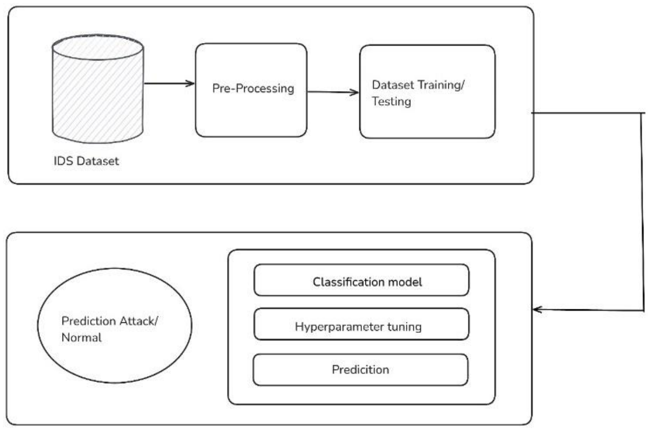


Figure 1: Data Preprocessing

B. Classification types of Detected Anomalies

Different types of Attacks identified in the dataset:

- **DoS Hulk (Denial of Service)**: This attack floods web servers with excessive HTTP GET requests to exhaust resources, causing crashes, performance issues, or denial of service to legitimate users.
- **BENIGN**: Represents normal network traffic, serving as a baseline to differentiate legitimate activities from malicious attacks.
- **PortScan**: A reconnaissance attack that scans for open ports and services to identify vulnerabilities, often preceding more severe breaches like unauthorized access or malware attacks.
- **DDoS (Distributed Denial of Service)**: A large-scale attack using multiple systems to overwhelm a target, disrupting services, exhausting resources, and causing operational or financial losses.
- **DoS GoldenEye**: An HTTP-based DoS attack sending resource-intensive requests to exhaust a server's capacity, causing slowdowns or service unavailability.
- **FTP-Patator**: A brute-force attack on FTP servers, attempting various credential combinations to gain unauthorized access, potentially resulting in data theft or malicious uploads.
- **SSH-Patator**: A brute-force attack targeting SSH services to gain unauthorized access, potentially allowing attackers to control servers, modify data, or deploy malware.
- **DoS Slowloris**: A DoS attack sending incomplete HTTP headers to keep connections open, exhausting server resources and denying access to legitimate users.
- **DoS Slowhttptest**: A DoS attack using slow, incomplete HTTP requests to exhaust server resources, causing service disruptions for users and businesses.
- **Bot**: Malicious automated programs used for tasks like spamming, phishing, or DDoS attacks, compromising network security and enabling large-scale attacks.
- **Web Attack – Brute Force**: An attack that tries all username and password combinations to gain unauthorized access, potentially leading to account compromise, data theft, or privilege escalation.
- **Web Attack – XSS (Cross-Site Scripting)**: An attack where malicious scripts are injected into web pages to steal user data, deface websites, or launch phishing attacks, compromising privacy and damaging the site's reputation.
- **Infiltration**: Unauthorized access to systems via malware or vulnerabilities, aiming to steal data, conduct espionage, or cause system damage, potentially leading to data breaches and loss of sensitive information.
- **Web Attack – SQL Injection**: An attack where malicious SQL queries manipulate databases to retrieve sensitive data, modify contents, or gain administrative access, leading to data breaches or unauthorized system access.
- **Heartbleed**: A vulnerability in OpenSSL that allows attackers to exploit the heartbeat extension to steal sensitive data, such as encryption keys and passwords, compromising security and enabling impersonation.

IV. ML and DL algorithms for IDS

Machine Learning (ML) essentially improves Intrusion Detection System (IDS) by mechanizing the distinguishing proof of advanced cyberattacks. Not at all like conventional rule-based IDS, which battle with obscure dangers like zero-day assaults, ML-based frameworks can adaptively learn and distinguish bizarre behavior designs. ML's scalability allows it to process and analyze large volumes of high-speed network traffic, making it well-suited to modern, dynamic environments. Its adaptability ensures that models evolve based on changing attack strategies, reducing the need for static rule sets. Additionally, ML improves detection accuracy by uncovering complex correlations in data, minimizing false positives, and providing better protection against advanced threats.

Deep Learning (DL) has been changing Intrusion Detection Systems (IDS) by automating feature extraction and analyzing complex, high-dimensional data such as raw traffic and system logs, eliminating the need for manual engineering. Its scalability allows real-time processing of huge data sets, making it ideal for fast, dynamic networks. By continuously being trained on new data and adapting to evolving attack patterns, DL-based IDS excel at detecting known and unknown threats, including zero-day exploits and advanced persistent threats (APTs), overcoming the limitations of traditional rule-based systems.

Parameters	**Machine Learning**	**Deep Learning**
Strengths	Machine learning-based IDS offers adaptive threat detection by evolving with new data, enabling effective identification of emerging and novel attacks. It achieves high accuracy by detecting complex patterns, reducing false positives and negatives. Automation allows it to analyse large datasets without manual intervention, while its scalability ensures real-time processing, making it ideal for high-bandwidth networks.	Deep learning (DL) enhances IDS with improved detection rates by identifying subtle, complex patterns, ensuring high accuracy for both known and novel attacks. It scales efficiently, analysing massive datasets in real-time for high-traffic networks. DL systems adapt to evolving threats by learning from new data and automate feature extraction, eliminating manual engineering to save time and reduce errors.
Weaknesses	Machine learning-based IDS faces challenges such as dependence on high-quality, well-labelled data, with imbalanced datasets leading to biased models. While anomaly-based systems excel at detecting novel attacks, signature-based models are limited to known threats. False positives can overwhelm security teams with "alert fatigue," and ML models require ongoing tuning and retraining to adapt to evolving threats. Additionally, these systems are resource-intensive, with some algorithms demanding significant computational power.	Deep learning (DL) in IDS faces challenges such as the need for large, labelled datasets, which are scarce due to privacy concerns and rare attack types. Its high computational overhead requires significant resources, like GPUs and cloud infrastructure, which may not be accessible to all organizations. DL models often lack interpretability, functioning as "black boxes" that can hinder analyst trust. Additionally, they are prone to overfitting, especially when trained on imbalanced datasets with fewer attack samples than benign ones.

Table 3: ML vs DL in IDS

A. Logistic Regression (LR)

A linear regression model that is mostly used for binary classification but can be used for multi-class classification using *SoftMax Function.* By calculating an instance's probability of belonging to a certain class using SoftMax function.

$$\boldsymbol{F(y = k\,|x)} = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

where $z_k = w_k^T x + b_k$ is the class k score. The output provides probability for every class and the class with highest probability is chosen.

The model in fast, easy to implement, and is efficient for large datasets like CIC-NID-2017. Since it assumes a linear relationship features and output, the performance can be limited for complex data.

B. Random Forest (XGBoost)

An ensemble method that works by building multiple decision trees, trained on random subset of data. Decision trees then make individual predictions and the class with the highest average is predicted as final output. The feature selection is automatic and provides feature importance based on the performance of features in the decision trees. You can extensively control the behaviour of model by providing build criteria such as Gini impurity or information gain to determine the best feature, maximum number of trees (as per the need/hardware capability), maximum depth of the trees.

XGBoost builds trees sequentially with a gradient boosting algorithm that tries to correct the mistakes of previous trees by minimizing a loss function using gradient descent.

$$\theta = \theta - \alpha \nabla_{\theta} J(\theta)$$

The model is very robust against noise, outliers, missing data, and it can also prevent overfitting by randomizing the data subset. The drawback is that the cost of computation is extremely high which can take a toll on hardware, especially with substantial number of trees.

C. K-Nearest Neighbour (KNN)

The model is a lazy learning algorithm that classifies an instance based on K nearest neighbours' classes in the n_f *dimensional* feature space by choosing the class with majority. The distances between instances can be calculated by Euclidean distance (or Manhattan distance).

$$Distance(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

The model is simple and intuitive, no assumptions about data distribution and no parametric tuning is required. The simpleness comes at a cost though, the model can be computationally expensive for large datasets since it has to calculate the distance to every other instance which can slow down the process. It is sensitive to noise and outliers, and high dimensionality can reduce the effectiveness of the model. Thus, making feature selection important.

