Project Report

Topic: Network traffic analysis

INTRUSTION DETECTION USING CICIDS2017

Team members

Ta Quang Duy – 20214884

Dao Ha Xuan Mai – 2021

Vu Tung Linh – 2021

Course: Applied Statistics and Experimental Design 141179-IT2022E
Instructor: Pr. Nguyen Linh Giang

ABSTRACT

Network traffic plays a crucial role in shaping our daily lives, whether we are aware of it or not. As the backbone of modern communication, network traffic enables seamless internet browsing, instant messaging, and social media interactions, keeping us connected to friends, family, and the world at large. It also underpins critical services like online banking, e-commerce, and telemedicine, enhancing convenience and accessibility. However, network congestion and performance issues can lead to frustration and hinder productivity, highlighting the need for continuous advancements in network infrastructure to meet the ever-growing demands of our interconnected society.

1. INTRODUCTION

In this project, the aim is to detect network anomalies using machine learning methods. Network anomalies can indicate a wide range of issues, including cyber attacks, network congestion, and hardware failures. Traditional methods for detecting these anomalies rely on pre-defined signatures and can be ineffective against new or unknown threats. This project addresses this challenge by using advanced machine learning algorithms to analyze network traffic data and detect anomalies.

The dataset used in this project is the CICIDS2017 dataset, which contains benign and the most up-to-date common attacks, resembling true real-world data. It includes the results of network traffic analysis using CICFlowMeter with labeled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols, and attack.

The problem being addressed in this project is the challenge of detecting network anomalies in an effective and adaptable manner. Traditional methods for detecting these anomalies can be ineffective against new or unknown threats, making it difficult to ensure the smooth operation of computer networks. By using advanced machine learning algorithms to analyze network traffic data, this project aims to provide a more effective solution for network traffic diagnosis.

2. DATA PREPARATION

2.1 CICIDS2017 Dataset

CICIDS 2017 (Intrusion Detection Evaluation Dataset) created by the Canadian Institute for Cybersecurity at the University of New Brunswick. This dataset consists of a 5-day (3rd July- 7th July 2017) data stream on a network created by computers using up-to-date operating systems such as Windows Vista / 7 / 8.1 / 10, Mac, Ubuntu 12/16 and Kali. Details of the dataset can be seen from Table below:

Flow Recording Day (Working Hours)	pcap File size	Duration	CSV File Size	Attack Name	Flow Count
Monday	10 GB	All Day	257 MB	No Attack	529918
Tuesday	10 GB	All Day	166 MB	FTP-Patator, SSH-Patator	445909
Wednesday	12 GB	All Day	272 MB	DoS Hulk, DoS GoldenEye, DoS slowloris, DoS Slowhttptest, Heartbleed	692703
Thursday	7.7GB	Morning	87.7 MB	Web Attacks (Brute Force, XSS, Sql Injection)	170366
		Afternoon	103 MB	Infiltration	288602
Friday	8.2GB	Morning	71.8 MB	Bot	192033
		Afternoon	92.7 MB	DDoS	225745
		Afternoon	97.1 MB	PortScan	286467

Table 1. Details of CICIDS2017 Dataset

The CICIDS 2017 dataset has several advantages :

- Real-World Data: The dataset was collected from a testbed consisting of actual computers, making it a realistic representation of network traffic.
- Operating System Diversity: The data streams were obtained from computers running up-to-date operating systems like Mac, Windows, and Linux, both as attacker and victim systems.
- Labelled Data: The dataset is labeled, making it suitable for applying machine learning methods. It underwent feature extraction, resulting in 85 features that can be used for analysis.
- Availability of Raw and Processed Data: Both the raw data (pcap files) containing captured network packets and processed data (CSV files) are available for use.
- Wide Variety of Attacks: The dataset incorporates a broad range of attacks based on the 2016 McAfee security report, ensuring it covers diverse and up-to-date attack scenarios.
- Protocol Abundance: CICIDS 2017 includes various protocols, such as FTP, HTTP, SSH, and email, and notably, the HTTPS protocol, which enhances its applicability.

Despite these advantages, the dataset also has some drawbacks:

Large Data Files: The raw and processed data files are substantial in size, with the raw files being 47.9 GB and the processed files 1147.3 MB.

