AdaBoost IDS to detect Zero Day attacks and reduce false positives

MSc Research Project
MSc Cyber Security

Benetto George
X21124485

School of Computing
National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Benetto George		
Student ID:	X21124485		
Programme:	MSc Cyber Security	**Year:**	2022 - 2023
Module:	MSc Research Project		
Supervisor:	Imran Khan		
Submission Due Date:	15-12-22		
Project Title:	AdaBoost IDS to detect Zero Day Attacks and reduce false positives		
Word Count:	25 ……………………………………… **Page Count**……6027………………………………………….……..		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Benetto George ………………………………………………………………………………………………………

Date: 14-12-2022 ………………………………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

AdaBoost IDS to detect zero-day attacks and to reduce False Positive rates

Benetto George
21124485
MSc in Cybersecurity
National College of Ireland

Abstract

Zero-day attacks are becoming more frequent nowadays. It is defined as when an attacker takes over a software or application with an exploit known to the developer, but the patch for the same is not available; the developer has zero days to fix it, hence it is called zero-day attack. Preventing such an attack is extremely crucial for organizations. The present-day methods use an IDS (Intrusion Detection System) to detect such an attack. They are efficient in doing the same but produce high rate of false positives which wastes a lot of time because a non-attack is labelled as an attack. Therefore, accurate and fast IDS(s) are needed, which in turn saves time in detecting them. It also makes the company safe and ensures that the necessary steps are taken immediately to remedy it. This paper proposes a novel approach in detecting zero-day attacks and reducing the false positive rates with an AdaBoost IDS.

Keywords: AdaBoost algorithm, Zero-Day attack, Feature Selection, Accuracy, False Positive rate

1 Introduction

In recent years, the number of cyber-attacks has increased dramatically. Zero-day attacks are the most damaging of these. It takes months, often years, to find and fix flaws in them. The term "zero-day attack" refers to the idea that there are “zero-days" to fix the vulnerability because the developer just noticed the attack. These can take the form of botnet attacks, SSH brute force attacks, DOS, DDOS, intrusions, SQL injection, and many more. All of these are designed to slow down your network, extract sensitive data from your system, and commit data or money theft. These attacks can exploit vulnerabilities in operating systems, office programs, web browsers, and open-source systems, indicating that neither individuals nor businesses are truly immune from these threats. The most important of these, like Stuxnet, can jeopardize the weaknesses of Microsoft software and Iran's nuclear program .

Signature-based models are known to be fast and have high detection rates, but only known patterns can be found. Anomaly detection was performed using supervised learning techniques such as ANN, SVM, RNN, and decision trees, and unsupervised deep learning techniques such as the ZDAR system and autoencoder models. There is a significant percentage of false positives for unknown attack types, but a reasonable success rate for known attack types (Pitre et al., 2022).

Both false positives and false negatives have a negative impact on cyber security. False negatives endanger your system and expose it to malware when an attack is reported as non-attack. Newly introduced deep learning models have historically shown low false-negative rates. False positives, where non-attacks are misclassified as attacks, remain a challenge for this sector. This is problematic because it requires ongoing human involvement to ensure that the branded attack is an attack. Enterprises are forced to spend time and money on it, preventing the system from running at its fastest speed. Increased speed demands from both users and businesses are a challenge for the bulkier IDS models that have these issues (Pitre et al., 2022) .

The aim of this paper is to implement an AdaBoost IDS to effectively detect zero-day attacks while also minimizing the false positive rates. To achieve this, multiple models, feature selection techniques, weight adjustments, and several fine-tuning are used to minimize false positive attack detection rates while maintaining significant speed and false positive rates.

Research Question:

How effective is the AdaBoost algorithm in detecting zero-day attacks?

The current traditional intrusion detection systems are efficient but produce high rate of false positives. According to the researches performed (Moustafa et al., 2019; Rana and Sung, 2020; Tang et al., 2020), using AdaBoost algorithm in an IDS has been found to be highly efficient in detecting network intrusions because boosting method was able to drastically decrease the error rates. The aforementioned papers motivated me to use AdaBoost in my project rather than a machine learning algorithm that uses a single classifier as they show that the usage of AdaBoost algorithm in detecting cyber-attacks exhibit high accuracy and low false positive rates.

2 Literature Review

The related works have been categorized into the following domains:

2.1 Signature-Based Intrusion Detection Systems

Despite being an outdated paradigm, signature-based detection remains widely used because it can quickly and accurately identify known attacks. The paper (Holm, 2014) examines the usefulness of signature-based detection systems to identify current known and unidentified attacks.

The paper (Kumar and Sinha, 2021) has suggested signature extraction (HVA) for both high-volume attacks (LVA) and low-volume attacks. They also used flash detectors to reduce the number of false alarms. On the other hand, new zero-day attacks are difficult to intercept with signature-based detection systems and therefore require the incorporation of deep learning and machine learning modules.

2.2 Ensemble Learning based intrusion detection systems

Ensemble learning is an approach like machine learning which can give a much better outcome. AbaBoost is a boosting technique. Boosting is a method of merging distinctive low accuracy model which create high accuracy model. Adaboost is an ensemble learning algorithm which boosts the performance of any machine learning algorithm, has low generalization error and it can work with a broad range of classifiers since it is very easy to implement.

The AdaBoost Ensemble Network Intrusion Detection System (NIDS) (Moustafa et al., 2019) was built and analysed in a study using Decision Tree (DT), Naive Bayes (NB), and Artificial Neural Network (ANN) algorithms. This system can identify IoT attacks related to the application layer. This ensemble model was built using the NIMS botnet dataset and the UNSW-NB15 botnet dataset. Their research shows that the proposed model has an accuracy of 99.54% in detecting attacks on the UNSW-NB15 dataset and an accuracy of 98.29% in detecting attacks on the NIMS botnet dataset.

In a paper (Rana and Sung, 2020) , various approaches to ensemble learning have been explored, such as stacking, boosting, bagging and blending. In this study, I found that the boosting method could effectively eliminate the majority of significant errors by distinguishing between strong and weak learners.

There is a paper (Tang et al., 2020) that proposes a feature-rich AdaBoost system for identifying low-rate denial-of-service (LDoS) attacks. Network traffic was recorded at specified intervals and the resulting samples were evaluated using various statistical methods. Correlation values between features and class labels were obtained to select the best feature set. AdaBoost ensemble models were created using the best properties. They evaluated the performance of the model using the NS2 simulator and testbench, and the attack detection accuracy was 94.05% and 97.06%, respectively.

The paper (Shahraki et al., 2020) uses well-known intrusion detection datasets such as KDDCUP99, NSL-KDD, CICIDS2017, UNSW-NB15 and TRAAbID to perform a comparative analysis of different AdaBoost methods such as Real Adaboost, Gentle Adaboost and Modest Adaboost. In this study, the authors found that the Real AdaBoost and Gentle AdaBoost algorithms outperform the Modest AdaBoost method. However, the Modest AdaBoost algorithm was faster than the other AdaBoost algorithms.

To protect IoT networks (Anitha and Arockiam, 2021) from version number attacks, DIS flooding attacks, and DAO attacks, this study uses the AdaBoost ensemble model to create an ensemble IDS named Ada-IDS. It is then installed on the border router. The results show that the Ada-IDS ensemble model accurately detected these three types of attacks with an alert rate of 99.6%. Therefore, it works as an anomaly-based intrusion system.

The paper (Pallippattu Mathai, 2021) explains the malware detection capability of AdaBoost on Android data.

To identify botnet attacks in connected cars, the study (Rehman Javed et al., 2022) proposed an AdaBoost ensemble classifier. The AdaBoost approach used a decision tree algorithm as the base estimator and set the cluster size to 100. Compared to decision trees, probabilistic neural networks, and sequential minimum optimization, the AdaBoost classifier performed better. Their research showed that the AdaBoost classifier outperformed previous models, showing a true positive rate of 99.7% and an accuracy of 99.1%.

2.3 Machine Learning based intrusion detection systems

The paper (Heba et al., 2010) uses SVM for anomalous-based intrusion detection systems due to its robustness. But, this implementation does not constitute an efficient intrusion detection system.

The paper (Farnaaz and Jabbar, 2016) implements a Random Forest Classifier IDS to overcome overfitting. The research was only based on four types of attacks.

By developing a three-phase model, the paper (Goeschel, 2016) takes an innovative approach to reducing false alarms and increasing overall accuracy. The first phase uses the SVM classifier, the second phase uses the decision tree, and the third phase uses naive bayes. The combination of these three classifiers greatly improves the accuracy of IDS.

Detecting zero-day attacks using various supervised machine learning models has been the subject of extensive research. To train the model, the paper (Zhou and Pezaros, 2019) implements six machine learning classification models: Random Forest Classifier, Gauss Naive Bayes Classifier, Decision Tree Classifier, Multilayer Perceptron (MLP) Classifier, K-Nearest Neighbor (KNN) Classifier, and Quadratic discriminant analysis classifier. Tests have shown that the Decision Tree classifier provides the most accurate results for most attack types.