D. Feedforward Neural Network (FNN)

The model is composed of an input layer, one or more hidden layer, and an output layer. Every layer is made up of neurons, which computes a weighted sum of inputs and feed the result to activation function *(ReLU)*, which introduces non-linearity in the model, enabling it to learn complex functions as well. Weights are adjusted using backpropagation and gradient descent to diminish the error between actual output and predicted.

The model can be powerful against complex relationships in the dataset and has a flexible architecture, allowing for the model to intricate patterns. The catch being a great thirst for data, a large data sample is required to perform optimally and, it is computationally expensive.

E. Long Short-Term Memory (LSTM)

A form of Recurrent Neural Network (RNN) built to work on sequential data and long-term dependency. LSTM's architecture consists of gates allowing for the model to remember long-term dependency:

- Input Gate: Manages the amount of input to be stored into memory.
- Forget Gate: Controls what information is forgotten by the memory.
- Output Gate: Manages the amount of information that should be outputted.

The model is robust against both vanishing and exploding gradient problem. It performs better than traditional RNNs as it can better store long-term dependencies. The cost of computation high and can be time consuming, especially for large datasets and, it is difficult to tune due to high sensitivity of the model.

V. Evaluation Criteria

We are using the following criteria and metrics to the overall performance of the models.

- Confusion matrix:

A table that is used to evaluate the overall classification performance of a given model by comparing the actual values with predicted values to analyse the classification results.

Components of the Table:

- True Positives (TP): Correctly predicted positive instances. Instances of that class that are correctly classified.
- True Negatives (TN): Correctly predicted negative instances. Instances that are not of that class and correctly classified as negative.
- False Positives (FP): Incorrectly predicted positive instances. Instances that are not of that class but classified as one.
- False Negatives (FN): Incorrectly predicted negative instances. Instances that are of that class but negatively classified.

- Accuracy:

It is the ration of correctly classified instances to total number of instances. A measure to understand overall performance of the model. Higher accuracy represents the model is correctly classifying most of the instances.

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- Precision

It is the ratio of correctly classified instances of a class to total predicted instances of that class. Also called true positive rate.

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Recall

A measure of how accurately the model classifies all positive instances. The ratio of correctly classified instances to total actual class instances.

$$Recall \; = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- F1-score

It is the harmonic mean of recall and precision to get an even balance between the two. Higher F1-Score means great balance between precision and recall. It is useful for uneven class distributions and can help maintain a balance trade off.

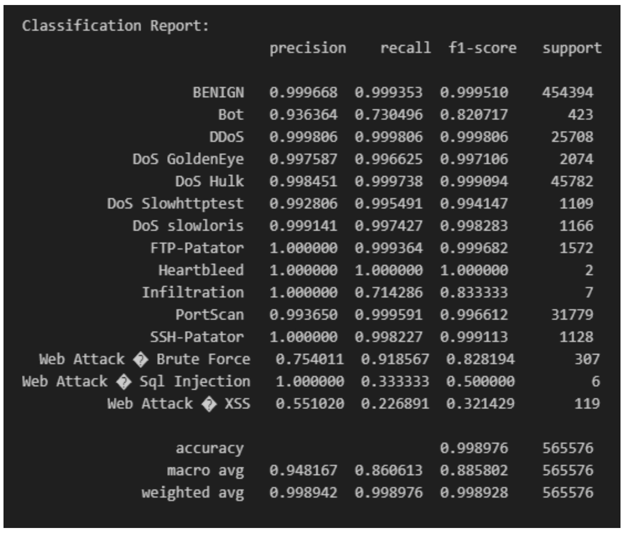
$$F1\ Score \; = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

VI. Result

A. Logistic Regression

The performance of Logistic Regression model on the CIC-IDS-2017 dataset is very impressive given the simple and efficient nature of the model. The model gave us a very high weighted accuracy score of 99.897% and all other metrics are also near 99.89%.

The classes with large sample size like BENIGN, DDoS and PortScan gave us near-perfect metric scores. Attack types like DoS Hulk, DoS GoldenEye and DoS slowloris are showing results over 99% and even the classes with relatively low sample size (Heartbleed, Infiltration and Web Attacks) are somewhat correctly identified, since we do not have large enough sample size for these, the accuracy takes a hit.

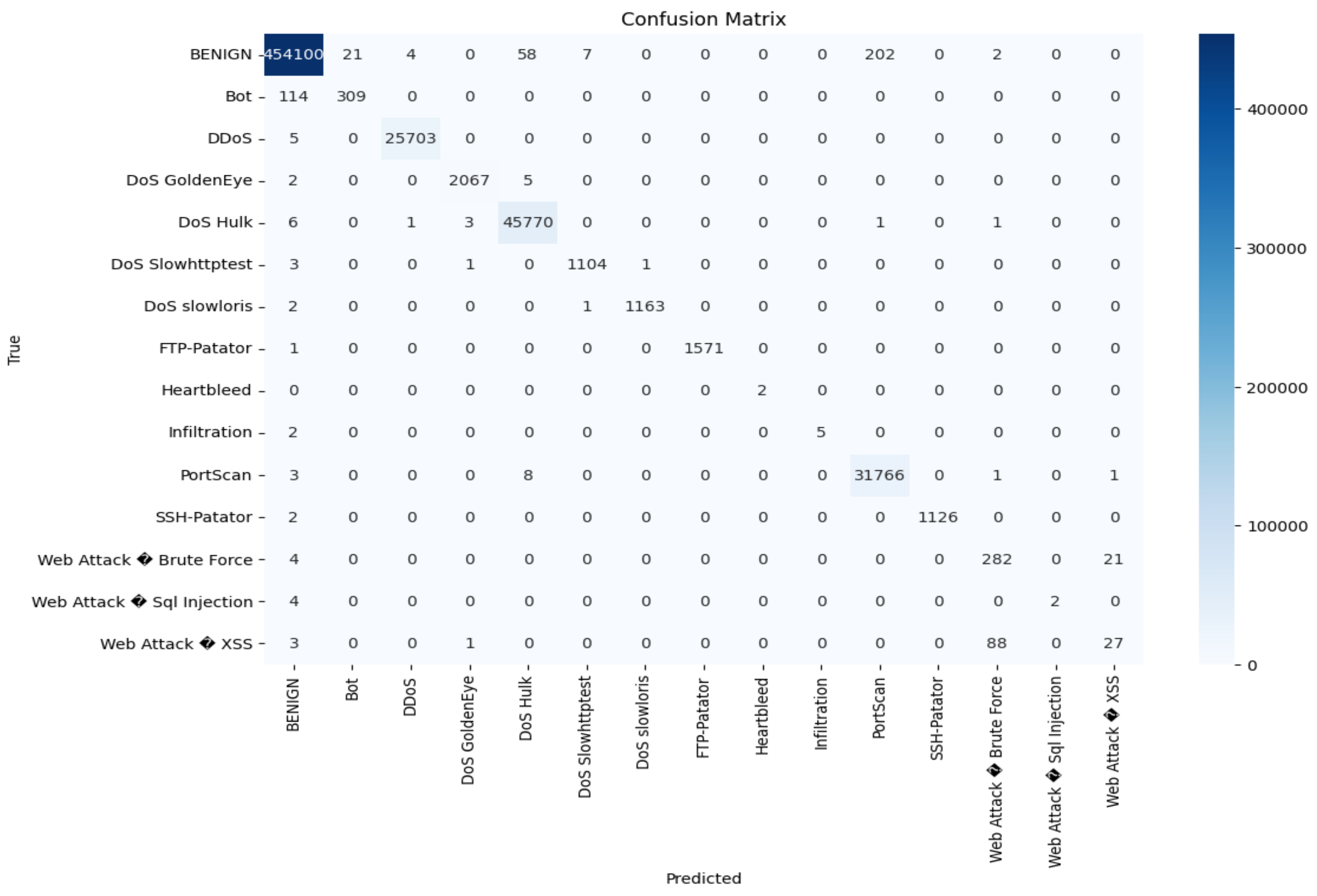


```
Classification Report:
                              precision    recall  f1-score   support

                      BENIGN   0.999668  0.999353  0.999510    454394
                         Bot   0.936364  0.730496  0.820717       423
                        DDoS   0.999806  0.999806  0.999806     25708
               DoS GoldenEye   0.997587  0.996625  0.997106      2074
                    DoS Hulk   0.998451  0.999738  0.999094     45782
            DoS Slowhttptest   0.992806  0.995491  0.994147      1109
               DoS slowloris   0.999141  0.997427  0.998283      1166
                 FTP-Patator   1.000000  0.999364  0.999682      1572
                  Heartbleed   1.000000  1.000000  1.000000         2
                Infiltration   1.000000  0.714286  0.833333         7
                    PortScan   0.993650  0.999591  0.996612     31779
                 SSH-Patator   1.000000  0.998227  0.999113      1128
  Web Attack � Brute Force    0.754011  0.918567  0.828194       307
Web Attack � Sql Injection    1.000000  0.333333  0.500000         6
          Web Attack � XSS    0.551020  0.226891  0.321429       119

                    accuracy                       0.998976    565576
                   macro avg   0.948167  0.860613  0.885802    565576
                weighted avg   0.998942  0.998976  0.998928    565576
```