- Lack of Separate Training and Testing Files: Unlike some other datasets, CICIDS 2017 doesn't have predefined training and testing files. Users need to create these sections themselves.
- Potential Minor Mistakes: As a relatively new dataset, CICIDS 2017 might contain minor errors or deficiencies that require further investigation and data cleansing.

Considering the comparison, the CICIDS2017-processed data (CSV files) is selected as the dataset for the implementation phase due to the following reasons:

- Up-to-Date and Wide Protocol Coverage: The dataset is current and includes a diverse range of protocols and attacks, making it relevant for contemporary network security research.

- Potential Contribution to Literature: As there are relatively few works done with this dataset, working with it could lead to a significant contribution to the field of intrusion detection and cybersecurity research.

In this section, a detailed examination of the types of attacks present in the dataset is conducted. The dataset comprises data instances that are labeled with 15 different tags. Among these tags, one is labeled as "Benign," representing normal network activities, while the remaining 14 tags correspond to different types of attacks.

The "Benign" records are created using Mail services, SSH, FTP, HTTP, and HTTPS protocols to simulate non-harmful and normal data streams on the network, resembling real user data [6]. These benign records serve as a baseline for distinguishing normal network behavior from malicious activities.

The names and corresponding numbers of these 15 labels can be found in Figure 1 of the dataset documentation, providing a comprehensive overview of the attack types represented in the dataset.

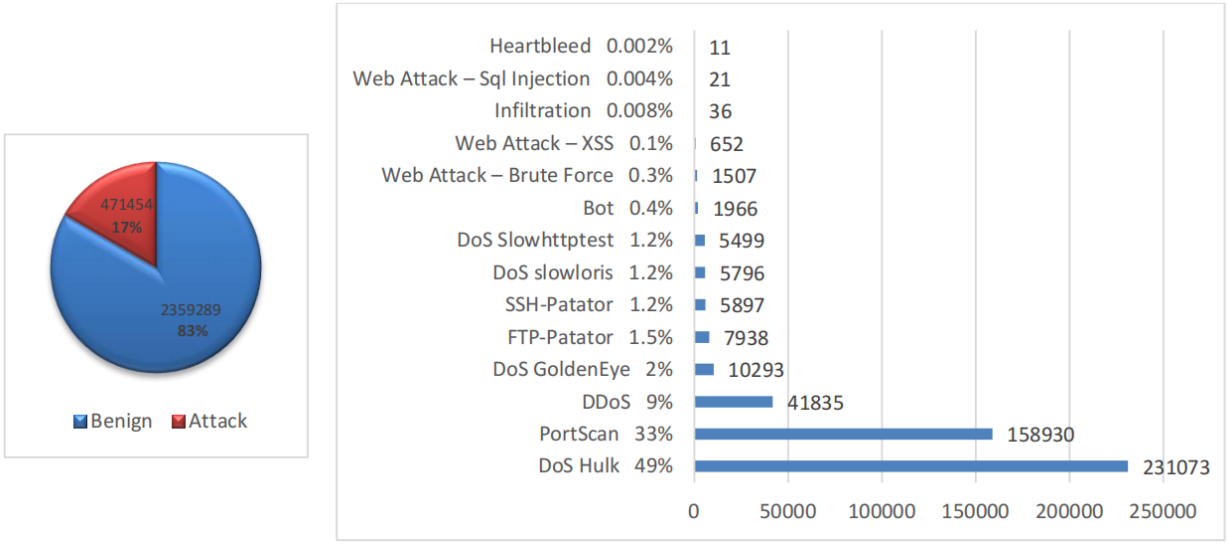


Figure 1. The distribution of data flow and attack types in the dataset.

2.2 Preprocessing data

2.2.1 Data cleansing

It may be necessary to make some changes to the dataset before using it in practice, making it more efficient. For this purpose, in this section, some defects of the CICIDS2017 dataset are corrected, and some data are edited.

The dataset file contains 3119345 stream records. The distribution of these stream records can be seen from Table 2. When these records are examined, it can be seen that the 288602 record is incorrect / incomplete3. The first step in the pre-processing process will be to delete these unnecessary records.