In a paper by Pitre (Pitre et al., 2022) provided a way to combat zero-day attacks and reduce false positive rates by combining feature selection techniques and fine-tuning datasets dedicated to false positive detection. These techniques were repeatedly tested with different optimizers and models and the results were compared. The study achieved 95.1% accuracy after fine tuning on the positives from Stage 1 of the experiment. This approach motivated me to pursue the two stage model to reduce the false positive rate.

2.4 Deep learning based intrusion detection systems

With the sophistication of threats, people are now using deep learning to detect intrusions.

Papers (Kwon et al., 2018) and (Kim et al., 2020) used a CNN model for classification. The first used three different internal CNN depths and concluded that the depth had little or no effect on the results. In the second study, they implemented a CNN model to identify DoS attacks and evaluated their effectiveness by comparing their performance to the RNN model.

The paper (Soltani et al., 2021) compared four deep novelty-based classifiers (DOC, DOC ++, OpenMax, autoSVM) and found that DOC ++ provided the most promising results. Thanks to this method, they were able to classify unidentified attacks.

2.5 Feature selection in intrusion detection systems

Due to the large number of features in most attack datasets, feature selection has become an important component of attack detection systems as it improves classification during dataset training by choosing a subset of relevant features.

The paper (Akashdeep et al., 2017) performs feature ranking using correlation and information gain algorithms. The correct feature subset is then selected based on this ranking.

A research (Pallaprolu et al., 2017) identifies the most distinctive features of the dataset by using the minimum redundancy maximum relevance (MRMR) feature selection technique on the Apache Spark platform. The drawback is that the research depends on the result of a single classifier which decreases the accuracy.

The paper (Research Scholar, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamilnadu, India. et al., 2019), combines ExtraTreesClassifier, Percentile, and KBest, which are feature selection methods. With feature selection, the model is much more accurate than it would be without it.

The paper (Lin et al., 2021) implements k-fold(k=10) cross validation to achieve the automatic selection of the ideal parameters and number of features, and it then employs three well-known feature selection metrics simultaneously: mutual information, chi-squared, and ANOVA F value.
Therefore, feature selection is essential to the intrusion detection system training process.

3 Research Methodology

The below figure shows the components of the system and proposed flow.

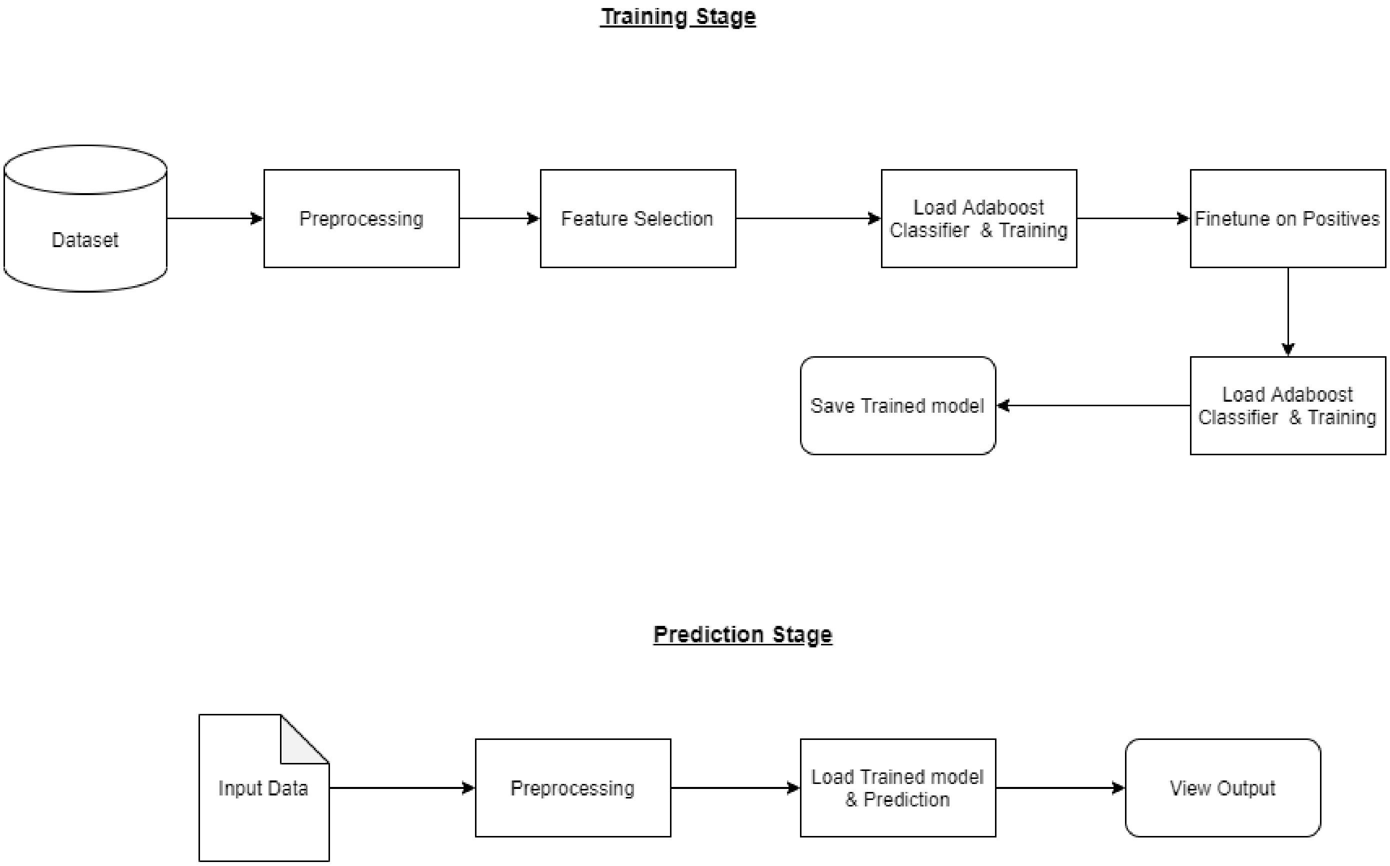


Figure 1: Proposed Flow of the System

A summary of the steps are explained below:

A. Training Stage

1. **Dataset**: A standard dataset which is called IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018) dataset of the day 02-15-2018 is used. The dataset is taken from Kaggle, which is a publicly available platform to download various datasets (“IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018),” n.d.).
2. **Preprocessing**: The dataset is preprocessed which involves removing unwanted columns, removing columns that contain only zeroes and removing infinite values.

3. **Feature Selection:** This is one of the most important processes in a training algorithm, as it allows you to choose the features from the dataset that best fit your model. Choosing a single feature can cause the model to be biased towards a particular feature and reduce accuracy. I tested three Ensemble feature selection techniques by searching the Ensemble algorithms that are suited for IDS(s) from the paper (Venkatarathinam et al., 2018), which, upon searching the internet were Chi square, Ensemble Extremely Randomized Trees classifier and Exhaustive Feature Selection. Chi square method was found to be the most effective and this was used in this project.
4. **Load AdaBoost Classifier and training:** The dataset with selected features is then trained with AdaBoost classifier.
5. **Finetune on the positives:** After testing a small portion of the dataset, a confusion matrix containing the false positive rate, true positive rate, false negative rate, and true negative rate is obtained. Since I need to reduce the false positive rate, a subset of the dataset is created consisting of the positives (TP and FP) from the original test.
6. **Save trained model:** The dataset subset is then again trained with AdaBoost classifier, and a final confusion matrix is generated with the reduced false positive rate. This trained model is then saved for future prediction.

B. Prediction stage

1. **Input data:** The trained model is loaded. The parameters are inputted by the user.
2. **Preprocessing:** The inputted details are converted to a dictionary which is in turn converted to a data frame.
3. **Load trained model and prediction:** The data frame is loaded to the model for prediction.
4. **Output:** The output which can be Malicious or Benign is viewed.

4 Design Specification

The algorithm I used in my implementation, called Adaboost, was developed by Yoav Freund and Robert Schapire in 1996 and uses a boosting technique. This is also an iterative ensemble method. The AdaBoost classifier combines many ineffective classifiers to create a strong classifier with a high level of accuracy. To better predict new observations, classifier weights are set and data samples are trained in separate iterations. For the Adaboost algorithm to work, two requirements must be met:

First, the classifier has to be interactively trained on different weighted training samples. Each training trial should then aim to minimize the training error to produce the best possible fit for the instance. Working of AdaBoost algorithm is as follows:[1]

1. AdaBoost initially chooses a training subset at random.
2. An AdaBoost machine learning model is iteratively trained by choosing a training set based on the accuracy of previous training.
3. It gives more weight to misclassified observations, so they are more likely to be classified correctly in the next round.
4. In addition, weights are assigned according to the accuracy of the trained classifier at each iteration. More weight is given to more accurate classifiers.
5. This process is repeated until the entire training set is completely filled or the specified maximum number of estimators is reached.
6. Vote for all the learning algorithms you have created to classify them.

Weak learners are integrated in the AdaBoost ensemble strategy to increase accuracy, which is accomplished repeatedly by addressing the faults of the weak classifier.

