Indian Institute of Technology Gandhinagar

Anomaly Detection in Network Security by AI Algorithms

CS 433 Final Report

Members

Aaryan Darad (21110001)
Abhay Kumar Upparwal (21110004)
Aman Singh (21110020)
Disha Chopra (21110057)

Under the guidance of

Prof. Sameer Kulkarni

CONTENTS

Github Repository link :

https://github.com/BinaryBeast-007/AI_for_Network_Security/tree/main

Streamlit Video Link :

https://drive.google.com/drive/folders/1QzMBKPM6A1o9dFjF5bA0ye31ihiJVvrn?usp=sharing

OVERVIEW

In our investigation of anomalies in network security, we focused on identifying various types of attacks within the network. Our initial attempt at feature selection using a random approach on the CIC-IDS2017 dataset was hindered by computational limitations, preventing the execution of all intended algorithms. As an alternative, we turned our attention to the NSL-KDD dataset, which is more compact, and employed feature selection through Random Forest (RF).
Subsequently, we applied machine learning algorithms, including Random Forest, Logistic Regression, and Naive Bayes, to analyse the NSL-KDD dataset. Moreover, to enhance our anomaly detection capabilities, we implemented advanced techniques involving neural networks. Specifically, we explored the effectiveness of 1D Convolutional Neural Networks (CNN) and Variational Autoencoders, both of which exhibited promising results in identifying anomalies within the network. This multi-faceted approach allowed us to gain insights into various aspects of anomaly detection, ranging from traditional machine learning methods to cutting-edge neural network architectures.

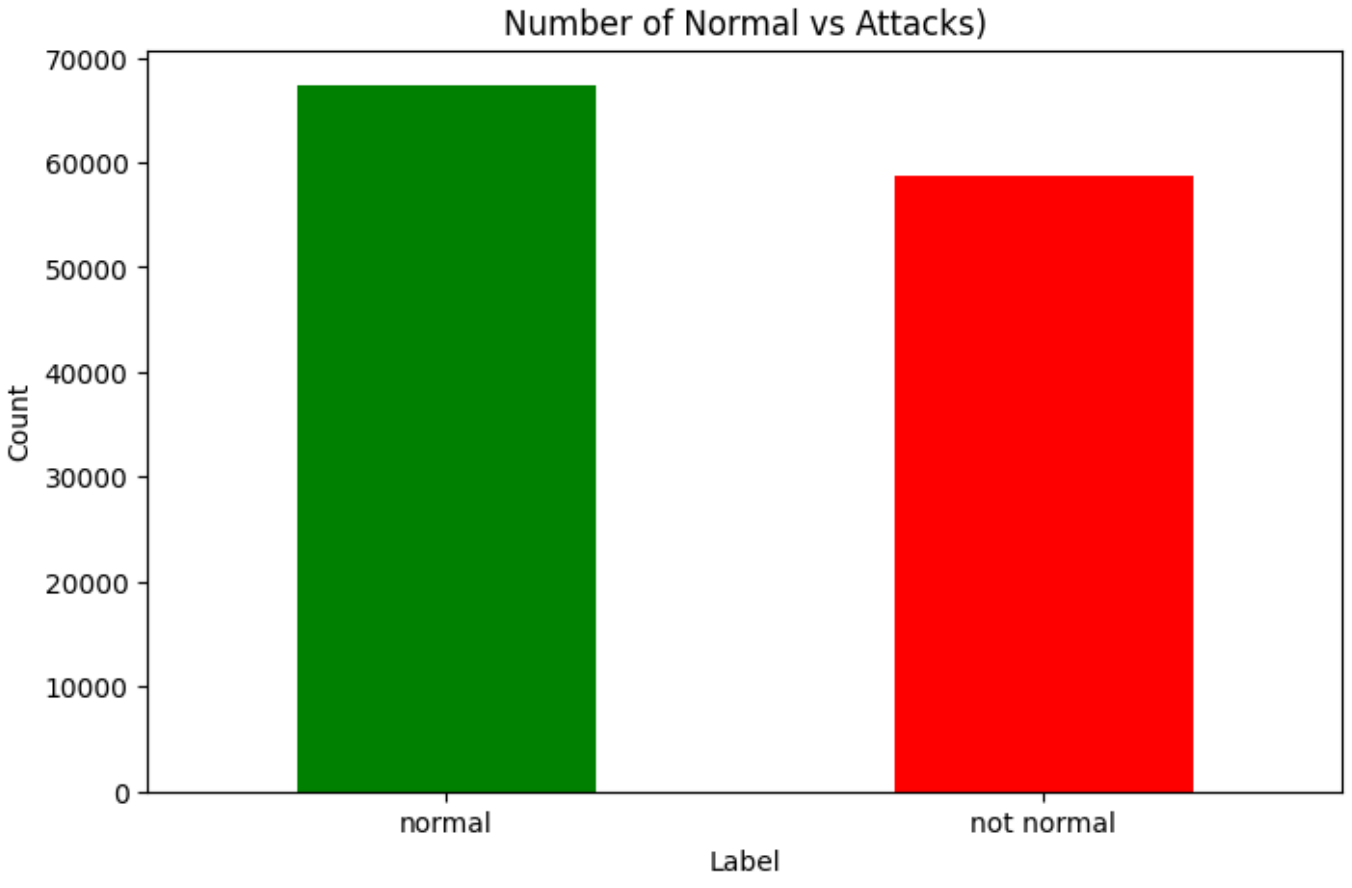
We tried implementing anomaly detection on the (CIC-IDS2017) dataset implementing various ML algorithms, feature selection using random forest. We first tried on our local CPU. Then we tried on google colab T4-Gpu. But due to less processing power, we couldn't train all the models. Then we tried a shorter dataset NSL-KDD.