Figure 2: Classification report (Logistic Regression)



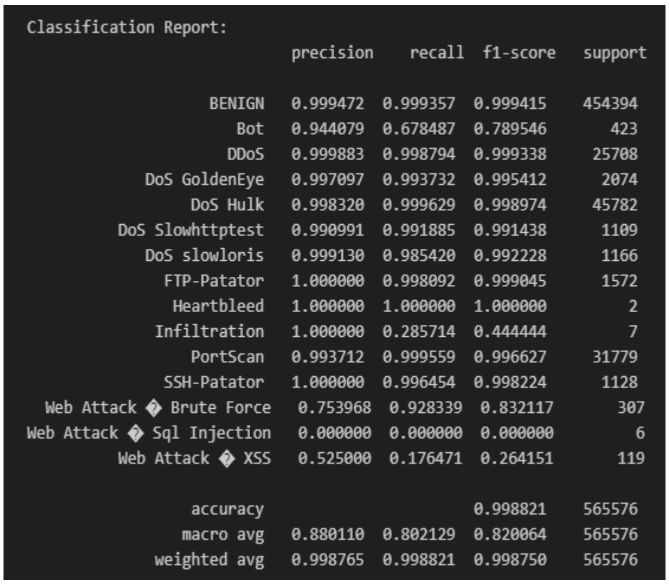
	BENIGN	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS slowloris	FTP-Patator	Heartbleed	Infiltration	PortScan	SSH-Patator	Web Attack � Brute Force	Web Attack � Sql Injection	Web Attack � XSS
BENIGN	454100	21	4	0	58	7	0	0	0	0	202	0	2	0	0
Bot	114	309	0	0	0	0	0	0	0	0	0	0	0	0	0
DDoS	5	0	25703	0	0	0	0	0	0	0	0	0	0	0	0
DoS GoldenEye	2	0	0	2067	5	0	0	0	0	0	0	0	0	0	0
DoS Hulk	6	0	1	3	45770	0	0	0	0	0	1	0	1	0	0
DoS Slowhttptest	3	0	0	1	0	1104	1	0	0	0	0	0	0	0	0
DoS slowloris	2	0	0	0	0	1	1163	0	0	0	0	0	0	0	0
FTP-Patator	1	0	0	0	0	0	0	1571	0	0	0	0	0	0	0
Heartbleed	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
Infiltration	2	0	0	0	0	0	0	0	0	5	0	0	0	0	0
PortScan	3	0	0	0	8	0	0	0	0	0	31766	0	1	0	1
SSH-Patator	2	0	0	0	0	0	0	0	0	0	0	1126	0	0	0
Web Attack � Brute Force	4	0	0	0	0	0	0	0	0	0	0	0	282	0	21
Web Attack � Sql Injection	4	0	0	0	0	0	0	0	0	0	0	0	0	2	0
Web Attack � XSS	3	0	0	1	0	0	0	0	0	0	0	0	88	0	27

Figure 3: Confusion Matrix (Logistic Regression)

B. Random Forest (XGBoost)

The Random Forest (XGBoost) model's performance was overall high and similar to Logistic Regression in most areas apart from minority classes where Random Forest suffers a bit more. Although the performance was exceptionally well for majority classes (BENIGN, DDoS, DoSs, FTP-Patator) achieving an accuracy of 99.882% and all other metrics over 99.875%, the classes with relatively low sample size suffers even more. Some (Web Attacks and Infiltration) flat out not being detected at all.

Further parameter tuning may improve the model somewhat but the major bottleneck for the perform is lack of minority class samples. On the whole, the model performs on a high and can be improved further with more time and resource investment.

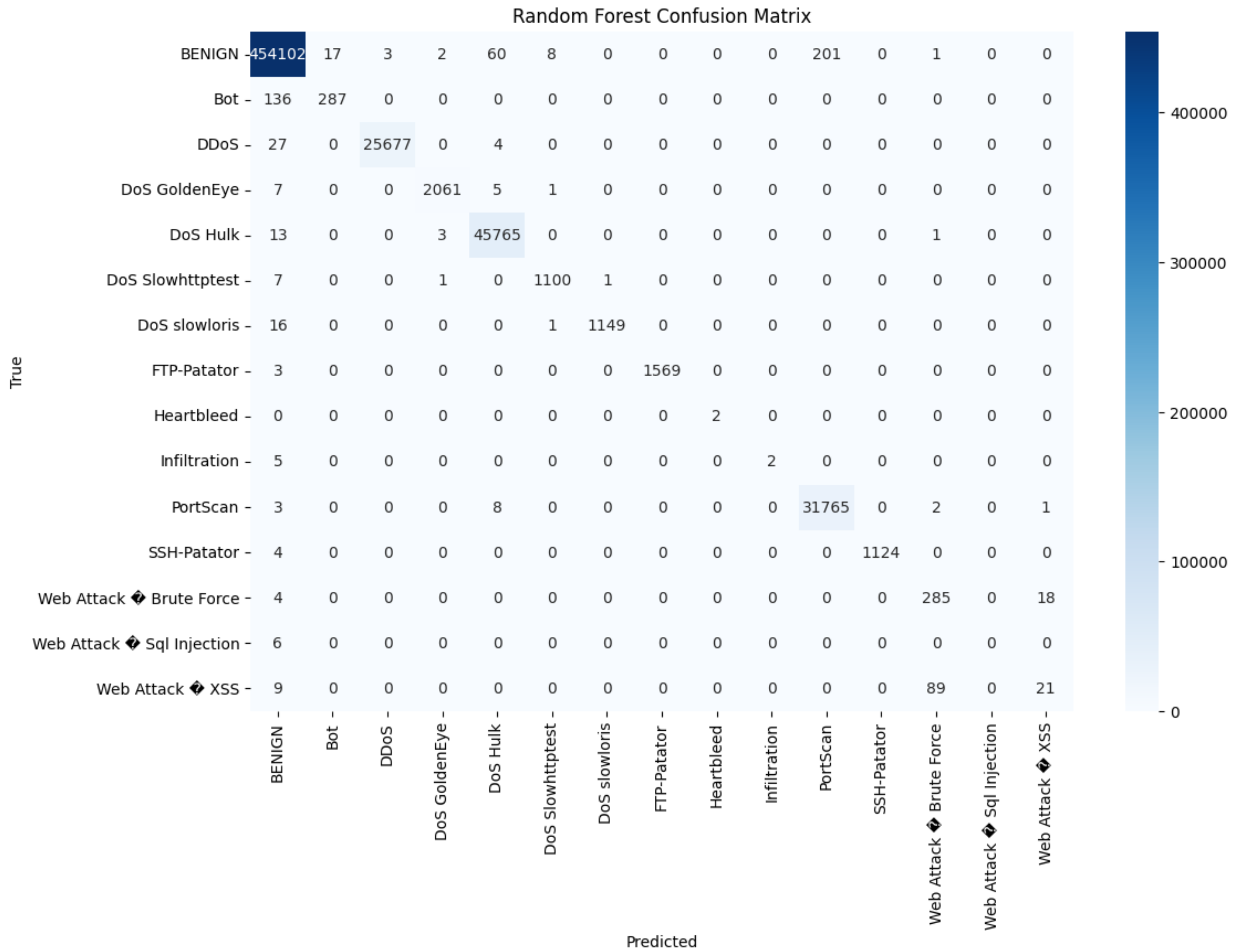


```
Classification Report:
                            precision    recall  f1-score   support

                    BENIGN   0.999472  0.999357  0.999415    454394
                       Bot   0.944079  0.678487  0.789546       423
                      DDoS   0.999883  0.998794  0.999338     25708
             DoS GoldenEye   0.997097  0.993732  0.995412      2074
                  DoS Hulk   0.998320  0.999629  0.998974     45782
          DoS Slowhttptest   0.990991  0.991885  0.991438      1109
             DoS slowloris   0.999130  0.985420  0.992228      1166
               FTP-Patator   1.000000  0.998092  0.999045      1572
                Heartbleed   1.000000  1.000000  1.000000         2
              Infiltration   1.000000  0.285714  0.444444         7
                  PortScan   0.993712  0.999559  0.996627     31779
               SSH-Patator   1.000000  0.996454  0.998224      1128
  Web Attack � Brute Force   0.753968  0.928339  0.832117       307
Web Attack � Sql Injection   0.000000  0.000000  0.000000         6
         Web Attack � XSS   0.525000  0.176471  0.264151       119

                  accuracy                       0.998821    565576
                 macro avg   0.880110  0.802129  0.820064    565576
              weighted avg   0.998765  0.998821  0.998750    565576
```