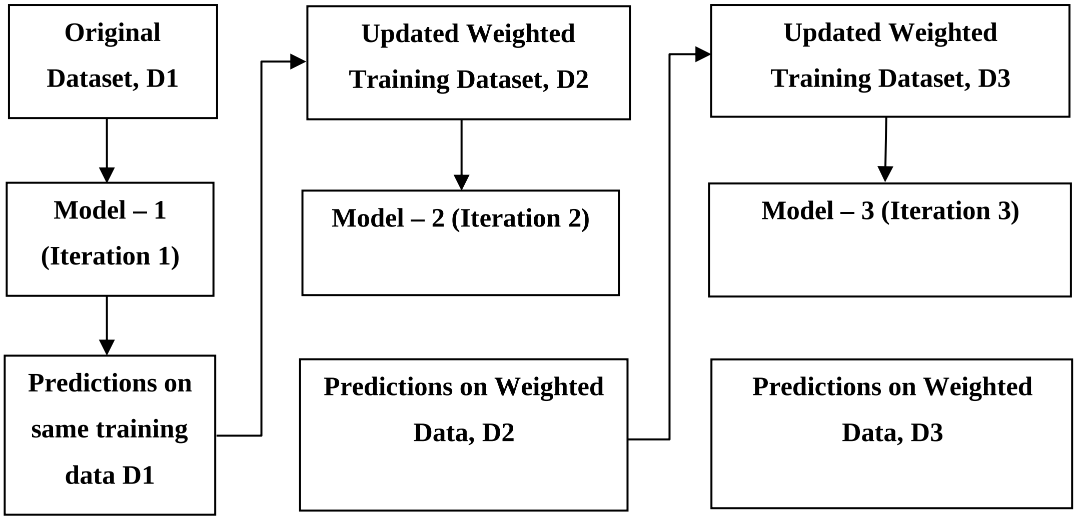

Original Dataset, D1
Model – 1 (Iteration 1)
Predictions on same training data D1
Updated Weighted Training Dataset, D2
Model – 2 (Iteration 2)
Predictions on Weighted Data, D2
Updated Weighted Training Dataset, D3
Model – 3 (Iteration 3)
Predictions on Weighted Data, D3

Figure 2: AdaBoost Classifier

[1] https://www.datacamp.com/tutorial/adaboost-classifier-python

4.1 System Specifications

The experiment was performed on a desktop with Windows 11 which had Python, Visual Studio and Anaconda software installed.

Desktop Specification:

- Processor: Intel i5 11th Gen
- 16 GB DDR4 RAM
- 512 GB SSD storage

To perform the data cleansing operations such as reading the dataset and removing infinite values, I used Python’s data science libraries Pandas and NumPy.

Software used:

- Windows 11 64-bit
- Python 3.10.5
- Microsoft Visual Studio Code v1.73.1

5 Implementation

5.1 Environment Setup

The project is written in Python programming language. Anaconda software is used to create the environment for machine learning and Visual Studio Code is used to run the code.

5.2 Dataset Description

The dataset is taken from Kaggle[2] which is a publicly available platform to download various datasets. It is downloaded in CSV format in which it consists of 80 columns, each of which is an IDS logging system entry. The important features in the dataset are Dst Port, Protocol, Flow Duration, Tot Fwd Pkts, Tot Bwd Pkts and Label.

[2] https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv

5.3 Package Installation

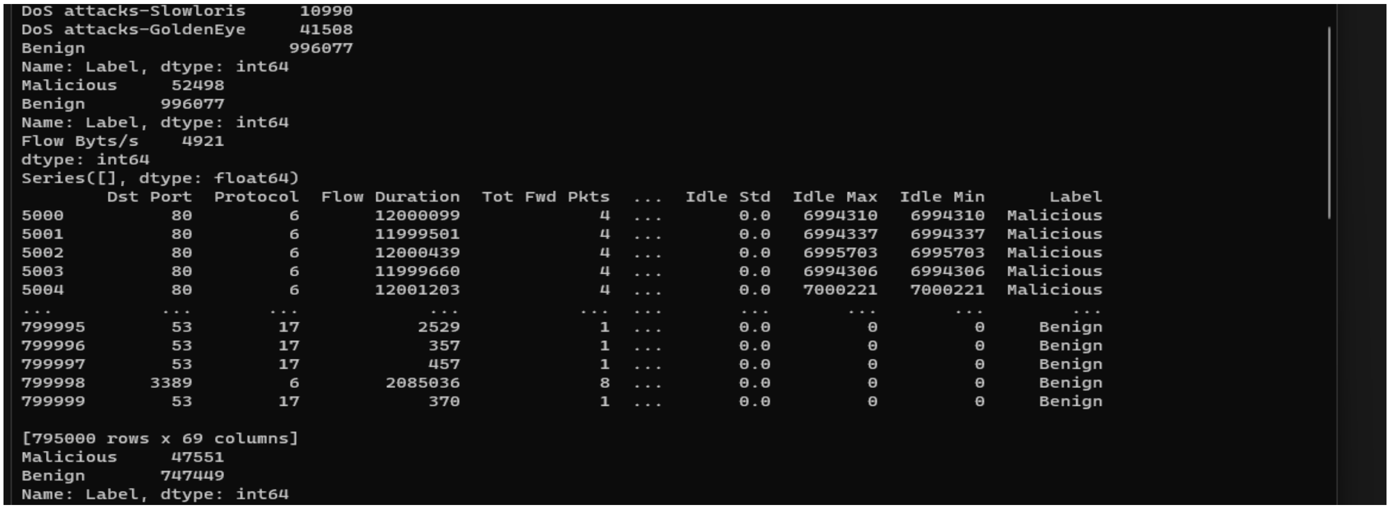
Here in this project, Anaconda software is also used, which is a distribution for Python programming languages for data science. It simplifies package management and deployment. An environment, named “**ml_env**” is created. It consists of the necessary machine learning python libraries that are custom installed which in turn help to run the whole code. A small number of libraries that are present in the environment are:

- Numpy: Used for array operations
- Sklearn: Used for selection of features and for train-test splitting
- Pandas: Used to read the dataset
- Matplotlib: Is used for data visualization

5.4 Training Stage

5.4.1 Dataset Preprocessing

This is the first step that is done in the training stage. The dataset is read and then number of occurrences of “Label”, number of Malicious and Benign entries is counted. The column entries which consists of attacks in the dataset which are “DoS attacks-Slowloris”, “DoS attacks-GoldenEye” are replaced with the value “Malicious”. An unwanted column named “Timestamp” is removed, because it is not a numeric value. Then, cleaning of the data is done by removing columns that contain only zeroes, unwanted columns, null values; which are specified as NaN(Not A Number) in python and infinite values using Numpy and Pandas libraries. A new data frame from row 5000 to 800000 is created to have a balance between Benign and Malicious Entries. Data division is done in which it is split into an Independent variable and a Dependent variable. The Dependent variable (model output) is taken as “Label”(y) and Independent variable (model input) is taken as all the features excluding label(x).



```
DoS attacks-Slowloris      10990
DoS attacks-GoldenEye      41508
Benign                    996077
Name: Label, dtype: int64
Malicious     52498
Benign       996077
Name: Label, dtype: int64
Flow Byts/s    4921
dtype: int64
Series([], dtype: float64)
        Dst Port  Protocol  Flow Duration  Tot Fwd Pkts  ...  Idle Std  Idle Max  Idle Min      Label
5000          80         6       12000099             4  ...       0.0   6994310   6994310  Malicious
5001          80         6       11999501             4  ...       0.0   6994337   6994337  Malicious
5002          80         6       12000439             4  ...       0.0   6995703   6995703  Malicious
5003          80         6       11999660             4  ...       0.0   6994306   6994306  Malicious
5004          80         6       12001203             4  ...       0.0   7000221   7000221  Malicious
...          ...       ...            ...           ...  ...       ...       ...       ...        ...
799995        53        17           2529             1  ...       0.0         0         0     Benign
799996        53        17            357             1  ...       0.0         0         0     Benign
799997        53        17            457             1  ...       0.0         0         0     Benign
799998      3389         6        2085036             8  ...       0.0         0         0     Benign
799999        53        17            370             1  ...       0.0         0         0     Benign

[795000 rows x 69 columns]
Malicious     47551
Benign       747449
Name: Label, dtype: int64
```