DATASET

The dataset contains 125973 stream records. The distribution of these stream records can
be seen below. The first step in the pre-processing process will be to delete these unnecessary records. After preprocessing the dataset had 41473 rows × 49 columns. This dataset was almost well cleaned.

Statistics of redundant records in the KDD train set
Original records | Distinct records | Reduction rate
Attacks: 3,925,650 | 262,178 | 93.32%
Normal: 972,781 | 812,814 | 16.44%
Total: 4,898,431 | 1,074,992 | 78.05%

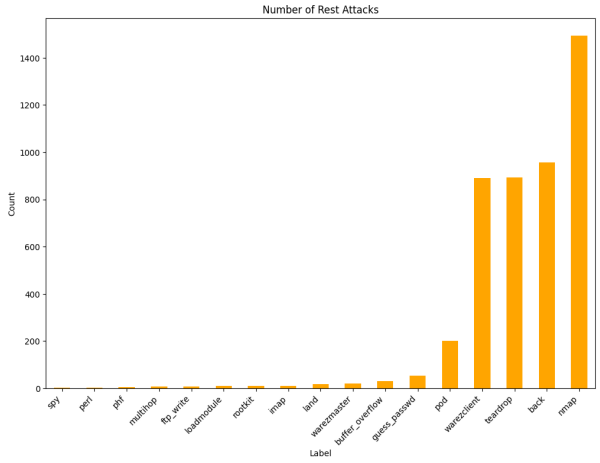
Statistics of redundant records in the KDD test set
Original records | Distinct records | Reduction rate
Attacks: 250,436 | 29,378 | 88.26%
Normal: 60,591 | 47,911 | 20.92%
Total: 311,027 | 77,289 | 75.15%

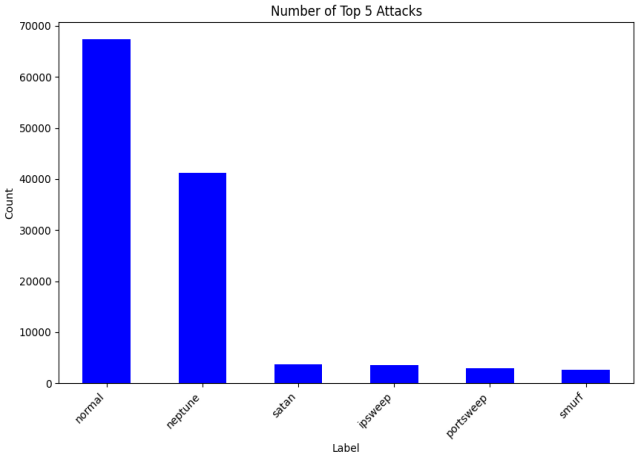


22 types of attacks found in the dataset are:

- [11] Normal (OriginalLabel: normal): This label indicates normal network behavior, i.e., non-malicious activity.
- [9] Neptune (OriginalLabel: neptune): Denial of Service (DoS) attack. It involves overwhelming the target system with a high volume of traffic, rendering it unavailable.
- [21] Warezclient (OriginalLabel: warezclient): Unauthorized access related to the use of illegal software distribution (warez).
- [5] Ipsweep (OriginalLabel: ipsweep): Network scanning or probing activity to identify active hosts on a network.
- [15] Portsweep (OriginalLabel: portsweep): Similar to Ipsweep, it involves systematically scanning a range of ports on a target system to identify open ports.
- [20] Teardrop (OriginalLabel: teardrop): Denial of Service (DoS) attack involving sending fragmented packets to crash the target system.
- [10] Nmap (OriginalLabel: nmap): Network scanning tool often used for reconnaissance.
- [17] Satan (OriginalLabel: satan): Network scanning tool similar to Nmap, used for probing and reconnaissance.
- [18] Smurf (OriginalLabel: smurf): Denial of Service (DoS) attack involving sending ICMP Echo Request (ping) packets to a broadcast address, causing amplification.
- [14] Pod (OriginalLabel: pod): Denial of Service (DoS) attack involving sending a flood of packets to the target to consume its resources.
- [0]Back (OriginalLabel: back): Unauthorized access often related to a backdoor or remote control of a system.
- [3] Guess_passwd (OriginalLabel: guess_passwd): Unauthorized access involving repeated login attempts to guess passwords.
- [2] Ftp_write (OriginalLabel: ftp_write): Unauthorized access involving writing or modifying files via FTP.
- [8] Multihop (OriginalLabel: multihop): Network activity involving multiple hops or intermediate systems.
- [16] Rootkit (OriginalLabel: rootkit): Malicious software designed to gain unauthorized access to a system and hide its presence.
- [1] Buffer_overflow (OriginalLabel: buffer_overflow): Exploiting a software vulnerability to overflow a buffer and execute arbitrary code.
- [4] Imap (OriginalLabel: imap): Unauthorized access or activity related to the Internet Message Access Protocol.