Figure 4: Classification Report (Random Forest)



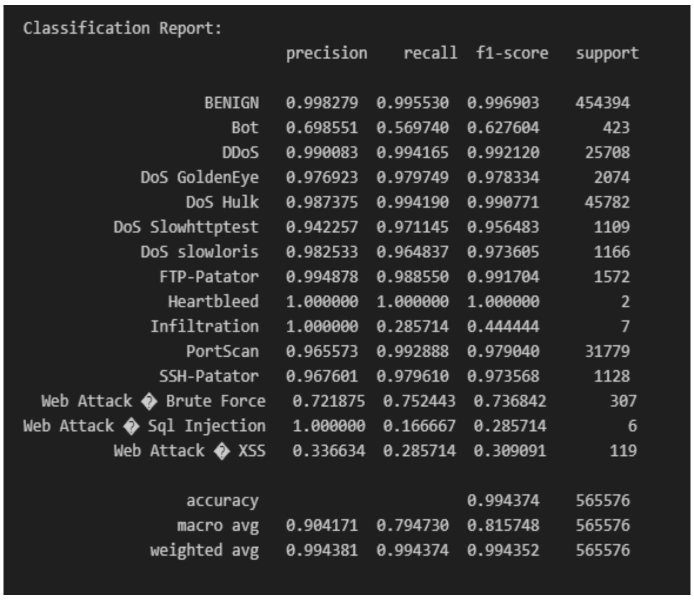
True \ Predicted	BENIGN	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS slowloris	FTP-Patator	Heartbleed	Infiltration	PortScan	SSH-Patator	Web Attack � Brute Force	Web Attack � Sql Injection	Web Attack � XSS
BENIGN	454102	17	3	2	60	8	0	0	0	0	201	0	1	0	0
Bot	136	287	0	0	0	0	0	0	0	0	0	0	0	0	0
DDoS	27	0	25677	0	4	0	0	0	0	0	0	0	0	0	0
DoS GoldenEye	7	0	0	2061	5	1	0	0	0	0	0	0	0	0	0
DoS Hulk	13	0	0	3	45765	0	0	0	0	0	0	0	1	0	0
DoS Slowhttptest	7	0	0	1	0	1100	1	0	0	0	0	0	0	0	0
DoS slowloris	16	0	0	0	0	1	1149	0	0	0	0	0	0	0	0
FTP-Patator	3	0	0	0	0	0	0	1569	0	0	0	0	0	0	0
Heartbleed	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
Infiltration	5	0	0	0	0	0	0	0	0	2	0	0	0	0	0
PortScan	3	0	0	0	8	0	0	0	0	0	31765	0	2	0	1
SSH-Patator	4	0	0	0	0	0	0	0	0	0	0	1124	0	0	0
Web Attack � Brute Force	4	0	0	0	0	0	0	0	0	0	0	0	285	0	18
Web Attack � Sql Injection	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Web Attack � XSS	9	0	0	0	0	0	0	0	0	0	0	0	89	0	21

Figure 5: Confusion Matrix (Random Forest)

C. K-Nearest Neighbour (KNN)

The KNN model produces a decent result with an accuracy of 99.22% which might look impressive at first glance but lacks performance in the minority classes (Web Attacks, Bot, Infiltration). Signifying a major struggle with minority classes which might be due to relatively low sample size. Since the model uses its nearest neighbours to predict the output, the inherent lack of minority class instances really hurts the performance.

The model performs well given its lazy nature, but for any improvements to be made it requires more data samples of minority classes making it less viable when compared to Logistic Regression (99.89% accuracy) and Random Forest (99.88% accuracy) at least for this particular dataset.

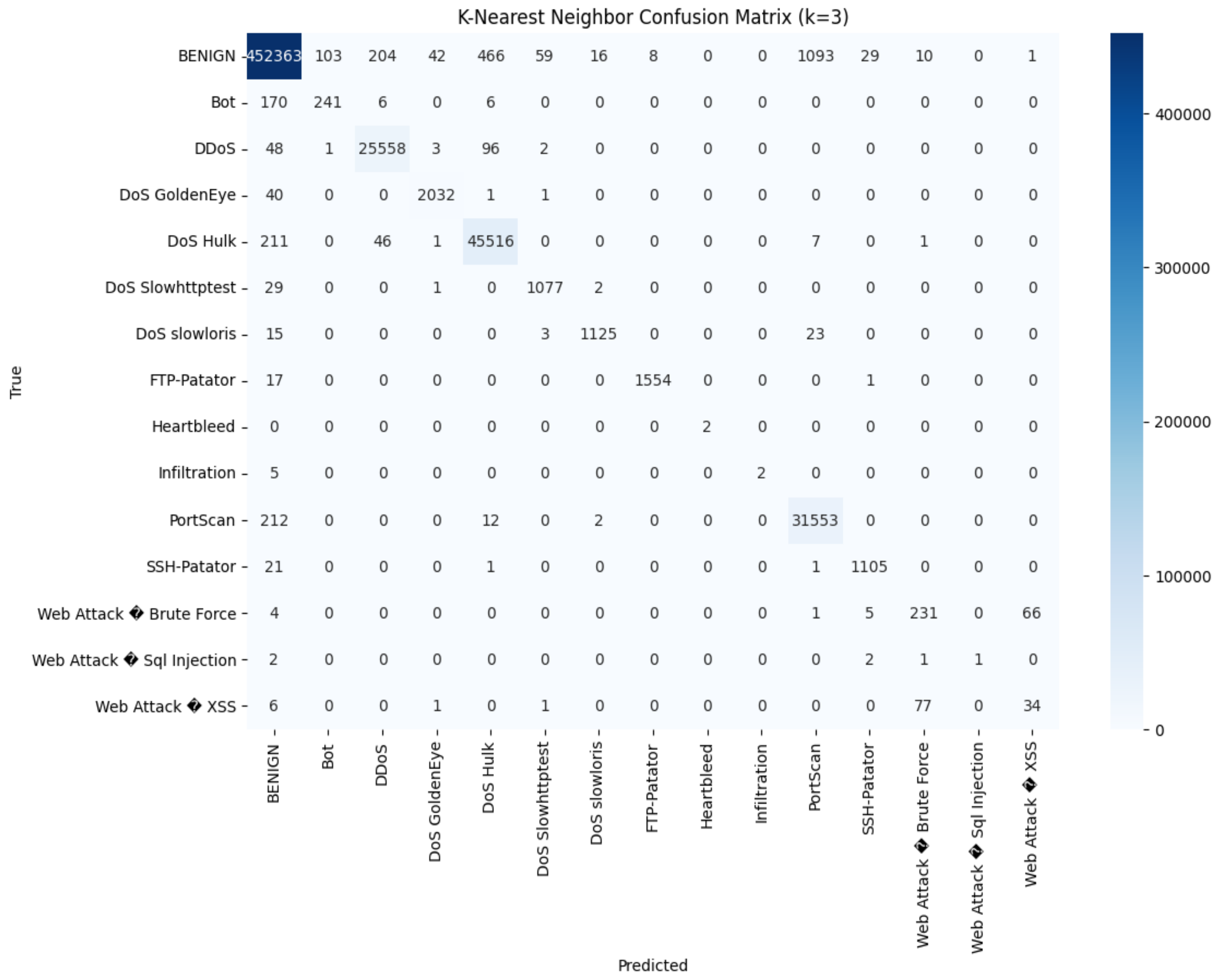


```
Classification Report:
                              precision    recall  f1-score   support

                      BENIGN   0.998279  0.995530  0.996903    454394
                         Bot   0.698551  0.569740  0.627604       423
                        DDoS   0.990083  0.994165  0.992120     25708
               DoS GoldenEye   0.976923  0.979749  0.978334      2074
                    DoS Hulk   0.987375  0.994190  0.990771     45782
            DoS Slowhttptest   0.942257  0.971145  0.956483      1109
               DoS slowloris   0.982533  0.964837  0.973605      1166
                 FTP-Patator   0.994878  0.988550  0.991704      1572
                  Heartbleed   1.000000  1.000000  1.000000         2
                Infiltration   1.000000  0.285714  0.444444         7
                    PortScan   0.965573  0.992888  0.979040     31779
                 SSH-Patator   0.967601  0.979610  0.973568      1128
   Web Attack � Brute Force    0.721875  0.752443  0.736842       307
Web Attack � Sql Injection     1.000000  0.166667  0.285714         6
           Web Attack � XSS    0.336634  0.285714  0.309091       119

                    accuracy                       0.994374    565576
                   macro avg   0.904171  0.794730  0.815748    565576
                weighted avg   0.994381  0.994374  0.994352    565576
```