Figure 3: Dataset Preprocessing

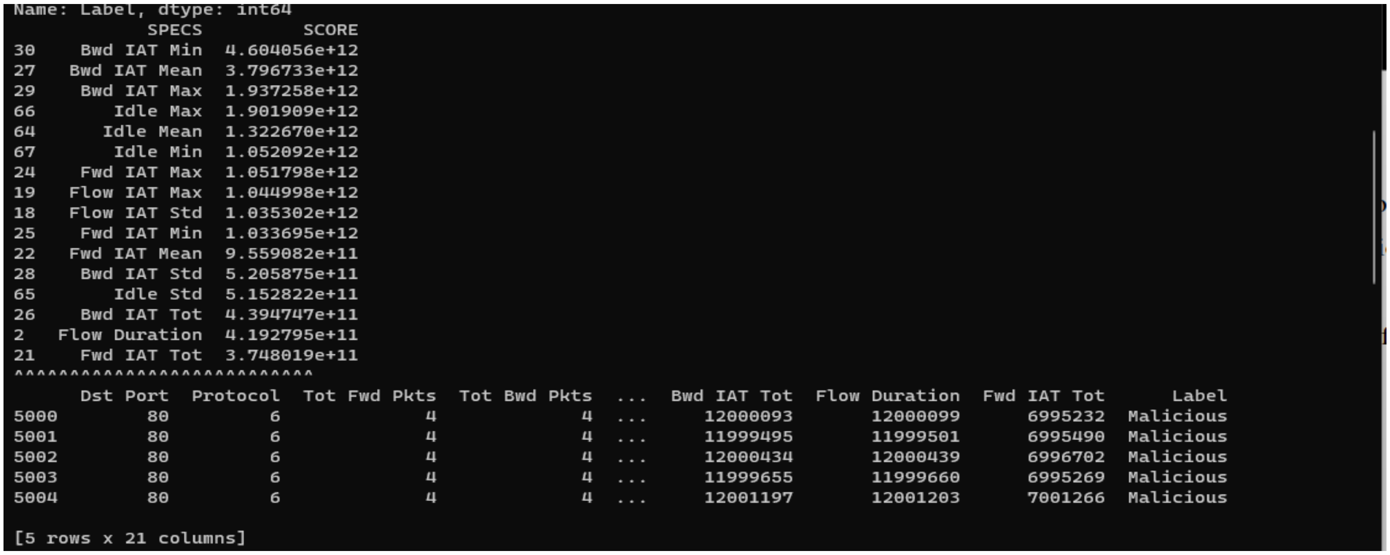
5.4.2 Feature Selection

The next step is Feature Selection which is important to train the model efficiently. This is done by removing unwanted features and only selecting those features which have high feature scores. This process increases the precision and reduces the training time of the model.

The method used for the same is SelectKBest chi2 method, in which, the 16 best features out of the total 66 are selected in descending order of score. These features along with the important features mentioned in a previous section are used to create a new dataset.

The methods Ensemble Extremely Randomized Trees Classifier and Ensemble Exhaustive feature selection was also tried out. The first method did not give uniform scores for the features and the second method did not work even after multiple tries of running the code.

After this, y_final is taken as model output which consists of only “Label” and x_final is taken as model input without the “Label” parameter but including the 20 features.



```
Name: Label, dtype: int64
              SPECS          SCORE
30     Bwd IAT Min  4.604056e+12
27    Bwd IAT Mean  3.796733e+12
29     Bwd IAT Max  1.937258e+12
66        Idle Max  1.901909e+12
64       Idle Mean  1.322670e+12
67        Idle Min  1.052092e+12
24     Fwd IAT Max  1.051798e+12
19    Flow IAT Max  1.044998e+12
18    Flow IAT Std  1.035302e+12
25     Fwd IAT Min  1.033695e+12
22    Fwd IAT Mean  9.559082e+11
28     Bwd IAT Std  5.205875e+11
65        Idle Std  5.152822e+11
26     Bwd IAT Tot  4.394747e+11
2    Flow Duration  4.192795e+11
21     Fwd IAT Tot  3.748019e+11
^^^^^^^^^^^^^^^^^^^^^^^^^^^
      Dst Port  Protocol  Tot Fwd Pkts  Tot Bwd Pkts  ...  Bwd IAT Tot  Flow Duration  Fwd IAT Tot      Label
5000        80         6             4             4  ...     12000093       12000099      6995232  Malicious
5001        80         6             4             4  ...     11999495       11999501      6995490  Malicious
5002        80         6             4             4  ...     12000434       12000439      6996702  Malicious
5003        80         6             4             4  ...     11999655       11999660      6995269  Malicious
5004        80         6             4             4  ...     12001197       12001203      7001266  Malicious

[5 rows x 21 columns]
```

Figure 2:Feature Selection using chi2

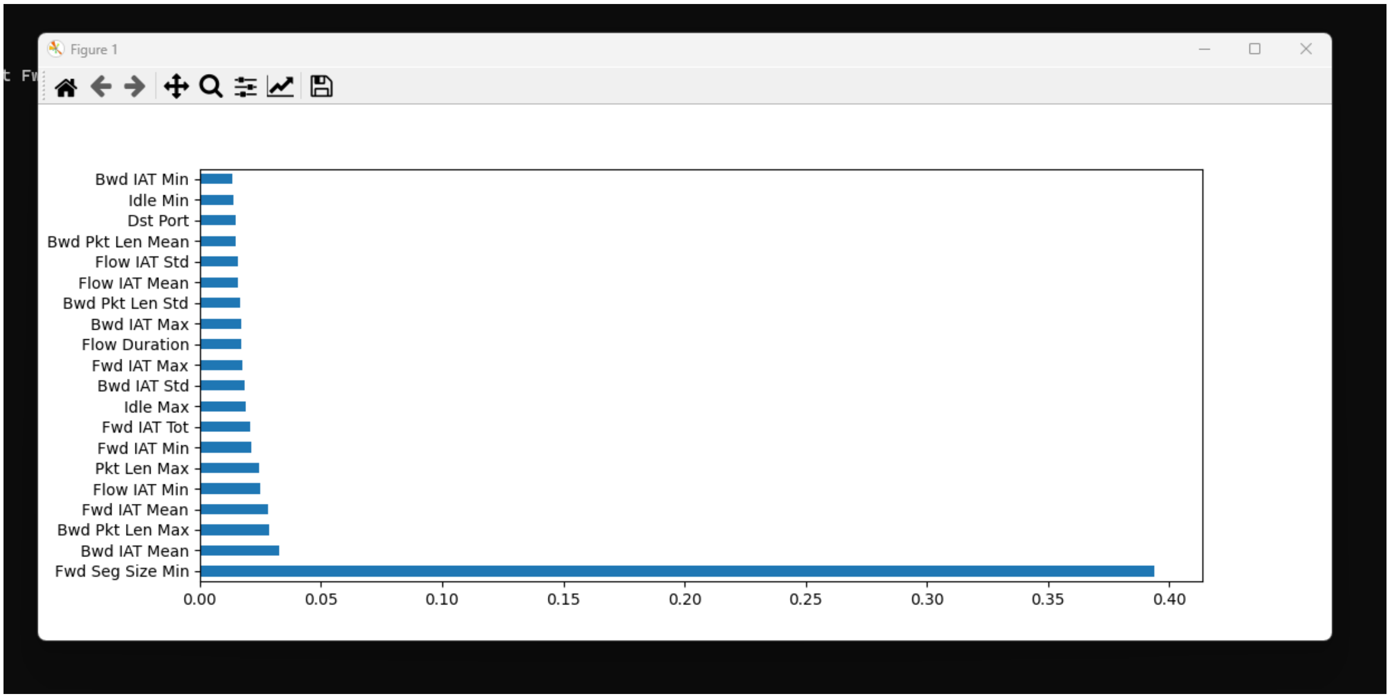
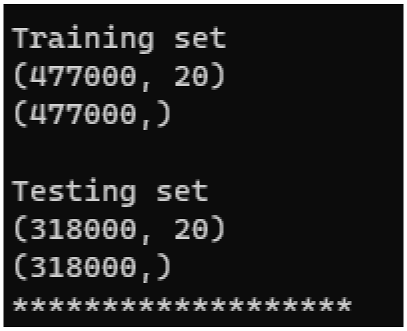


Figure 4: Scores of Extremely Randomized Trees Classifier

5.4.3 Train-Test Splitting 1

When a machine learning algorithm is used to predict data that was not used to train the model, its performance is estimated using the training test split method. This is a quick and easy process whose results allow us to compare the effectiveness of machine learning algorithms for specific predictive modeling problems. The train-test-split approach is appropriate when you need to quickly evaluate model performance, when models are expensive to train, or when you need very large datasets.

Here in my project, x_final and y_final is given as input and test_size is taken as 0.4 which means that 40% of the dataset is taken for testing and 60% is for training. x_train and y_train is the training dataset and x_test and y_test is the testing dataset. The testing dataset consists of Malicious and Benign entries.



```
Training set
(477000, 20)
(477000,)

Testing set
(318000, 20)
(318000,)
*******************
```

Figure 5:Train-test split 1

5.4.4 AdaBoost Classifier

The AdaBoost classifier is loaded and has attributes with values n_estimators=50 which is number of weak learners to train iteratively, the weak learner used here is DecisionTreeClassifier. It is the default weak learner.

x_train and y_train is passed as input to train the AdaBoost model. x_test is used to evaluate the model and the resulting output y_pred is the predicted value of the model, i.e., whether it is Malicious or Benign.

A confusion matrix is then plotted to evaluate the performance of the machine learning model. It takes the model predicted output which is y_pred and the original dataset's testing dataset, i.e., y_test as inputs.