- [22] Warezmaster (OriginalLabel: warezmaster): Similar to Warezclient, indicating unauthorized access related to illegal software distribution.
- [13] Phf (OriginalLabel: phf): Unauthorized access involving exploiting the PHF CGI vulnerability.
- [6]Land (OriginalLabel: land): Denial of Service (DoS) attack involving sending spoofed packets with the same source and destination.
- [7]Loadmodule (OriginalLabel: loadmodule): Unauthorized access involving loading or executing a module on a system.
- [19] Spy (OriginalLabel: spy): Malicious activity related to spying or unauthorized information gathering.
- [12] Perl (OriginalLabel: perl): Unauthorized access or activity related to the Perl programming language.





From our previous attempt we learned about feature importance in training on these datasets. Then we used binary classification which classifies all streams of packets into two classes, normal or anomaly.

However, some 7 features (Flow ID, Source IP, Source Port, Destination IP, Destination Port, Timestamp, External IP) must not be included in the calculation, when the weight of importance is calculated. Although these features are used in classical approaches, it is possible that an attacker would prefer not to use well-known ports to escape control or to circumvent operating system constraints or he can use generated / fake IP addresses. Also, many ports are used dynamically, and many applications are transmitted over the same port. So, it can be misleading to use the port number.

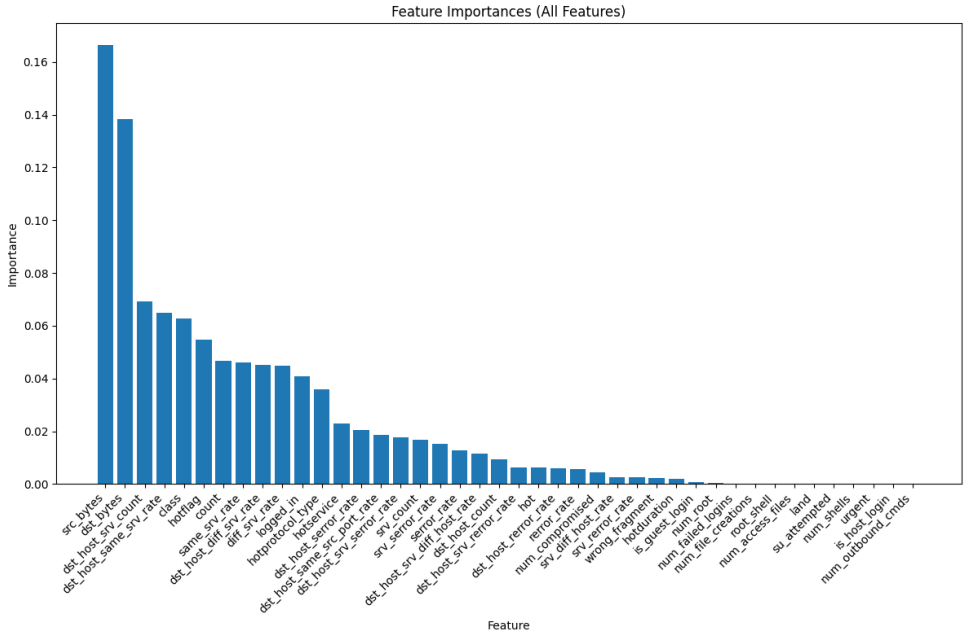
These features were excluded from the CIC-IDS2017 dataset during our initial attempt and were not explicitly present in the NSL-KDD dataset already.

Now we did our further analysis on the NSL-KDD dataset.

In the first approach, the entire dataset is treated as a unified entity, classifying each packet as either normal or anomalous. In the second approach, the data is segmented based on different attack types, allowing for a more specific analysis tailored to each type of attack. This approach offers a nuanced understanding of the dataset by considering distinct attack categories.

Feature Importance for Binary Class

The Random Forest Regressor[58] class of Sklearn is used when importance weights of features are calculated. This algorithm creates a decision-forest. In this decision forest, each feature is given a weight of importance as to how useful they arverrverre in the construction of the decision-tree. When the process is finished, these important weights of features are compared and sorted. The sum of the importance weights of all the properties gives the total importance weight of the decision tree. The comparison of the score of any feature to the score of the whole tree gives information about the importance of that feature in the decision tree.



Possible explanation of the above graph and certain features getting larger importance can be:

src_bytes, dst_bytes, count, same_srv_rate, diff_srv_rate, logged_in, srv_count, serror_rate:

These features are related to the amount of data transferred, the number of connections, and the success or failure rates of connections. In network security, abnormal patterns in data transfer, high connection counts, and unusual success or failure rates can indicate potential security threats.

dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_same_src_port_rate, dst_host_srv_serror_rate:

These features focus on the behaviour of the destination host, such as the count of connections, the rate of connections to the same service, and the error rates. Unusual patterns in these aspects might suggest anomalous behaviour on the destination host.

class:

The "class" feature is a crucial target variable indicating whether a network connection is normal or an attack. Naturally, it has high importance as it is the variable the model is trying to predict.

hotflag, hotprotocol_type, hotservice, hotduration:

These features are engineered or derived features. They might capture specific patterns or combinations of other features that are indicative of attacks. Feature engineering is crucial in improving the model's ability to capture complex relationships in the data.

srv_serror_rate, srv_diff_host_rate, srv_rerror_rate:

These features focus on the behaviour of the server. Unusual server error rates or differences in host rates might indicate abnormal server behaviour, potentially signalling an attack.

rerror_rate, dst_host_srv_diff_host_rate, dst_host_count, dst_host_srv_rerror_rate, hot, dst_host_rerror_rate, num_compromised, srv_diff_host_rate:

These features represent various aspects of connection errors, counts, or server behaviour. In network security, anomalies in these features could be indicative of attacks or malicious activities.

is_guest_login, num_root, num_failed_logins, num_file_creations, root_shell, num_access_files, land, su_attempted, num_shells, urgent, is_host_login, num_outbound_cmds:

These features relate to user activity, login attempts, file creations, and various system-level activities. Anomalies in these features may signal unauthorised access or malicious activity.