Figure 6: Classification Report (KNN)



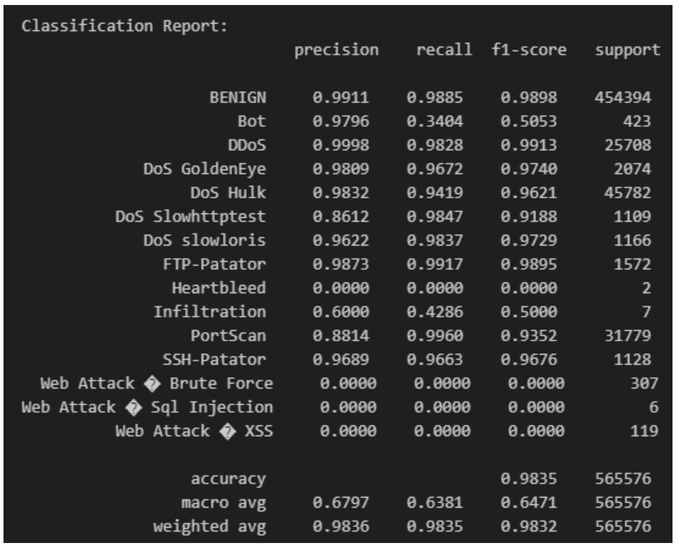
	BENIGN	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS slowloris	FTP-Patator	Heartbleed	Infiltration	PortScan	SSH-Patator	Web Attack � Brute Force	Web Attack � Sql Injection	Web Attack � XSS
BENIGN	452363	103	204	42	466	59	16	8	0	0	1093	29	10	0	1
Bot	170	241	6	0	6	0	0	0	0	0	0	0	0	0	0
DDoS	48	1	25558	3	96	2	0	0	0	0	0	0	0	0	0
DoS GoldenEye	40	0	0	2032	1	1	0	0	0	0	0	0	0	0	0
DoS Hulk	211	0	46	1	45516	0	0	0	0	0	7	0	1	0	0
DoS Slowhttptest	29	0	0	1	0	1077	2	0	0	0	0	0	0	0	0
DoS slowloris	15	0	0	0	0	3	1125	0	0	0	23	0	0	0	0
FTP-Patator	17	0	0	0	0	0	0	1554	0	0	0	1	0	0	0
Heartbleed	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
Infiltration	5	0	0	0	0	0	0	0	0	2	0	0	0	0	0
PortScan	212	0	0	0	12	0	2	0	0	0	31553	0	0	0	0
SSH-Patator	21	0	0	0	1	0	0	0	0	0	1	1105	0	0	0
Web Attack � Brute Force	4	0	0	0	0	0	0	0	0	0	1	5	231	0	66
Web Attack � Sql Injection	2	0	0	0	0	0	0	0	0	0	0	2	1	1	0
Web Attack � XSS	6	0	0	1	0	1	0	0	0	0	0	0	77	0	34

Figure 7: Confusion Matrix (KNN)

D. Feedforward Neural Network

The Feedforward Neural Network (FNN) produces a decent result with an overall accuracy of 98.35% by correctly classifying all majority classes mostly. The performance for medium sized classes (PortScan, DoSs, SSH-Patator) is subpar with an overall F1-Score of 91%. The model, like rest struggles with minority classes producing a below average evaluation metric.

The model is potentially overfitting to majority classes thus influencing minority classification. Compared to other models used so far, this does not show any advantage at the moment possibly due to above mentioned reasons and low data samples but since neural networks can be sensitive, hyperparameter tuning can help the case (increasing the hidden layer and neurons) or transitioning to a more complex model, such as CNN can help better understand the complex patterns of the dataset.

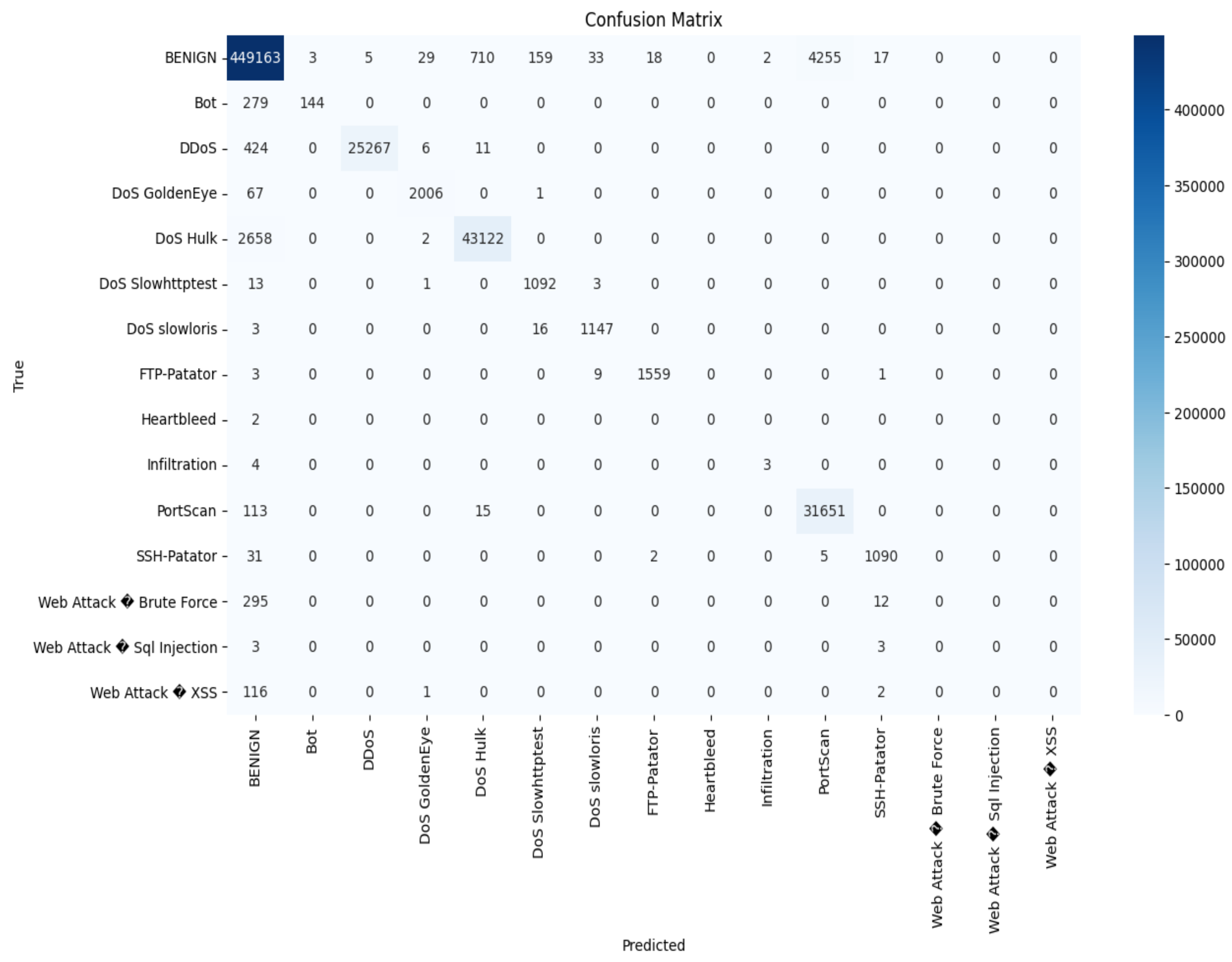


```
Classification Report:
                             precision    recall  f1-score   support

                    BENIGN      0.9911    0.9885    0.9898    454394
                       Bot      0.9796    0.3404    0.5053       423
                      DDoS      0.9998    0.9828    0.9913     25708
             DoS GoldenEye      0.9809    0.9672    0.9740      2074
                  DoS Hulk      0.9832    0.9419    0.9621     45782
          DoS Slowhttptest      0.8612    0.9847    0.9188      1109
             DoS slowloris      0.9622    0.9837    0.9729      1166
               FTP-Patator      0.9873    0.9917    0.9895      1572
                Heartbleed      0.0000    0.0000    0.0000         2
              Infiltration      0.6000    0.4286    0.5000         7
                  PortScan      0.8814    0.9960    0.9352     31779
               SSH-Patator      0.9689    0.9663    0.9676      1128
  Web Attack � Brute Force      0.0000    0.0000    0.0000       307
Web Attack � Sql Injection      0.0000    0.0000    0.0000         6
          Web Attack � XSS      0.0000    0.0000    0.0000       119

                  accuracy                          0.9835    565576
                 macro avg      0.6797    0.6381    0.6471    565576
              weighted avg      0.9836    0.9835    0.9832    565576
```