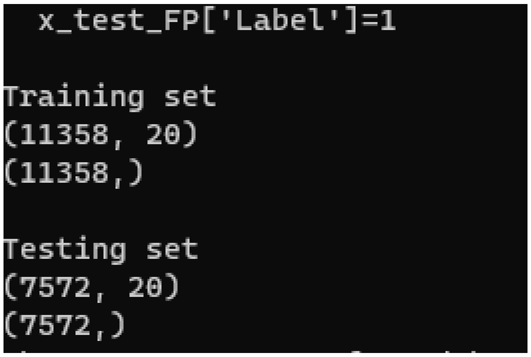
The all positives consists of True positives and False positives. If y_test predicts correctly, that entry is added to the list TP_indexes, else it is added to FP_indexes.

		Predicted	
		Negative	Positive
Actual	**Negative**	TN	FP
	Positive	FN	TP

Figure 6: Confusion Matrix

One of the main aim in this project is to reduce the false positive rates. Therefore, to do that, the true positive and false positive lists are merged and converted into a single dataset, which is a fine tuned dataset. y_get is the model output with only "Label" column in it and x_get is the model input excluding the "Label" parameter.

Similarly, train-test splitting is done again by giving the above two variables as input.



```
  x_test_FP['Label']=1

Training set
(11358, 20)
(11358,)

Testing set
(7572, 20)
(7572,)
```

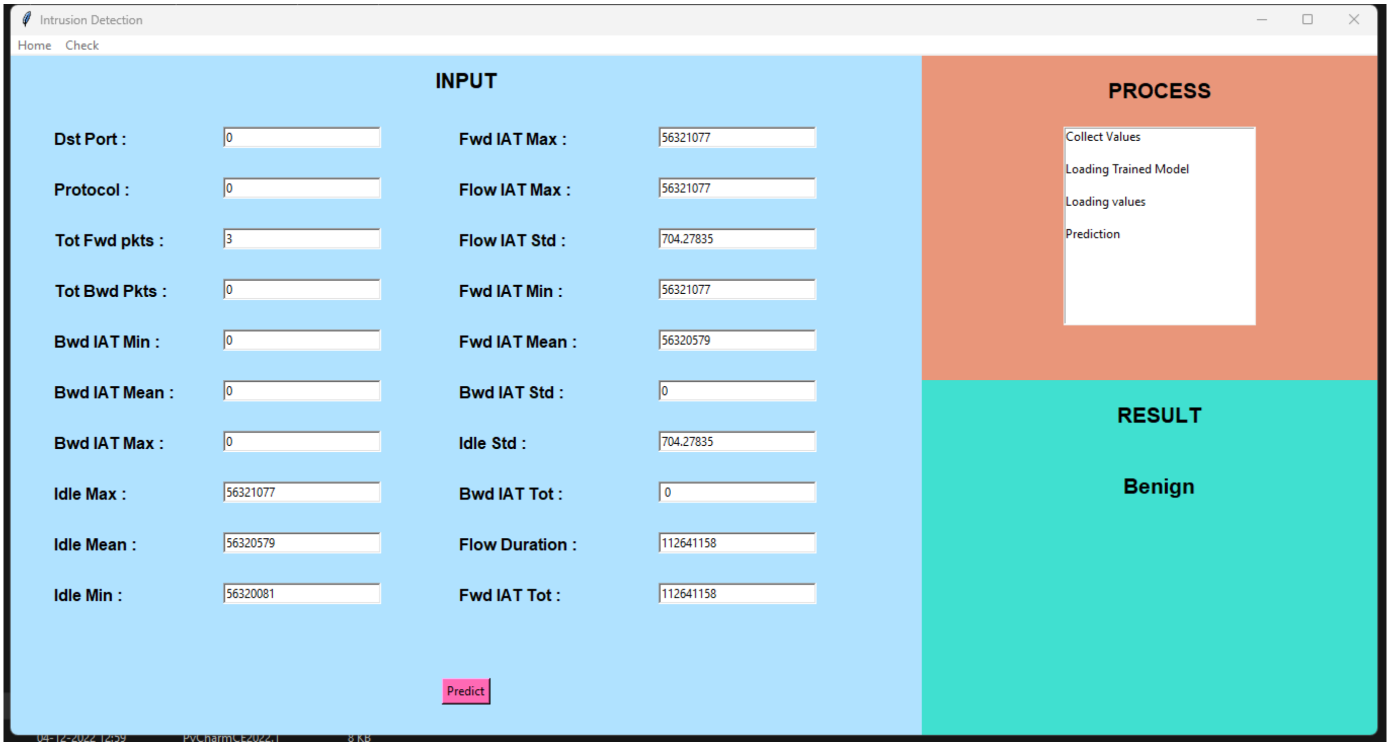
Figure 7:Train-test split 2

The AdaBoost classifier is again loaded and trained and then a second confusion matrix is plotted with the reduced false positive rate. The accuracy is also calculated using the fine tuned dataset's testing set (y_test1) and the AdaBoost model's second round of predicted output (y_pred1).

Finally, the trained model is saved for future prediction.

5.5 Prediction Stage

The saved model is loaded using joblib function. An empty list called Info and a list called Parameters is also created which contains the 20 features. The user enters the data of the corresponding features. Each entry is appended to the "info" list. The two lists are converted to a Dictionary and then converted to a Dataframe named my_data using Pandas library. The Dataframe is then loaded to the model to make the prediction, which will be two values, Malicious or Benign and the output variable is called my_pred. The result can be viewed in the command prompt and also in the Graphical User Interface (GUI). To add the data shown below and in the GUI, there is a text file called **Test.txt** with the values of the 20 features.



INPUT

Dst Port :	0	Fwd IAT Max :	56321077
Protocol :	0	Flow IAT Max :	56321077
Tot Fwd pkts :	3	Flow IAT Std :	704.27835
Tot Bwd Pkts :	0	Fwd IAT Min :	56321077
Bwd IAT Min :	0	Fwd IAT Mean :	56320579
Bwd IAT Mean :	0	Bwd IAT Std :	0
Bwd IAT Max :	0	Idle Std :	704.27835
Idle Max :	56321077	Bwd IAT Tot :	0
Idle Mean :	56320579	Flow Duration :	112641158
Idle Min :	56320081	Fwd IAT Tot :	112641158

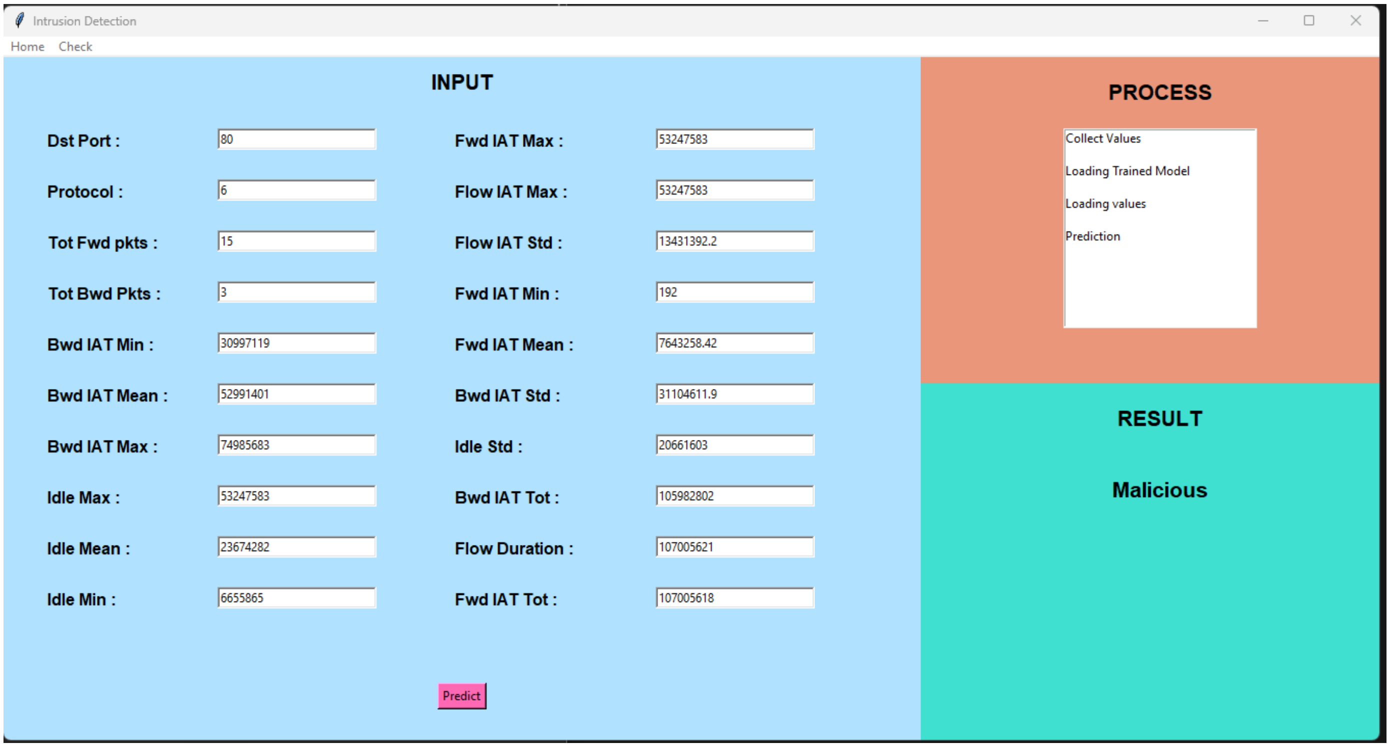
PROCESS

Collect Values
Loading Trained Model
Loading values
Prediction

RESULT

Benign

Figure 8: Entries which shows result as Benign



Dst Port :	80	Fwd IAT Max :	53247583
Protocol :	6	Flow IAT Max :	53247583
Tot Fwd pkts :	15	Flow IAT Std :	13431392.2
Tot Bwd Pkts :	3	Fwd IAT Min :	192
Bwd IAT Min :	30997119	Fwd IAT Mean :	7643258.42
Bwd IAT Mean :	52991401	Bwd IAT Std :	31104611.9
Bwd IAT Max :	74985683	Idle Std :	20661603
Idle Max :	53247583	Bwd IAT Tot :	105982802
Idle Mean :	23674282	Flow Duration :	107005621
Idle Min :	6655865	Fwd IAT Tot :	107005618