Implementation of Machine Learning Algorithms for Binary Classification

Random Forest:

```
Random Forest for Binary Classification:
Accuracy: 0.999920617910187
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     58630
           1       1.00      1.00      1.00     67343

    accuracy                           1.00    125973
   macro avg       1.00      1.00      1.00    125973
weighted avg       1.00      1.00      1.00    125973
```

Naive Bayes:

```
Naive Bayes for Binary Classification:
Accuracy: 0.5374326244512713
Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.01      0.03     58630
           1       0.54      0.99      0.70     67343

    accuracy                           0.54    125973
   macro avg       0.58      0.50      0.36    125973
weighted avg       0.58      0.54      0.39    125973
```

Logistic Regression:

```
Logistic Regression for Binary Classification:
Accuracy: 0.8709167837552492
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.86      0.86     58630
           1       0.88      0.88      0.88     67343

    accuracy                           0.87    125973
   macro avg       0.87      0.87      0.87    125973
weighted avg       0.87      0.87      0.87    125973
```

AdaBoost

```
Adaboost Classifier for Binary Classification:
Accuracy: 0.99199828534686O1
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     58630
           1       0.99      0.99      0.99     67343

    accuracy                           0.99    125973
   macro avg       0.99      0.99      0.99    125973
weighted avg       0.99      0.99      0.99    125973
```

KNN

```
KNN Classifier for Binary Classification:
Accuracy: 0.9972613179014551
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     58630
           1       1.00      1.00      1.00     67343

    accuracy                           1.00    125973
   macro avg       1.00      1.00      1.00    125973
weighted avg       1.00      1.00      1.00    125973
```

MLP

```
MLP Classifier for Binary Classification:
Accuracy: 0.9682233494478976
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.97     58630
           1       0.96      0.98      0.97     67343

    accuracy                           0.97    125973
   macro avg       0.97      0.97      0.97    125973
weighted avg       0.97      0.97      0.97    125973
```

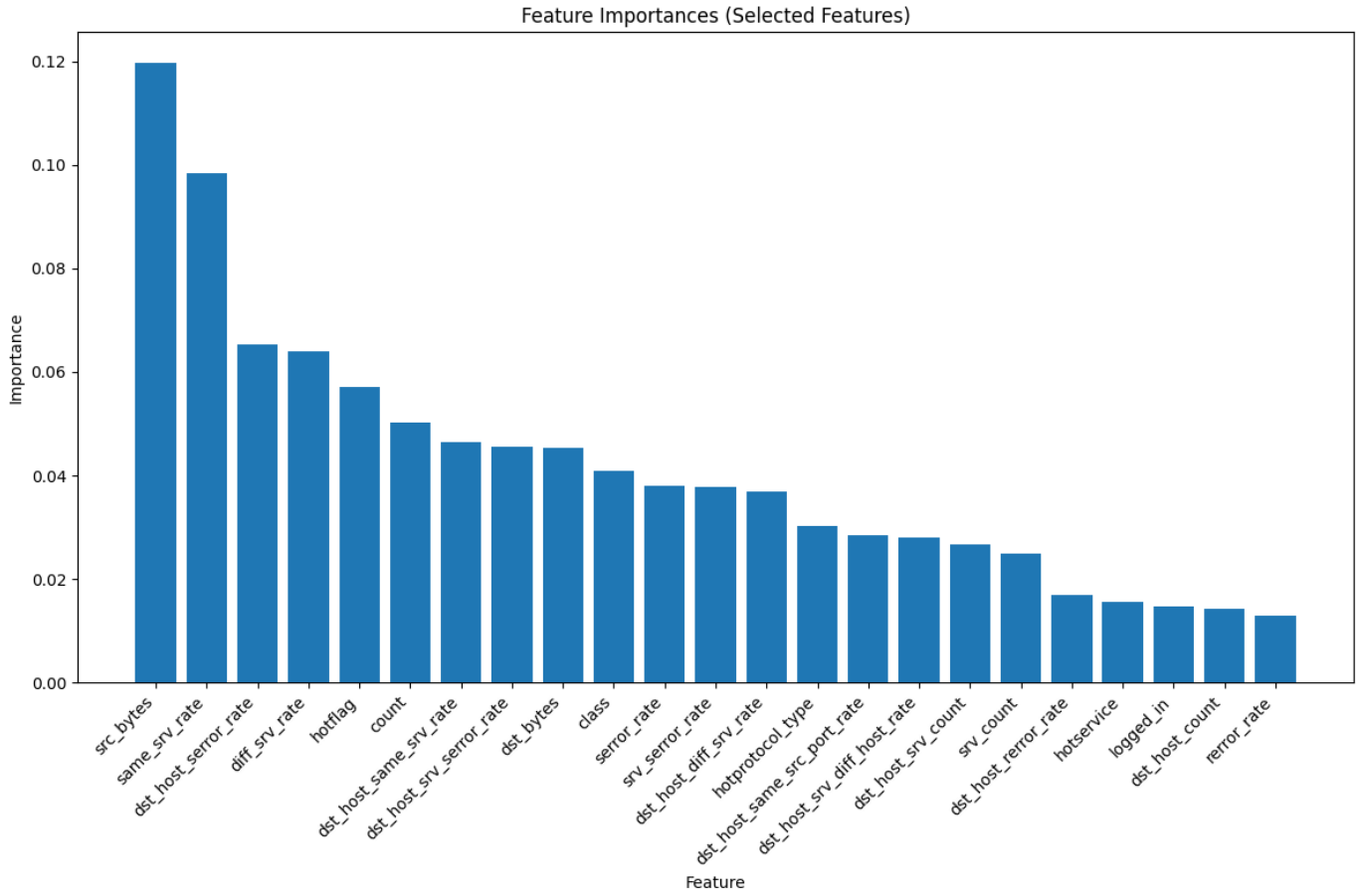
QDA

```
QDA Classifier for Binary Classification:
Accuracy: 0.9824724345693124
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98     58630
           1       0.98      0.99      0.98     67343

    accuracy                           0.98    125973
   macro avg       0.98      0.98      0.98    125973
weighted avg       0.98      0.98      0.98    125973
```