Figure 8: Classification Report (FNN)



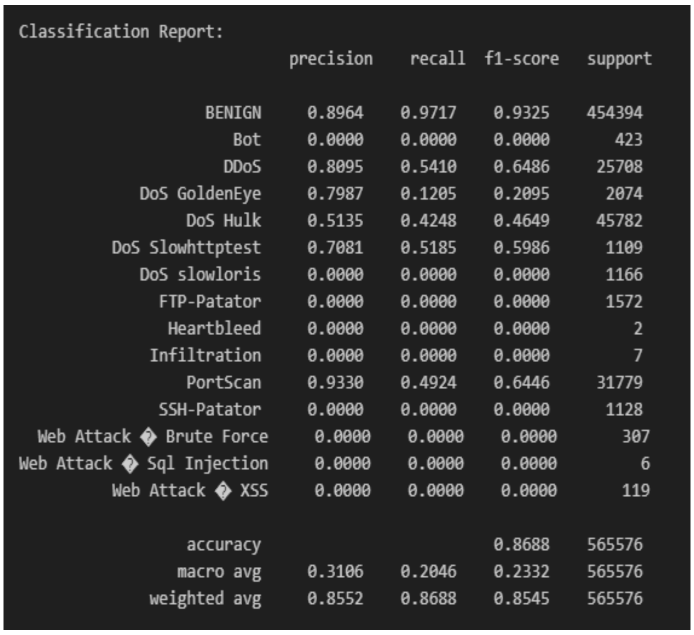
True \ Predicted	BENIGN	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS slowloris	FTP-Patator	Heartbleed	Infiltration	PortScan	SSH-Patator	Web Attack � Brute Force	Web Attack � Sql Injection	Web Attack � XSS
BENIGN	449163	3	5	29	710	159	33	18	0	2	4255	17	0	0	0
Bot	279	144	0	0	0	0	0	0	0	0	0	0	0	0	0
DDoS	424	0	25267	6	11	0	0	0	0	0	0	0	0	0	0
DoS GoldenEye	67	0	0	2006	0	1	0	0	0	0	0	0	0	0	0
DoS Hulk	2658	0	0	2	43122	0	0	0	0	0	0	0	0	0	0
DoS Slowhttptest	13	0	0	1	0	1092	3	0	0	0	0	0	0	0	0
DoS slowloris	3	0	0	0	0	16	1147	0	0	0	0	0	0	0	0
FTP-Patator	3	0	0	0	0	0	9	1559	0	0	0	1	0	0	0
Heartbleed	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Infiltration	4	0	0	0	0	0	0	0	0	3	0	0	0	0	0
PortScan	113	0	0	0	15	0	0	0	0	0	31651	0	0	0	0
SSH-Patator	31	0	0	0	0	0	0	2	0	0	5	1090	0	0	0
Web Attack � Brute Force	295	0	0	0	0	0	0	0	0	0	0	12	0	0	0
Web Attack � Sql Injection	3	0	0	0	0	0	0	0	0	0	0	3	0	0	0
Web Attack � XSS	116	0	0	1	0	0	0	0	0	0	0	2	0	0	0

Figure 9: Confusion Matrix (FNN)

E. Long Short-Term Memory (LSTM)

The Long Short-Term Memory shows significant weakness, mainly in case of minority class, and an overall relatively underwhelming performance for the dataset. BENIGN receives a high recall of 97.17% and an F1-score of 93.25, but other classes archive moderate performance recall of 54.10%, 49.24%, 46.49% for DDoS, PortScan and DoS Hulk respectively. Some even completely failed to be recognised by the model, mostly minority classes (Web Attacks, SSH-Patator, Infiltration, FTP-Patator, Heartbleed).

Even though the model shows poor performance here, LSTMs are inherently more suitable for sequential data, so there might be room for improvement with better data processing and parameter tuning. Potential overfitting for majority classes and insufficient training for minority classes might be hurting the model more than we expected. Since LSTM is sensitive to tuning and requires large data samples to perform at its best, there is room for improvements.