Label Name	Number
Benign	2359289
Faulty	288602
DoS Hulk	231073
PortScan	158930
DDoS	41835
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack – Brute Force	1507
Web Attack – XSS	652
Infiltration	36
Web Attack – SQL Injection	21
Heartbleed	11

Table 2. Distribution of stream records in the CICIDS2017 dataset.

In the dataset, there is an error in the columns that constitute the features. The dataset file contains 86 columns that define various flow properties such as Flow ID, Source IP, Source Port, etc. However, the feature "Fwd Header Length," which represents the forward direction data flow for total bytes used, is mistakenly duplicated and appears in both the 41st and 62nd columns. To rectify this error, the repeating column (column 62) should be removed from the dataset.

Additionally, there is a need to convert properties that include categorical and string values, such as Flow ID, Source IP, Destination IP, Timestamp, and External IP, into numerical data to be suitable for use in machine learning algorithms. This conversion can be achieved by employing the LabelEncoder() function from the Sklearn classes. By using LabelEncoder(), various string values will be assigned integer values ranging from 0 to n-1, making them more suitable for processing in machine learning operations. This step ensures that the data is in a format that can be effectively used in machine learning models.

It appears that the "Label" tag, which represents the categorical feature indicating different attack types, has not been modified. The reason for retaining the original categories is to allow for classification of attack types in various forms and to enable the exploration of different approaches during data processing.

However, there are some minor structural changes that need to be made to the dataset, as follows:

- Handling Web Attack Subtypes: In the "Label" feature, the character "–" (Unicode Decimal Code &#8211) is used to identify web attack subtypes like "Web Attack - Brute Force," "Web Attack - XSS," and "Web Attack - SQL Injection." To avoid compatibility issues with the Pandas library, this character should be replaced with "-" (Unicode Decimal Code &#45) since utf-8, the default codec of Pandas, does not recognize it. This change ensures that the Pandas library properly processes the data without any failures.
- Handling "Infinity" and "NaN" values: The "Flow Bytes/s" and "Flow Packets/s" features include values such as "Infinity" and "NaN" in addition to numerical values. To make these features suitable for machine learning algorithms, the "Infinity" values can be modified to -1, and the "NaN" values can be modified to 0.

2.2.1 Creation of Training and Testing data

In the machine learning process, training data is essential for the algorithm to learn and acquire skills. Additionally, test data is required to evaluate the algorithm's performance and assess how well it generalizes to new, unseen data. The algorithm learns from the training data and applies its learned knowledge to the test data to measure its performance .

However, in the case of the CICIDS2017 dataset used, it does not come pre-divided into dedicated training and test data sets. Therefore, it becomes necessary to split the dataset into separate training and test data parts. For this purpose, the Sklearn command "train_test_split"[59] is utilized during the application phase. This command divides the data into two parts, with the user specifying the sizes of the partitions. The common preference is to allocate 80% of the data for training and 20% for testing [40], and this ratio is also adopted in this application.

The "train_test_split" command performs random selection when creating these data groups, which is known as cross-validation. To ensure robustness in the results obtained during the application, the process of creating training and test data is repeated 10 times consecutively. The final results are then calculated as the arithmetic mean of these repeated operations.

By following this approach, the machine learning algorithm can be trained on a representative portion of the data and evaluated on unseen data, providing a better assessment of its generalization and performance capabilities.

2.3 Feature selection

In this section, the features in the dataset are evaluated to determine which features are important to define which attack.

2.3.1 Feature Selection According to Attack Types

To do this calculation, a special file is created for every kind of attack by isolating the attack from other attacks. This file contains the entire stream identified as the attack and the data stream identified as randomly selected "Benign" (30% Attack, 70% Benign).

The Random Forest Regressor class from Sklearn is utilized to calculate the importance weights of features. This algorithm constructs a decision forest, where each feature is assigned an importance weight, indicating how useful they are in constructing decision trees. Once the process is complete, these importance weights

of features are compared and sorted . The sum of the importance weights of all properties in the decision forest gives the total importance weight of the tree. By comparing the importance score of any feature to the score of the whole tree, one can determine the importance of that feature in the decision tree.