Isolation Forest

```
              precision    recall  f1-score   support

           0       0.46      0.92      0.61      2365
           1       0.29      0.03      0.05      2674

    accuracy                           0.45      5039
   macro avg       0.37      0.48      0.33      5039
weighted avg       0.37      0.45      0.31      5039
```

Feature Selection for Multiclass Classification:



We observed that there is some variation in the result compared to that of binary class. The differences in feature importance between binary and multiclass classification can be attributed to variations in class imbalance, class separability, data complexity, feature correlation, model sensitivity, dataset characteristics, and feature engineering. Binary classification involves distinguishing between two classes, while multiclass classification deals with more than two. The complexity of multiclass problems and the presence of diverse classes can lead to different feature importance rankings compared to binary scenarios. Additionally, how features interact with classes and the specific characteristics of the dataset play key roles in determining feature importance.

Implementation of Machine Learning Algorithms for Multiclass Classification

Random forest

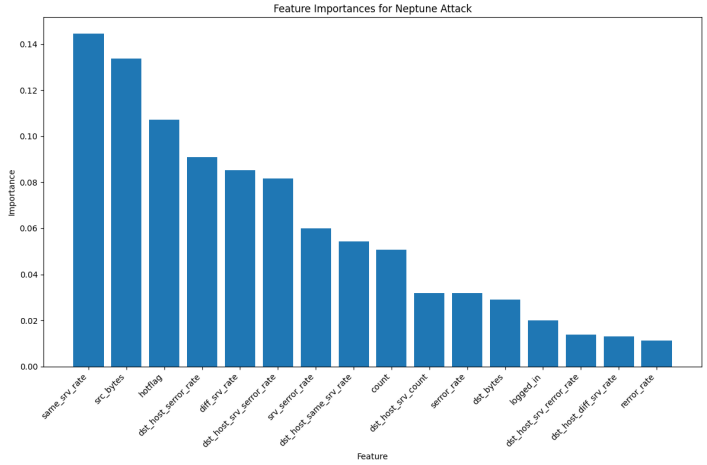
```
Random Forest for Multi-Class Classification:
Accuracy: 0.9986902163127604
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       185
           1       1.00      0.44      0.62         9
           2       0.00      0.00      0.00         0
           3       1.00      0.82      0.90        11
           4       1.00      1.00      1.00         1
           5       1.00      0.99      1.00       733
           6       0.50      0.33      0.40         3
           9       1.00      1.00      1.00      8228
          10       0.99      0.99      0.99       313
          11       1.00      1.00      1.00     13422
          12       0.00      0.00      0.00         1
          13       1.00      1.00      1.00         1
          14       1.00      0.93      0.96        43
          15       1.00      0.99      1.00       573
          16       0.00      0.00      0.00         1
          17       1.00      0.99      1.00       738
          18       1.00      1.00      1.00       534
          19       0.00      0.00      0.00         1
          20       1.00      1.00      1.00       188
          21       1.00      1.00      1.00       202
          22       1.00      0.88      0.93         8

    accuracy                           1.00     25195
   macro avg       0.78      0.73      0.75     25195
weighted avg       1.00      1.00      1.00     25195
```

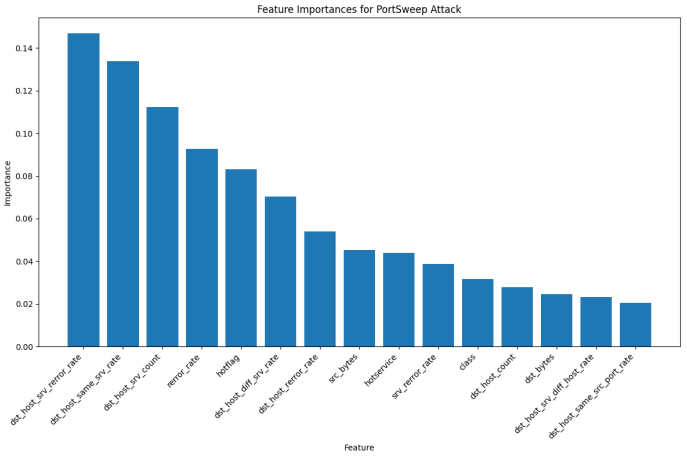
The above figure shows the precision and recall for each attack (22 attacks and 1 normal) on the whole dataset. The above figure for each algorithm is shown in the colab notebook. This detailed description could be used to identify which type of attack can be judged best by which type of algorithm.