RESULT

Malicious

Figure 9: Entries that show result as Malicious

6 Evaluation

The AdaBoost model has been implemented and two confusion matrices have been obtained. The first one is with the original dataset and the second one is with the fine tuned dataset to reduce the false positives. To evaluate the performance of the model, the accuracy metric is also calculated. For calculating the same. The below terms need to be first understood:

- True Positive (TP): A TP stands for the precise designation of an attack packet as an attack.
- True Negative (TN): Correct classification of normal packets as normal is provided by the TN.
- False Negative (FN): FN serves as an example of an attack packet misclassified as normal. Increasing this value affects confidentiality and availability, two major security concerns.
- False Positive (FP): FP stands for an incorrect classification, which marks normal packets as attacks.
- Accuracy: Divide the total number of accurate predictions by the total number of predictions to determine the proposed model's accuracy.

Accuracy= (TP+TN) / (TP+TN+FP+FN)

6.1 Case Study 1: Confusion Matrix with original dataset

The first Confusion Matrix is plotted with the AdaBoost model's predicted y_pred and the testing dataset y_test. The values in the four blocks of the below confusion matrix add up to the number of rows in the first testing set of the first iteration of the train test split, which is, 318000 packets. Out of these packets, 298907 Benign packets were correctly classified as Benign (TN). The 166 benign packets were incorrectly classified as Malicious (FP). The 163 Malicious packets were incorrectly classified as Benign (FN). The remaining 18764 malicious packets were correctly classified as Malicious (TP).

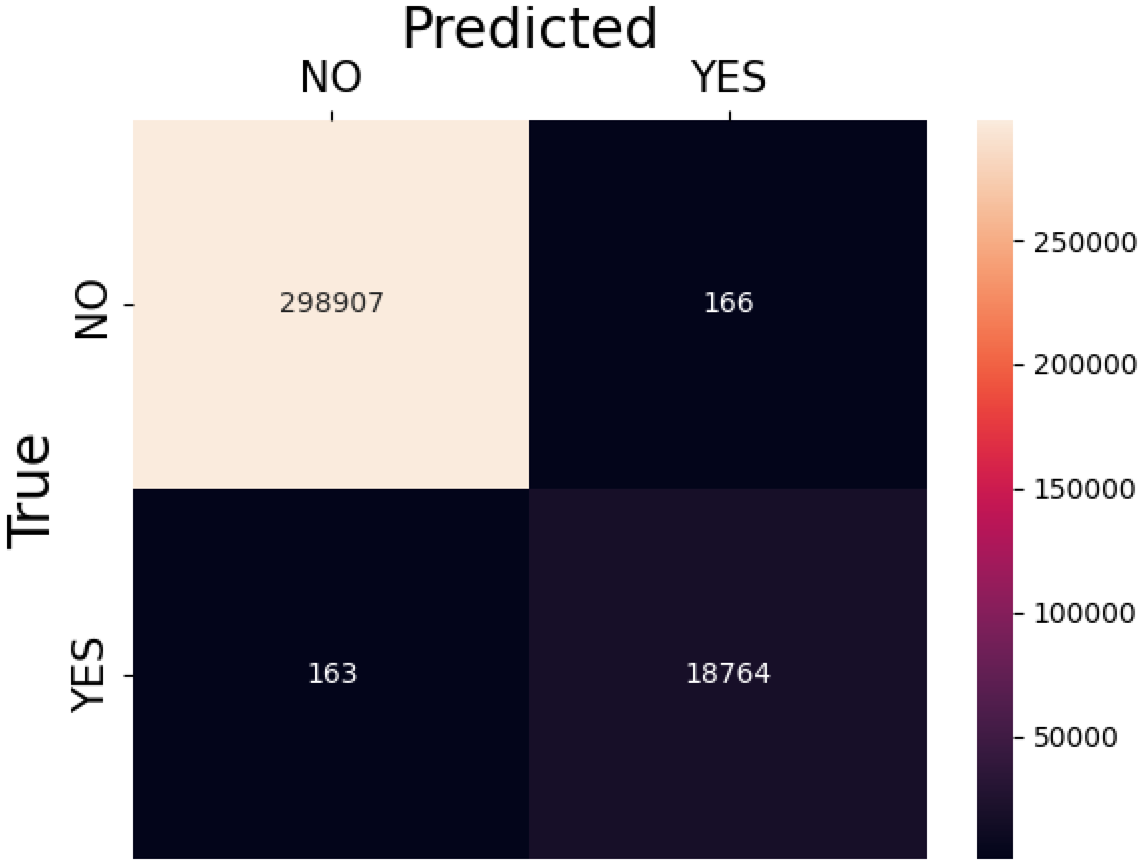


Figure 10: Confusion Matrix 1

6.2 Case Study 2: Confusion Matrix with fine-tuned dataset

The second confusion matrix is plotted with the AdaBoost model's y_pred1 and the fine-tuned dataset's testing set y_test1. The values in the below confusion matrix add up to the number of rows in the testing set of the second iteration of the train test split, which is, 7572 packets. Out of these packets, 7500 Benign packets were correctly classified as Benign. No benign packets were incorrectly classified as Malicious.

The 39 Malicious packets were incorrectly classified as Benign. The remaining 33 malicious packets were correctly classified as Malicious.

From the above results, it is found that the False Positive rate has decreased to 0 compared to the initial rate in the first matrix, thereby achieving the outcome which was proposed in this project.

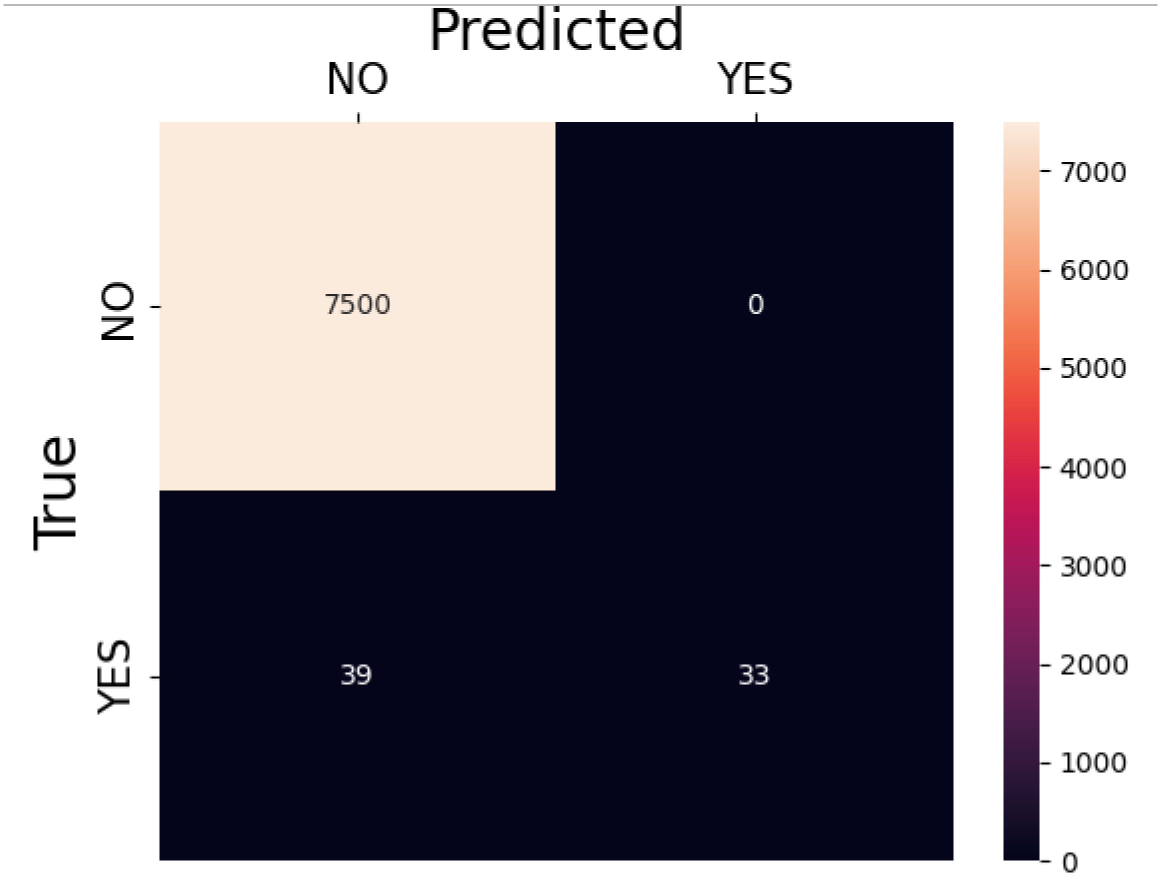
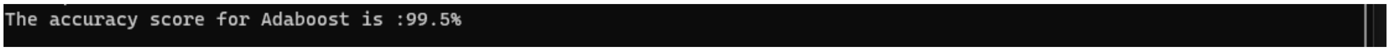