However, out of the 85 properties, 8 features (Flow ID, Source IP, Source Port, Destination IP, Destination Port, Protocol, Timestamp, External IP) should be excluded when calculating the importance weights. These features are omitted because attackers may avoid using well-known ports to evade detection or bypass operating system constraints, or they might employ generated or fake IP addresses. Additionally, many ports are used dynamically, and multiple applications can be transmitted over the same port, making it misleading to use the port number as a decisive feature [61].

To ensure more effective attribute selection for assessing attack importance, it is better to eliminate misleading features such as IP address, port number, and timestamp. Instead, more generic and invariant attributes should be used to define the attack. The shape of the data will provide more valuable information about whether an attack is occurring or not.

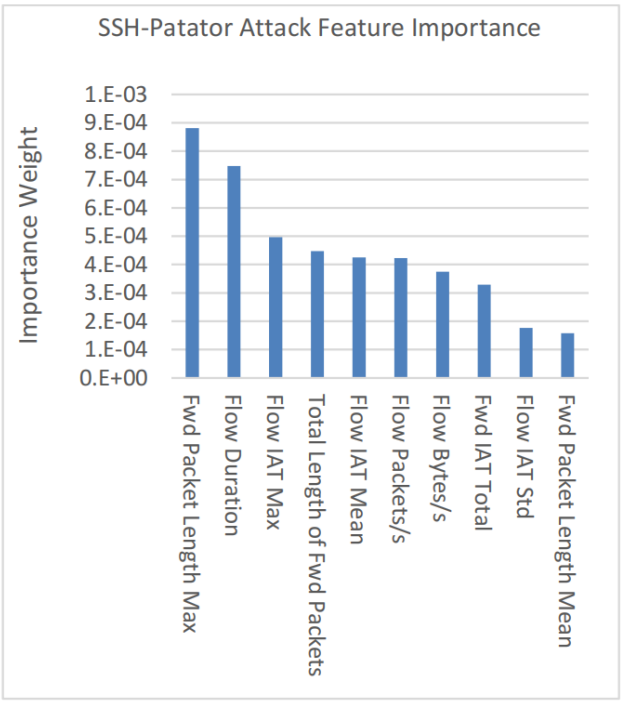
Table 3 displays the distribution of features and the four attributes with the most significant values for each attack, helping to identify the most crucial features when assessing the importance of attacks.

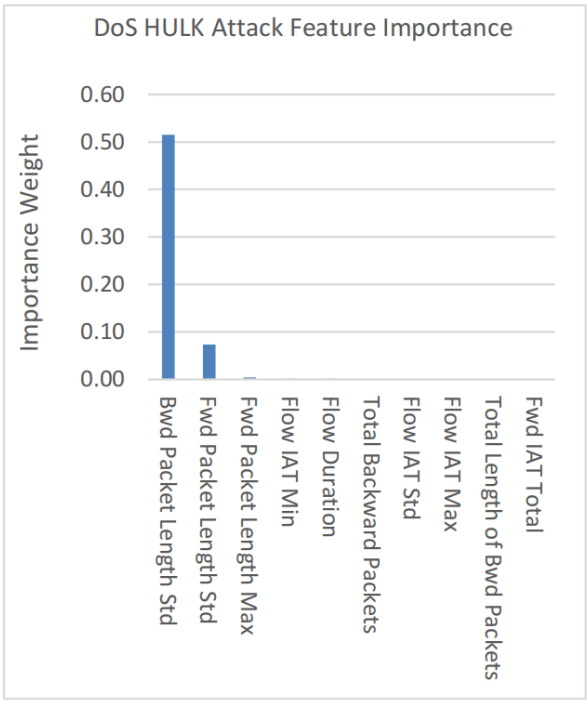
Attack / Feature Name	**Importance Weight**		**Attack / Feature Name**	**Importance Weight**
Bot			**FTP-Patator**	
Bwd Packet Length Mean	0.304823		Fwd Packet Length Max	0.063671
Flow IAT Max	0.034495		Fwd Packet Length Std	0.022751
Flow IAT Std	0.019464		Fwd Packet Length Mean	0.002179
Flow Duration	0.010129		Total Length of Bwd Packets	0.000746
DDoS			**Heartbleed**	
Bwd Packet Length Std	0.468089		Bwd Packet Length Mean	0.064
Total Backward Packets	0.094926		Total Length of Bwd Packets	0.056
Fwd IAT Total	0.012066		Flow IAT Min	0.056
Total Length of Fwd Packets	0.006438		Bwd Packet Length Std	0.044
DoS GoldenEye			**Infiltration**	
Flow IAT Max	0.442727		Total Length of Fwd Packets	0.05238
Bwd Packet Length Std	0.091185		Flow IAT Max	0.036096
Flow IAT Min	0.053795		Flow Duration	0.016453
Total Backward Packets	0.041583		Flow IAT Min	0.015448
DoS Hulk			**PortScan**	
Bwd Packet Length Std	0.514306		Flow Bytes/s	0.313402
Fwd Packet Length Std	0.069838		Total Length of Fwd Packets	0.304917
Fwd Packet Length Max	0.008542		Flow Duration	0.000485
Flow IAT Min	0.001716		Fwd Packet Length Max	0.00013