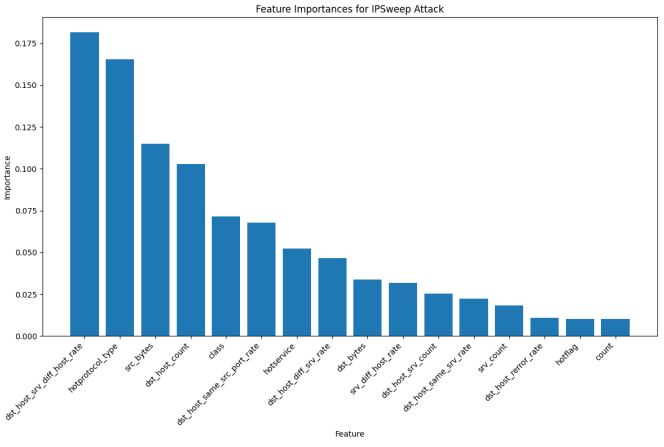
Other algorithms used are:

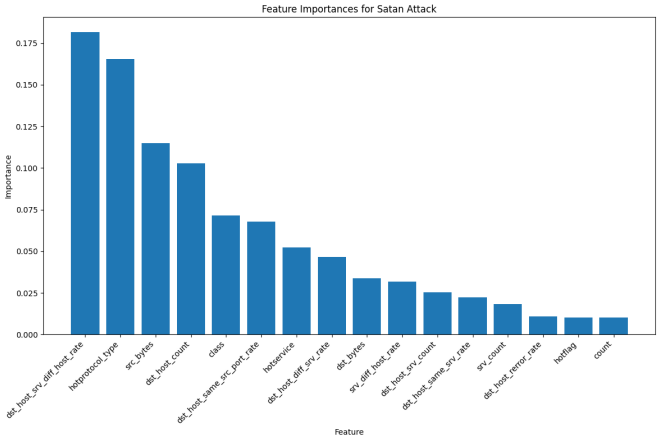
Naive Bayes : Accuracy: 0.43226830720381026
Logistic Regression : Accuracy: 0.8452470728319111
Adaboost Classifier : Accuracy: 0.8448501686842628
KNN Classifier : Accuracy: 0.9972613179014551

Next we also did find feature importance for 5 most found attacks in the dataset. Through this we can find that for each attack what features were important to detect that type of attack. We can use this data in cases where we need to find particular types of attacks in the packet streams and capture just those important features. By focusing on these essential features, we not only enhance the precision of attack detection but also have the potential to streamline the dataset, reducing unnecessary data and overall size. This approach contributes to a more efficient and targeted analysis of network traffic.









IPSweep Attack:
Type: Reconnaissance Attack
Description: IPSweep is a type of network scanning attack where an intruder systematically probes a range of IP addresses to identify live hosts on a network. This reconnaissance activity helps attackers map out potential targets for further exploitation.

Neptune Attack:
Type: Denial-of-Service (DoS) Attack
Description: Neptune is a variant of a DoS attack that overwhelms a computer system or network by flooding it with a massive volume of traffic. The goal is to exhaust the target's resources, causing it to become slow or unresponsive.

Satan Attack:
Type: Network Scanning and Enumeration Attack
Description: Satan is a network scanning tool that attackers use to identify vulnerabilities in computer systems. It performs extensive scans to discover open ports, services, and weaknesses in network security, aiding attackers in planning subsequent attacks.

Smurf Attack:
Type: Amplification Attack
Description: In a Smurf attack, the attacker sends a large volume of Internet Control Message Protocol (ICMP) echo request packets to an IP broadcast address. The requests are crafted to have a spoofed source address, causing numerous devices on the network to respond to the target, overwhelming it with traffic.

PortSweep Attack:
Type: Reconnaissance Attack
Description: PortSweep is a type of network scanning attack where an intruder systematically probes a range of ports on multiple machines to find open ports. This reconnaissance activity helps attackers identify potential points of entry or vulnerabilities in a network.

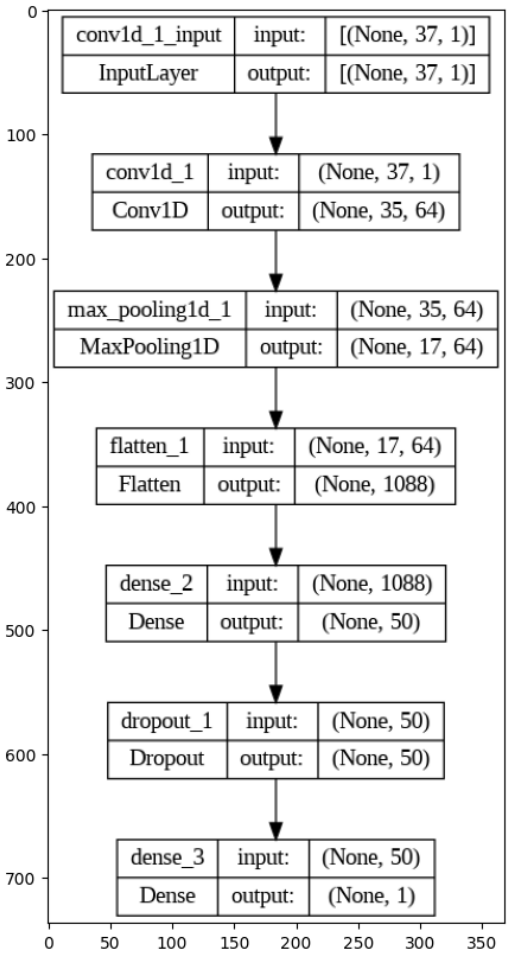
Implementation of Neural Networks on Binary classification

1D Convolutional Neural Networks

We explore the application of a 1D Convolutional Neural Network (1D CNN) for anomaly detection in network traffic data. The 1D CNN is a specialised neural network architecture designed for processing sequential data, making it suitable for detecting anomalies in time-series data.