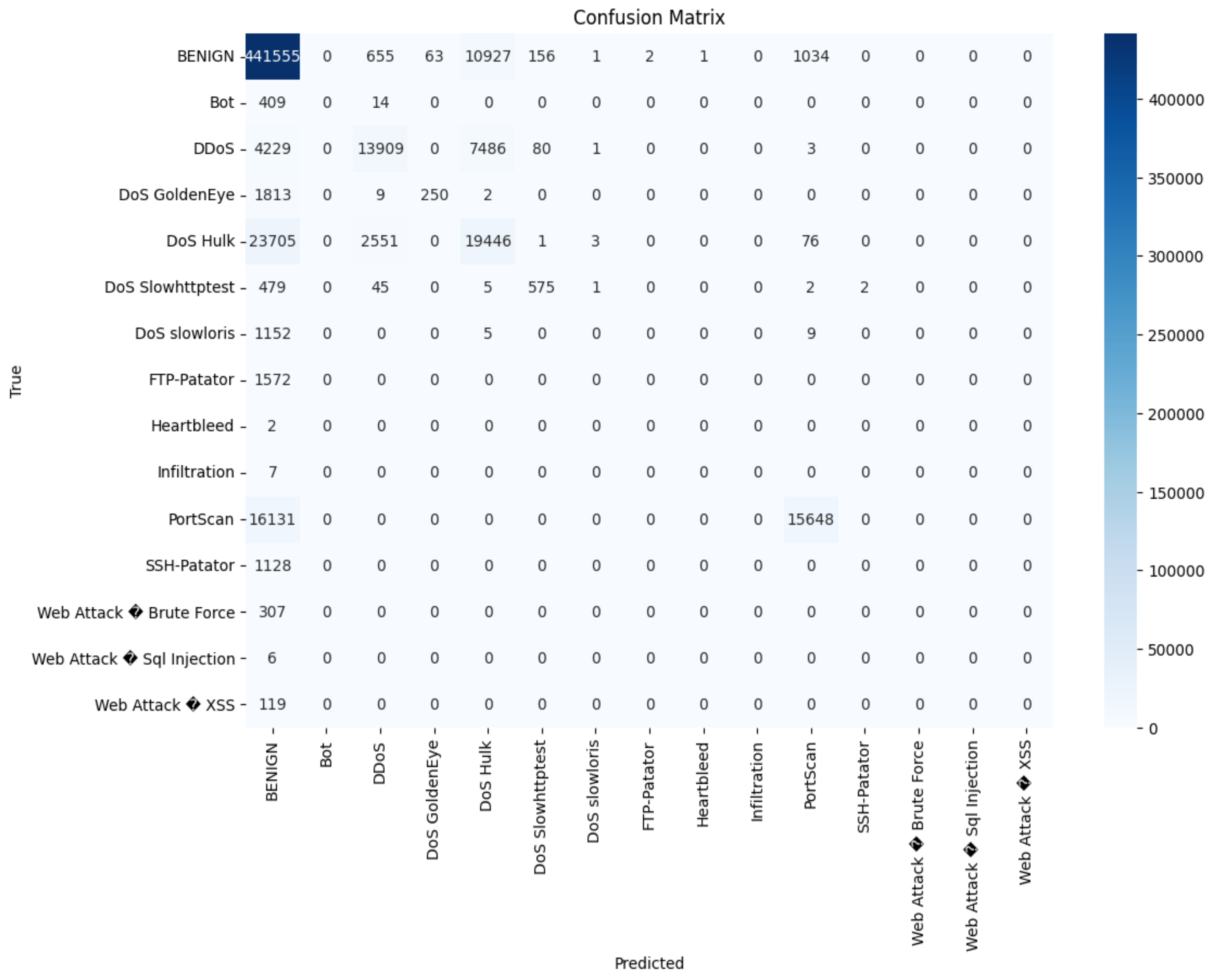


```
Classification Report:
                              precision    recall  f1-score   support

                      BENIGN     0.8964    0.9717    0.9325    454394
                         Bot     0.0000    0.0000    0.0000       423
                        DDoS     0.8095    0.5410    0.6486     25708
               DoS GoldenEye     0.7987    0.1205    0.2095      2074
                    DoS Hulk     0.5135    0.4248    0.4649     45782
            DoS Slowhttptest     0.7081    0.5185    0.5986      1109
               DoS slowloris     0.0000    0.0000    0.0000      1166
                 FTP-Patator     0.0000    0.0000    0.0000      1572
                  Heartbleed     0.0000    0.0000    0.0000         2
                Infiltration     0.0000    0.0000    0.0000         7
                    PortScan     0.9330    0.4924    0.6446     31779
                 SSH-Patator     0.0000    0.0000    0.0000      1128
  Web Attack � Brute Force     0.0000    0.0000    0.0000       307
Web Attack � Sql Injection     0.0000    0.0000    0.0000         6
          Web Attack � XSS     0.0000    0.0000    0.0000       119

                    accuracy                         0.8688    565576
                   macro avg     0.3106    0.2046    0.2332    565576
                weighted avg     0.8552    0.8688    0.8545    565576
```

Figure 10: Classification Report (LSTM)



	BENIGN	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS slowloris	FTP-Patator	Heartbleed	Infiltration	PortScan	SSH-Patator	Web Attack � Brute Force	Web Attack � Sql Injection	Web Attack � XSS
BENIGN	441555	0	655	63	10927	156	1	2	1	0	1034	0	0	0	0
Bot	409	0	14	0	0	0	0	0	0	0	0	0	0	0	0
DDoS	4229	0	13909	0	7486	80	1	0	0	0	3	0	0	0	0
DoS GoldenEye	1813	0	9	250	2	0	0	0	0	0	0	0	0	0	0
DoS Hulk	23705	0	2551	0	19446	1	3	0	0	0	76	0	0	0	0
DoS Slowhttptest	479	0	45	0	5	575	1	0	0	0	2	2	0	0	0
DoS slowloris	1152	0	0	0	5	0	0	0	0	0	9	0	0	0	0
FTP-Patator	1572	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Heartbleed	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Infiltration	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PortScan	16131	0	0	0	0	0	0	0	0	0	15648	0	0	0	0
SSH-Patator	1128	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Web Attack � Brute Force	307	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Web Attack � Sql Injection	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Web Attack � XSS	119	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11: Confution Matrix (LSTM)

VII. Conclusion

After evaluating multiple models for CIC-IDS-2017 dataset, we can safely say that some models like Logistic Regression (SoftMax) and Random Forest (XGBoost), delivered excellent performance in terms of class detection and accuracy, while others like LSTM, struggled significantly.

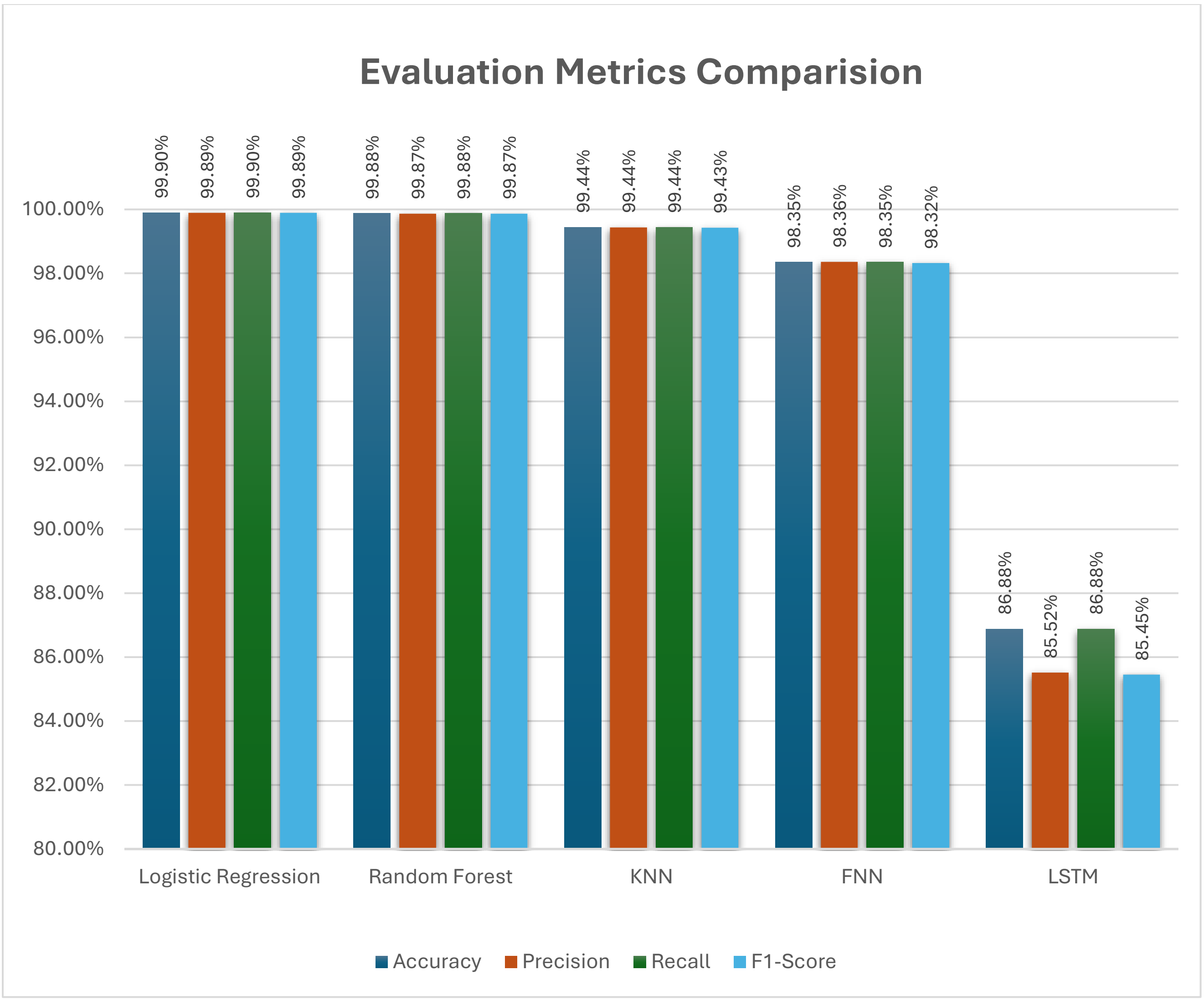


Figure 12: Evaluation Metrics Comparision

Random Forest gave overall best results terms of accuracy, precision, recall and F1-score across most classes. Managed to handle imbalance of the dataset effectively and produced highest accuracy (99.882%). Similarly, Logistic Regression (99.897%) was almost as good as Random Forest if not more while being simple and efficient. There is very little trade-off between the two in terms of performance, considering the low computational requirements of Logistic Regression, most times than not it can be preferred without settling for compromise.