DoS Slowhttptest			**SSH-Patator**	
Flow IAT Mean	0.64206		Flow Bytes/s	0.000846
Fwd Packet Length Min	0.075942		Total Length of Fwd Packets	0.000814
Fwd Packet Length Std	0.022194		Fwd Packet Length Max	0.000749
Bwd Packet Length Mean	0.020857		Flow IAT Mean	0.000734
DoS slowloris			**Web Attack**	
Flow IAT Mean	0.465561		Total Length of Fwd Packets	0.014697
Bwd Packet Length Mean	0.075633		Bwd Packet Length Std	0.00536
Total Length of Bwd Packets	0.049808		Flow Bytes/s	0.00257
Total Fwd Packets	0.01868		Bwd Packet Length Max	0.001922

Table 3. The distribution of features and four attributes with the most significant value for each attack.

When the distribution of properties is examined, it appears that there are one or two features that stand out for almost all attack types. On the other hand, the characteristics of the Heartbleed and SSH-Patator attacks are quite different. For these attacks, there are a few features that have values that are very close to one another, not one or two dominant features.





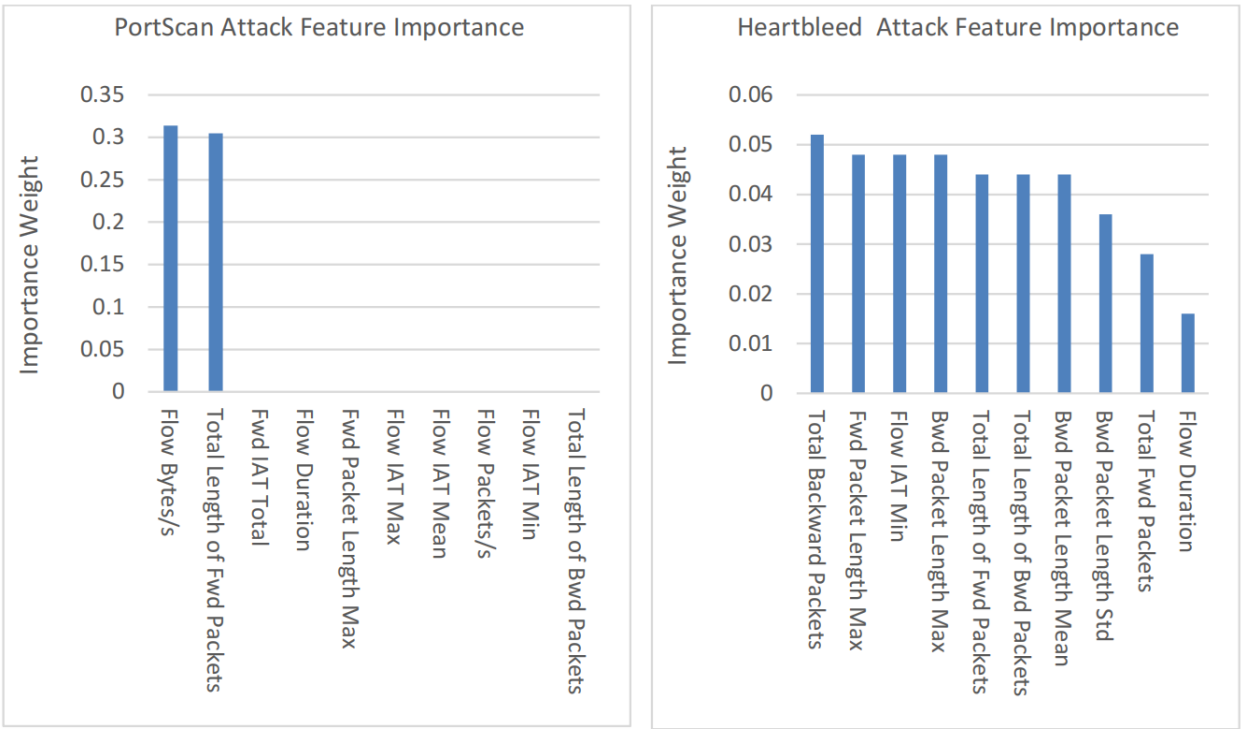


Figure 2. Graphs of feature importance weights of SSH-Patator, Heartbleed, DoS HULK, and PortScan attacks