In our model, there are 64 filters. Each filter learns different features from the input data.
kernel_size(3) defines the size of the convolutional window. It indicates the number of adjacent elements used by each filter during the convolution operation. A kernel size of 3 means that each filter looks at three consecutive elements at a time.

activation='relu': This parameter specifies the activation function applied to the output of each convolutional operation. ReLU (Rectified Linear Unit) is a commonly used activation function that introduces non-linearity to the model.



1D CNN Neural Network Diagram

	precision	recall	f1-score	support
0	0.98	0.98	0.98	2365
1	0.98	0.98	0.98	2674
accuracy			0.98	5039
macro avg	0.98	0.98	0.98	5039
weighted avg	0.98	0.98	0.98	5039

In this case, the precision is 0.98, which means that 98% of the anomalies that the CNN predicted are actually anomalies.

The recall is the proportion of true positives out of all actual positives. In this case, the recall is also 0.98, which means that the CNN is able to detect 98% of all anomalies in the network.

The F1-score is a harmonic mean of precision and recall. It is a good measure of overall performance when both precision and recall are important. In this case, the F1-score is also 0.98, which indicates that the CNN has very good overall performance.

The accuracy is the proportion of all correctly predicted samples, including both positives and negatives. In this case, the accuracy is 0.98, which means that the CNN is able to correctly predict 98% of all samples in the dataset.

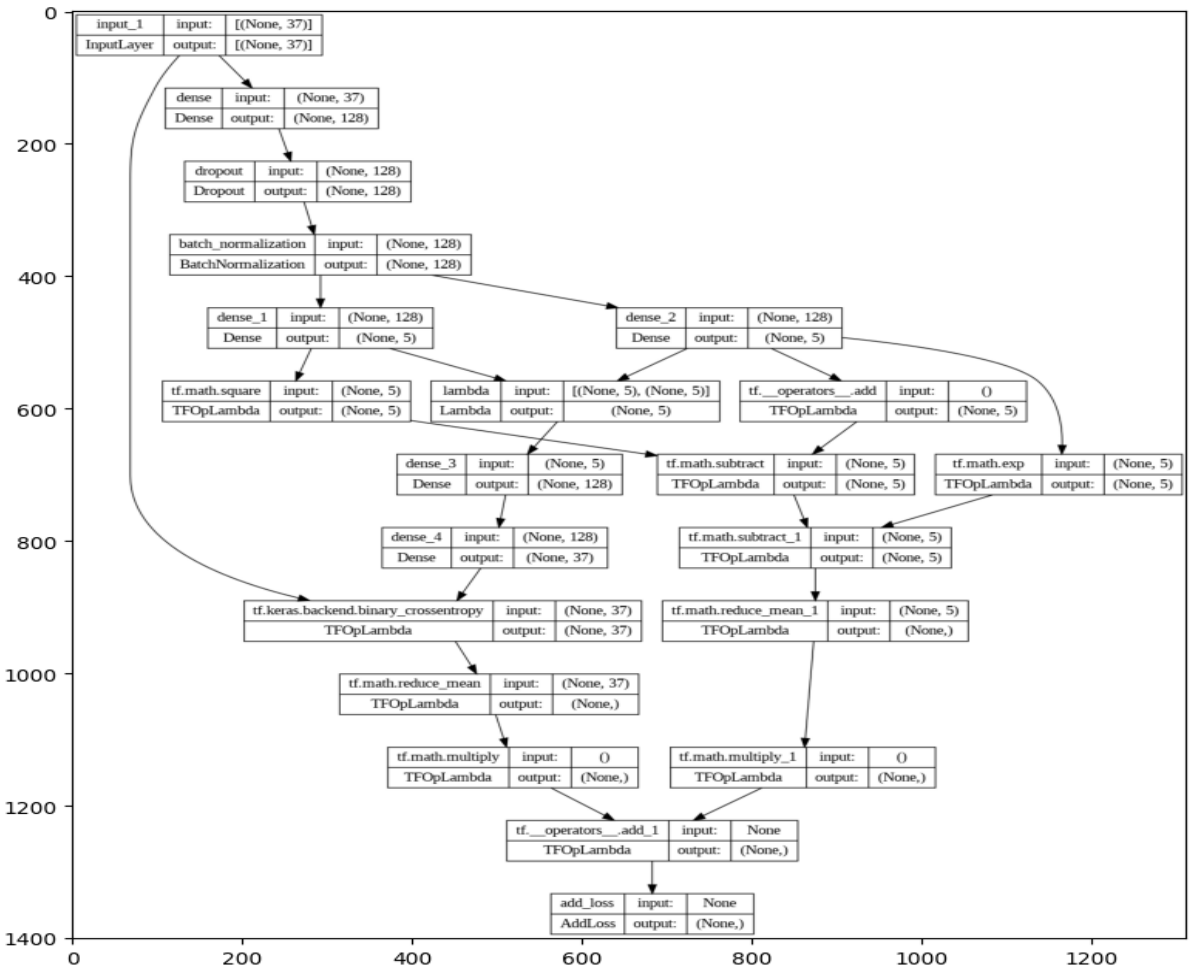
VARIATIONAL AUTOENCODERS

Variational Autoencoders (VAEs) are a type of generative model used for unsupervised learning, and they can be applied to anomaly detection in networks. Here's an overview of how VAEs can be used for this purpose:

Autoencoder Architecture:

- Encoder: The first part of a VAE is the encoder network, which takes input data and maps it to a latent space. In the context of network data, this could be the representation of normal network behaviour. The encoder compresses the input data into a lower-dimensional latent space, capturing its essential features.
- Latent Space: The latent space is a probabilistic representation of the input data. Unlike traditional autoencoders, VAEs model the latent space as a probability distribution, typically assuming a Gaussian distribution. This probabilistic nature allows the model to generate new, similar data points.

- Decoder: The decoder takes a point from the latent space and reconstructs the original input data. The goal of the decoder is to generate data that closely resembles the input data.



We obtain the following results:

	precision	recall	f1-score	support
0	0.98	0.82	0.89	2365
1	0.86	0.98	0.92	2674
accuracy			0.91	5039
macro avg	0.92	0.90	0.90	5039
weighted avg	0.91	0.91	0.91	5039

The precision of the model is 0.98, meaning that 98% of the anomalies detected by the model were actually anomalies. The recall of the model is 0.82, meaning that the model

detected 82% of the anomalies in the test dataset. The F1-score of the model is 0.89, which is a good overall measure of the performance of the model.

Comparing all methods and their accuracy

Methods	*Accuracy*
Random Forest	**99.99%**
Naive Bayes	53.74%
Logistic Regression	87.09%
AdaBoost	99.19%
KNN	99.72%
MLP	96.82%

QDA	98.24%
Isolation Forest	~45%
1D CNN	~98%
Variable AutoEncoder	~91%

CONCLUSION

In the course of this research, the primary objective was to employ machine learning methodologies for the detection of anomalous patterns within a computer network. The selection of the NSL-KDD dataset was predicated on its contemporaneity, inclusion of diverse attack types, and representation of various network protocols such as Mail services, SSH, FTP, HTTP, and HTTPS.

To discern the most influential features for machine learning, the Random Forest Regressor algorithm was leveraged to compute feature weights. Two distinct approaches were applied for this calculation: importance weights were computed individually for each attack type, and second, a collective group of all attacks was considered to determine common properties essential for all attack types.

Subsequently, seven machine learning algorithms were applied to the dataset, and their performance was assessed using the F-measure, a metric ranging from 0 to 1. **The algorithmic results were as follows: Naive Bayes: 0.53, QDA: 0.98, Random Forest: 0.99, AdaBoost: 0.99, MLP: 0.96, and K Nearest Neighbours: 0.99.**

Additionally, two neural network models were implemented, yielding the following results: 1D CNN: 0.98, Variable AutoEncoder: 0.91.

Insights from the study suggest that ensemble methods, such as Random Forest, AdaBoost, and K Nearest Neighbours, demonstrated superior performance in detecting anomalous patterns within the network. Moreover, the application of neural network models, specifically 1D CNN, exhibited promising results, showcasing their potential efficacy in handling diverse types of network attacks. The variation in performance among different algorithms underscores the importance of selecting appropriate methodologies tailored to the intricacies of the dataset and the nature of the anomalies being addressed.

FUTURE WORK

This research endeavour underscores its openness to future enhancements, and this section outlines potential areas for improvement. The study employed a dataset derived from network flow, serving as both training and test data. Regrettably, this approach's practical viability in real-world systems is limited. To address this concern, the implementation of a module capable of capturing authentic network data and seamlessly integrating it with machine learning algorithms is proposed.

Moreover, the study revealed a limitation in the independent application of various machine learning methods, yielding experimental results with weak practical applicability in real-world scenarios. To overcome this limitation, a prospective solution involves the design of a multi-layered hierarchical machine learning structure. This structured approach offers the potential to optimise resource utilisation, including time, CPU power, and memory.

For instance, in a two-tiered structure, the initial layer could be constructed using computationally efficient algorithms like Naive Bayes or QDA. This allows continuous and cost-effective observation of network traffic. Upon detecting any anomaly, the system would transmit the information to an upper layer comprising algorithms with higher performance capabilities. This upper layer functions by establishing a decision mechanism.

In the subsequent step of anomaly detection, the information is further transmitted to an upper layer encompassing algorithms of advanced capabilities, such as ID3, AdaBoost, and KNN. This final layer, constituting the determination mechanism, is tasked with making precautionary decisions to safeguard the network against potential attacks. This multi-layered hierarchical structure aims to enhance the practical applicability and efficiency of machine learning methods in real-life network security scenarios.

REFERENCES

[1]Kahramankostas, "Anomaly Detection in Networks Using Machine Learning," GitHub, Available at: https://github.com/kahramankostas/Anomaly-Detection-in-Networks-Using-Machine-Learning/tree/master (Accessed: 14 November 2023).

[2] "Evaluation of Machine Learning Algorithms for Anomaly Detection," *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9138871

[3] MDPI. (2022). "Title of the Article," Applied Sciences, 12(16), 7986. Available at: https://www.mdpi.com/2076-3417/12/16/7986.

[4] BinaryBeast-007. (2022). "Title of the Repository," GitHub. Available at: https://github.com/BinaryBeast-007/AI_for_Network_Security/tree/main.