KNN (99.43%) and FNN (98.35%) performed moderately, the reason for their relatively poor score is slightly lower performance for minority classes. Their performance for majority class was similar to that of the above models but their inability to distinguish minority classes due to relatively low data samples hindered them.

LSTM significantly underperformed when compared to others, this might have been due to lack of data sample for minority class or overfitting for majority class. Regardless of the current state of the model, we believe there is a significant room for improvement given the inherent nature of LSTM to especially work well on sequential data.

Note: The accuracy for above models might look impressive at first glance but we encourage you to look into the classification reports for each model to better understand which models captures most intricate details of the CIC-IDS-2017 dataset.

Below *(Figure 13)* is the feature importance chart which shows the importance of all the features in CIC-IDS-2017 dataset and how much they affect the output.

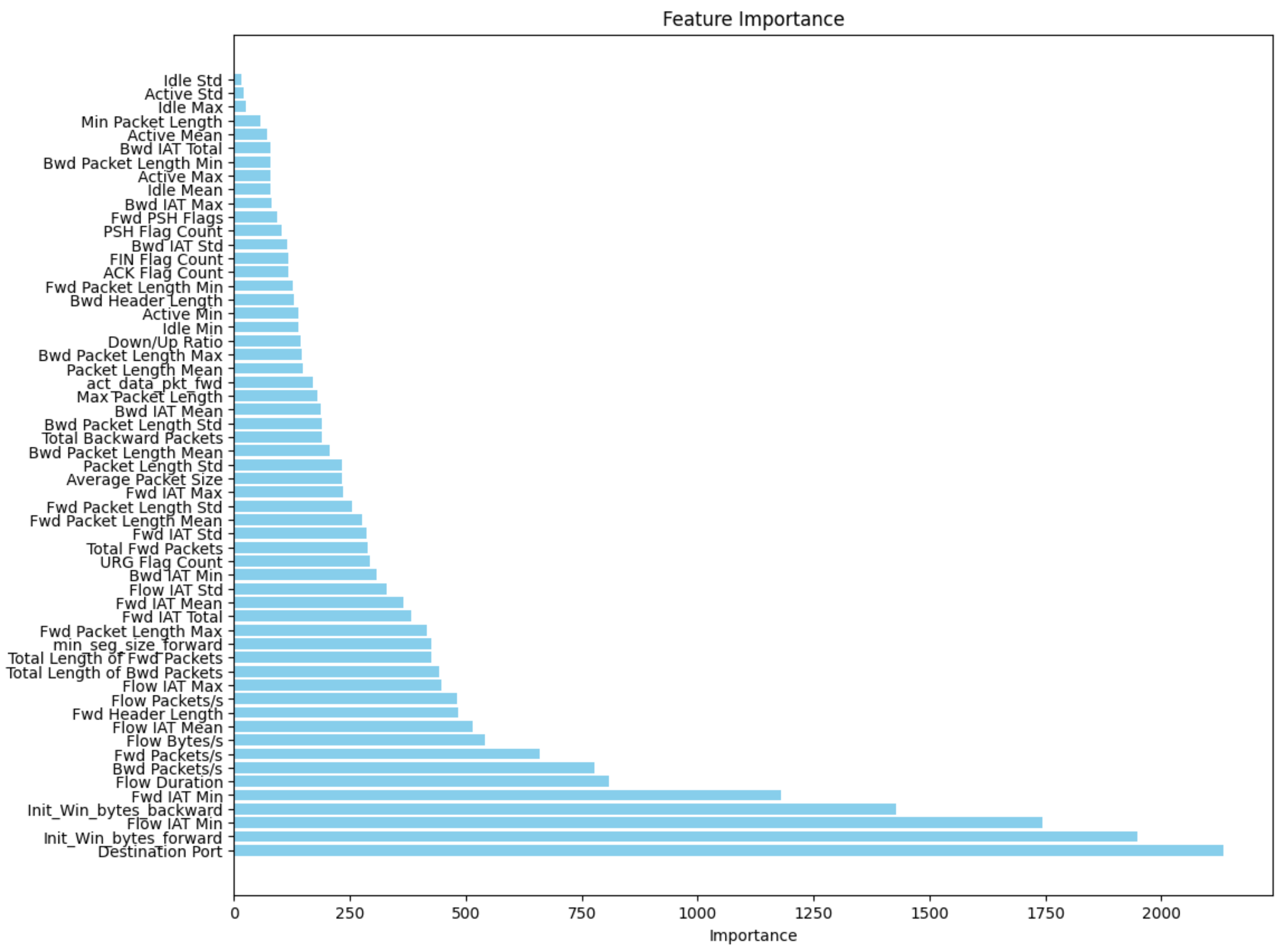


Figure 13: Feature Importance

VIII. References

[1] Faker O. and Dogdu E., "Intrusion Detection Using Big Data and Deep Learning Techniques," in Proceedings of the ACM Southeast Conference, New York, pp. 86-93, 2019.

[2] Kim J., Kim J., Kim H., Shim M., and Choi E., "CNN-Based Network Intrusion Detection against Denial-of-Service Attacks," Electronics, vol. 9, no. 6, pp. 1-21, 2020.

[3] Li Z. and Qin Z., "A Semantic Parsing Based LSTM Model for Intrusion Detection," in proceedings of International Conference on Neural Information Processing, Cambodia, pp. 600-609, 2018.

[4] Khan M., "Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System," Processes, vol. 9, no. 5, pp. 1 14, 2021.

[5] Mahbooba B., Sahal R, Alosaimi W., and Serrano M., "Trust in Intrusion Detection Systems: an Investigation of Performance Analysis for Machine Learning and Deep Learning Models," Complexity, 2021.

[6] Jan S., Ahmed S., Shakhov V., and Koo I., "Toward a Lightweight Intrusion Detection System for the Internet of Things," IEEE Access, vol. 7, pp. 42450-42471, 2019.

[7] Su T., Sun H., Zhu J., Wang S., and Li Y., "BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset," IEEE Access, vol. 8, pp. 29575-29585, 2020.

[8] Dutta V., Chora´s M., Kozik R., and Pawlicki M., "Hybrid Model for Improving the Classification Effectiveness of Network Intrusion Detection," in Proceeding of the Computational Intelligence in Security for Information Systems Conference, 2019.

[9] Li Z. and Qin Z., "A Semantic Parsing Based LSTM Model for Intrusion Detection," in proceedings of International Conference on Neural Information Processing, Cambodia, pp. 600-609, 2018.

[10] Moustafa N. and Slay J., "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems," in Proceedings of Military Communications and Information Systems Conference (MilCIS), Canberra, pp. 1-6, 2015.

[11] Surakhi O., García A., Jamoos M., and Alkhanafseh M., "A Comprehensive Survey for Machine Learning and Deep Learning Applications for Detecting Intrusion Detection," in 22nd International Arab Conference on Information Technology, Muscat, pp. 1-13, 2021.

[12] School of Engineering and Information Technology, UNSW, Australia. ADFA Linux data set (ADFA-LD) cyber security benchmark dataset,

http://www.cybersecurity.unsw.adfa.edu.au/ADFA%20IDS%20Datasets, 2021.

[13] Shahriar M., Haque N., Rahman M., and Alonso M., “G-IDS: Generative Adversarial Networks Assisted Intrusion Detection System,” in Proceedings of IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, pp. 376-385, 2020.

[14] Garcia-Teodoro P., Diaz-Verdejo J., Maciá Fernández G., and Vázquez E., “Anomaly-Based Network Intrusion Detection: Techniques Systems and Challenges,” Computers Security, vol. 28, no. 1-2, pp. 18-28, 2009.

[15] Liu H. and Lang B., “Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey,” Applied Sciences, vol. 9, no. 20, pp. 1-28, 2019.