For example, in the PortScan attack, two features ("Flow Bytes/s "- Number of flow bytes per second, "Total Length of Fwd Packets" - Total size of the packet in the forward direction) stand out. When the PortScan attack is examined, the attacker's behaviour is usually to send as many packets as possible with no payloads (only 20 Bytes transport layer header and 20 Bytes Internet layer header, 40 bytes in total) or with very few payloads . Thus, the attacker both accelerates the process and makes the attack more effective and does not consume his bandwidth unnecessarily. In this context, it is expected that the "Total Length of Fwd Packets" feature should be very low when it compared to benign flows. The data in Figure 3, are such as to support this prediction.

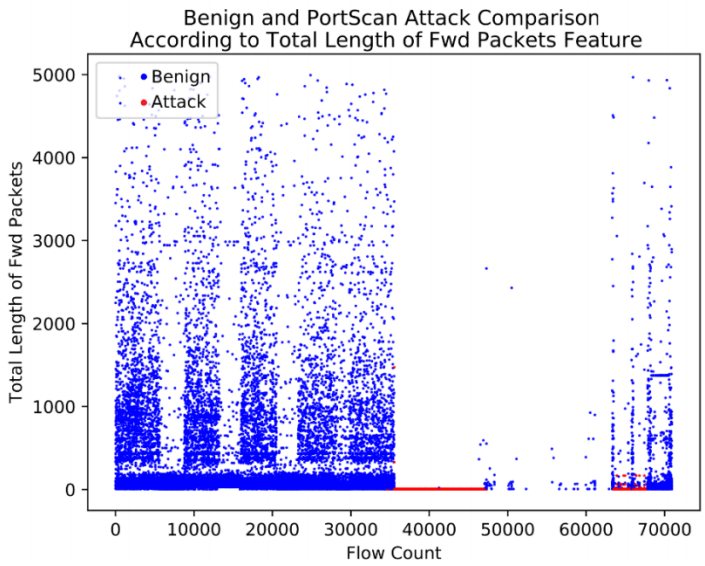


Figure 3. Benign and PortScan Attack Comparison According to Total Length of Fwd Packets Feature

On the other hand, the S SH-Patator attack does not reveal any obvious feature. The importance values are very close to each other. The most important reason for this is that the SSH-Patator attack does not consist of a single step but a complex structure with three steps (Scanning Phase, Brute-Force Phase and Die-off Phase)[31]. SSH-Patator attack contains both PortScan (Scanning/Probe) and Brute-Force attacks. In this context, it is possible to see traces of both attacks in Figure 4.