Figure 11: Confusion Matrix 2

The accuracy was also calculated for the model, which is 99.5%. The same is shown below:



```
The accuracy score for Adaboost is :99.5%
```

Figure 12: Accuracy of the model

The train_test_split function has a default attribute called “shuffle”. It randomly shuffles the amount of rows in the training and testing sets, and the number of Malicious and Benign entries in the Testing Dataset. As said before, the testing set contains Malicious and Benign entries. Due to the random shuffling, the accuracy and false positive rate therefore may vary by a small margin. The same is shown below:

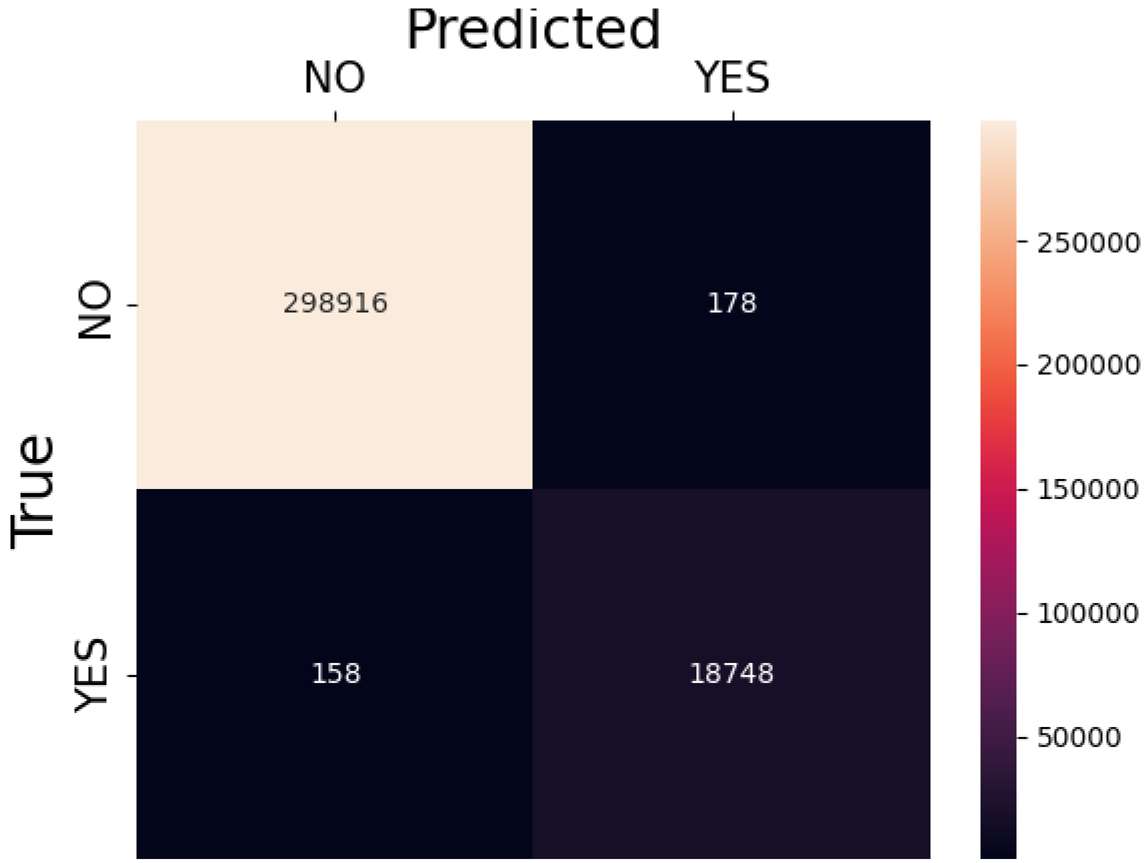


Figure 13: Confusion matrix in 2nd experiment after stage 1

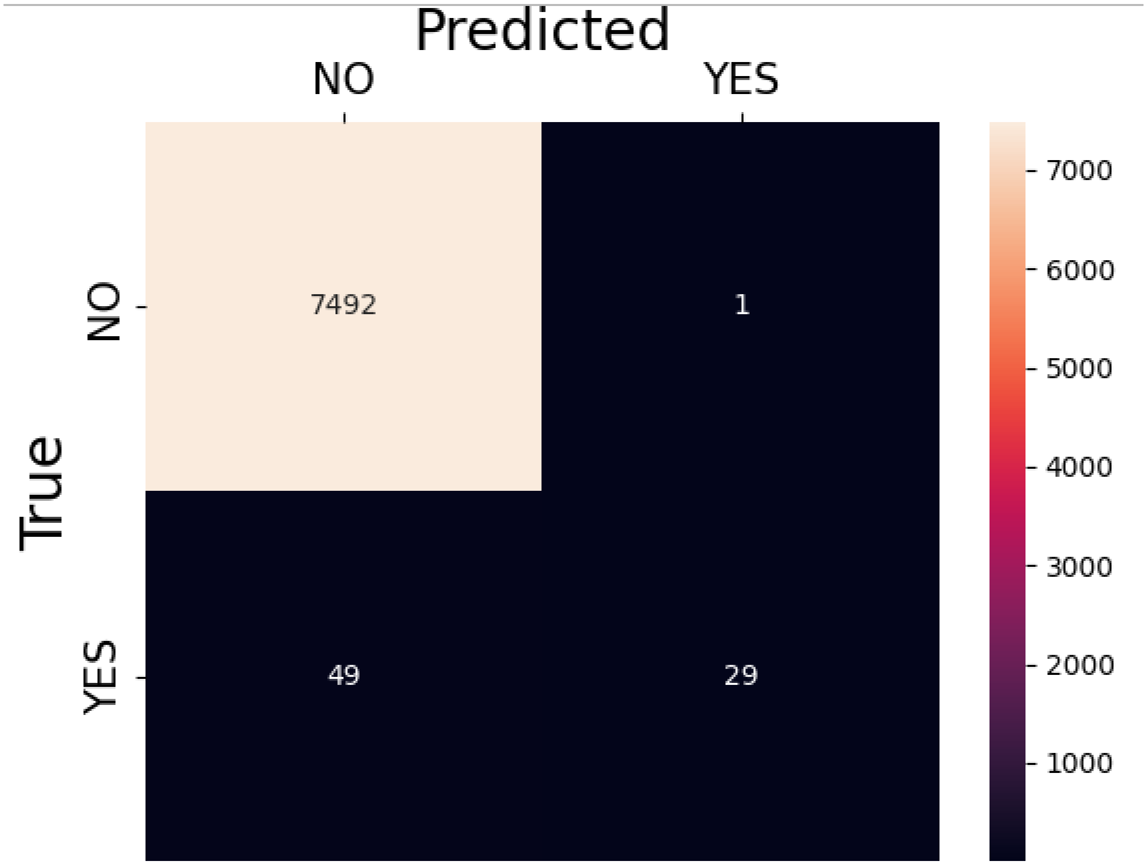
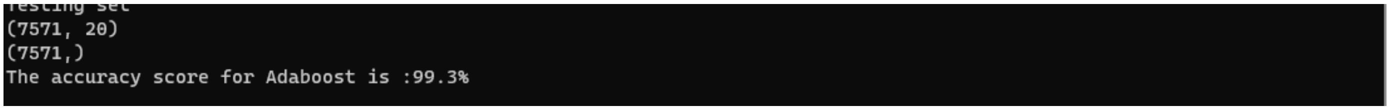


Figure 14: Confusion matrix in 2nd experiment after stage 2



```
Testing set
(7571, 20)
(7571,)
The accuracy score for Adaboost is :99.3%
```

Figure 15: Accuracy 2

The above figure shows the accuracy of the model after performing the experiment.

This section concludes that the accuracy of the model varies between 99.3% to 99.5%. This happens because of an attribute value called “shuffle = True”. It may go even lower but the same was not achieved even after multiple attempts of running the code.

7 Discussion

An IDS to detect Zero Day attacks and to reduce their false positive rate has been done using different methods. There were two important papers in which I researched on the topic and both of them exhibit less accuracy compared to my accuracy of 99.5%.

Research Paper	**Accuracy Obtained**	**My Proposed method's accuracy**
(Kumar and Sinha, 2021)	91.33%	**99.5%**
(Pitre et al., 2022)	95.1%	**99.5%**

The former paper did not use any feature selection technique. Hence, that may be the case of the accuracy difference compared to my approach.

The accuracy obtained for this project varies between 99.3% and 99.5% after many attempts in running the project. The said accuracy obtained is mainly dependent on the dataset used. If another dataset is used, the said accuracy may not be this good. Another factor is the usage of another feature selection algorithm other than chi2 which is used in this project. I have tested using two other feature selection algorithms as mentioned earlier in the "**Feature Selection**" section namely **Ensemble Extremely Randomized Trees Classifier** and **Ensemble Exhaustive feature selection**. The first one did not give uniform scores for the important features, which if implemented using the same, will have low accuracy compared to this as feature selection plays an important part in determining the accuracy of the model. The second algorithm did not run even after multiple attempts. The final factor is if another machine learning algorithm other than an ensemble machine learning algorithm is used, as AdaBoost algorithm used IDS(s) exhibit high accuracy and the algorithm combines many ineffective classifiers to create a strong classifier with a high level of accuracy. The papers mentioned in the literature review (Anitha and Arockiam, 2021; Moustafa et al., 2019; Rehman Javed et al., 2022) also support the high accuracy capability of AdaBoost algorithm.

Another aspect of this project is that this work focuses on Intrusion Detection, not Prevention. Which means that it is only able tell whether a packet is Malicious or Benign, and not the types of Malicious attacks detected for example like Malware, Spyware, Adware and so on. This is also a limitation. The dataset has only two classes for that Malicious and Benign, which make it a Binary Classification in Machine Learning. If the dataset consisted of types of attacks or Malware classes other than the two classifications for the packet, it will be a Multi Class classification. This will in turn make learning for the machine learning model with AdaBoost a bit harder and thereby reducing the accuracy.

8 Conclusion and Future Work

The paper focuses on detecting Zero day attacks and to reduce the false positive rates with the help of an ensemble machine learning approach called AdaBoost. With the help of chi2 feature selection technique, 16 features were extracted out of the 66 features from the original dataset. The proposed model is trained with

AdaBoost model and a confusion matrix was obtained. In order to reduce the false positives, the dataset was finetuned by merging the True Positive and False Positive indexes, which was then converted to a dataset for generating a second Confusion matrix. This matrix showed that the model had no false positive rate. The accuracy obtained using this technique is 99.5%, which is satisfactory compared to other technologies out there, and also makes AdaBoost a very efficient algorithm to detect zero day attacks.

For future work, since this model only does Intrusion Detection, using another dataset that has the type of intrusion such as whether it is a malware, spyware, adware and so on is possible. And also, usage of another ensemble machine learning approach such as XGBoost is also possible.

9 Link to full artefact

I am attaching a OneDrive link below to the artefact as my ZIP file had a size of 2.29GB, which cannot be uploaded on moodle. I have anyway submitted the code part of the project in a zip file on Moodle.

OneDrive Link: IDS_FULLCODE.zip

9 References

Akashdeep, Manzoor, I., Kumar, N., 2017. A feature reduced intrusion detection system using ANN classifier. Expert Syst. Appl. 88, 249–257. https://doi.org/10.1016/j.eswa.2017.07.005

Anitha, D., Arockiam, L., 2021. Ada-IDS: AdaBoost Intrusion Detection System for ICMPv6 based Attacks in Internet of Things. Int. J. Adv. Comput. Sci. Appl. 12, 499–506. https://doi.org/10.14569/IJACSA.2021.0121156

Farnaaz, N., Jabbar, M.A., 2016. Random Forest Modeling for Network Intrusion Detection System. Procedia Comput. Sci., Twelfth International Conference on Communication Networks, ICCN 2016, August 19– 21, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India 89, 213–217. https://doi.org/10.1016/j.procs.2016.06.047

Goeschel, K., 2016. Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis, in: SoutheastCon 2016. Presented at the SoutheastCon 2016, pp. 1–6. https://doi.org/10.1109/SECON.2016.7506774

Heba, F.E., Darwish, A., Hassanien, A.E., Abraham, A., 2010. Principle components analysis and Support Vector Machine based Intrusion Detection System, in: 2010 10th International Conference on Intelligent Systems Design and Applications. Presented at the 2010 10th International Conference on Intelligent Systems Design and Applications, pp. 363–367. https://doi.org/10.1109/ISDA.2010.5687239

Holm, H., 2014. Signature Based Intrusion Detection for Zero-Day Attacks: (Not) A Closed Chapter? 2014 47th Hawaii Int. Conf. Syst. Sci. Syst. Sci. HICSS 2014 47th Hawaii Int. Conf. Syst. Sci. HICSS 2013 46th Hawaii Int. Conf. On 4895–4904. https://doi.org/10.1109/HICSS.2014.600

IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018) [WWW Document], n.d. URL https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv (accessed 11.26.22).

Kim, Jiyeon, Kim, Jiwon, Kim, H., Shim, M., Choi, E., 2020. CNN-Based Network Intrusion Detection against Denial-of-Service Attacks. Electronics 9, 916. https://doi.org/10.3390/electronics9060916

Kumar, V., Sinha, D., 2021. A robust intelligent zero-day cyber-attack detection technique. Complex Intell. Syst. 7, 2211–2234. https://doi.org/10.1007/s40747-021-00396-9

Kwon, D., Natarajan, K., Suh, S., Kim, H., Kim, J., 2018. An Empirical Study on Network Anomaly Detection Using Convolutional Neural Networks. pp. 1595–1598. https://doi.org/10.1109/ICDCS.2018.00178

Lin, C., Li, A., Jiang, R., 2021. Automatic Feature Selection and Ensemble Classifier for Intrusion Detection. J. Phys. Conf. Ser. 1856, 012067. https://doi.org/10.1088/1742-6596/1856/1/012067

Moustafa, N., Turnbull, B., Choo, K.-K.R., 2019. An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. IEEE Internet Things J. 6, 4815–4830. https://doi.org/10.1109/JIOT.2018.2871719

Pallaprolu, S.C., Sankineni, R., Thevar, M., Karabatis, G., Wang, J., 2017. Zero-Day Attack Identification in Streaming Data Using Semantics and Spark, in: 2017 IEEE International Congress on Big Data (BigData Congress). Presented at the 2017 IEEE International Congress on Big Data (BigData Congress), pp. 121–128. https://doi.org/10.1109/BigDataCongress.2017.25

Pallippattu Mathai, L., 2021. Malware Detection on Android using Adaboost Algorithm (masters). Dublin, National College of Ireland.

Pitre, P., Gandhi, A., Konde, V., Adhao, R., Pachghare, V., 2022. An Intrusion Detection System for Zero-Day Attacks to Reduce False Positive Rates, in: 2022 International Conference for Advancement in Technology (ICONAT). Presented at the 2022 International Conference for Advancement in Technology (ICONAT), pp. 1–6. https://doi.org/10.1109/ICONAT53423.2022.9726105

Rana, Md.S., Sung, A.H., 2020. Evaluation of Advanced Ensemble Learning Techniques for Android Malware Detection. Vietnam J. Comput. Sci. 07, 145–159. https://doi.org/10.1142/S2196888820500086

Rehman Javed, A., Jalil, Z., Atif Moqurrab, S., Abbas, S., Liu, X., 2022. Ensemble Adaboost classifier for accurate and fast detection of botnet attacks in connected vehicles. Trans. Emerg. Telecommun. Technol. 33, e4088. https://doi.org/10.1002/ett.4088

Research Scholar, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamilnadu, India., Latha, P.H., Mohanasundaram, R., Associate Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamilnadu, India., 2019. A New Hybrid Strategy for Malware Detection Classification with Multiple Feature Selection Methods and Ensemble Learning Methods. Int. J. Eng. Adv. Technol. 9, 4013–4018. https://doi.org/10.35940/ijeat.B4666.129219

Shahraki, A., Abbasi, M., Haugen, Ø., 2020. Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost. Eng. Appl. Artif. Intell. 94, 103770. https://doi.org/10.1016/j.engappai.2020.103770

Soltani, M., Ousat, B., Jafari Siavoshani, M., Jahangir, A., 2021. An Adaptable Deep Learning-Based Intrusion Detection System to Zero-Day Attacks.

Tang, D., Tang, L., Dai, R., Chen, J., Li, X., Rodrigues, J.J.P.C., 2020. MF-Adaboost: LDoS attack detection based on multi-features and improved Adaboost. Future Gener. Comput. Syst. 106, 347–359. https://doi.org/10.1016/j.future.2019.12.034

Venkatarathinam, R., Raj, V.C., George, G., 2018. EEOFSA: An Embedded Ensemble Optimal Feature Selection Algorithm for Building Efficient Intrusion Detection System [WWW Document]. URL https://www.semanticscholar.org/paper/EEOFSA%3A-An-Embedded-Ensemble-Optimal-Feature-for-Venkatarathinam-Raj/ea7dcea4a2ccbadc5c414aa85a4e80133d804ad3 (accessed 7.29.22).

Zhou, Q., Pezaros, D., 2019. Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection -- An Analysis on CIC-AWS-2018 dataset.