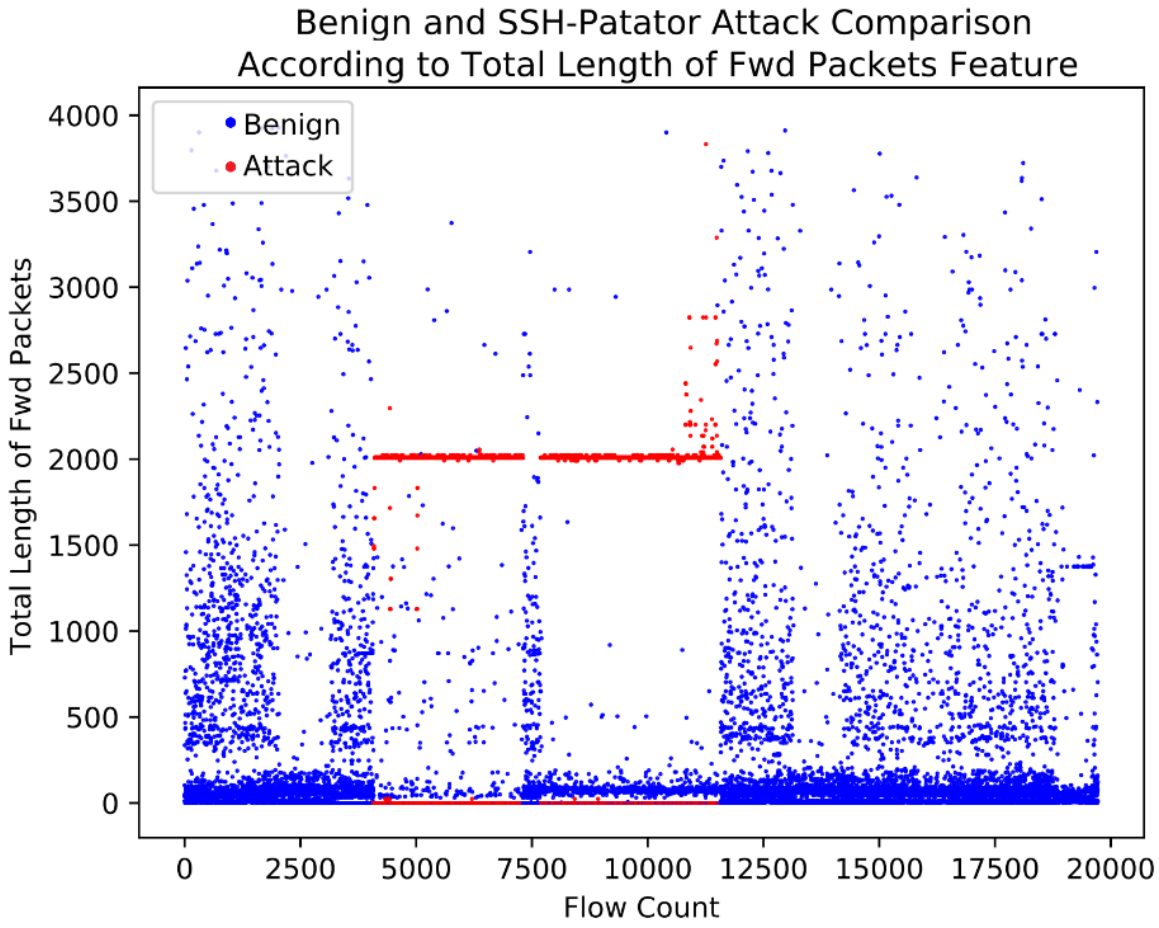


Figure 4. Benign and SSH-Patator Attack Comparison According to Total Length of Fwd Packets Feature.

2.3.2 Feature Selection According to Attack or Benign

To do this calculation, a special file is created for every kind of attack by isolating the attack from other attacks. This file contains the entire stream identified as the attack and the data stream identified as randomly selected "Benign" (30% Attack, 70% Benign).

Another approach in feature selection is to apply the Random Forest Regressor operation to the whole dataset by collecting all attack types under a single label; “attack”. So, the data in this file contains only the attack and benign tags. As a result of this operation, the feature list obtained is shown in Table 4 and graphics of feature in Figure 5.

Feature Name	Importance Weight		Feature Name	Importance Weight
Bwd Packet Length Std	0.246627		Flow IAT Mean	0.003266
Flow Bytes/s	0.178777		Total Length of Bwd Packets	0.001305
Total Length of Fwd Packets	0.102417		Fwd Packet Length Min	0.000670
Fwd Packet Length Std	0.063889		Bwd Packet Length Mean	0.000582
Flow IAT Std	0.009898		Flow Packets/s	0.000541
Flow IAT Min	0.006946		Fwd Packet Length Mean	0.000526
Fwd IAT Total	0.005121		Total Backward Packets	0.000169
Flow Duration	0.004150		Total Fwd Packets	0.000138
Bwd Packet Length Max	0.004007		Fwd Packet Length Max	0.000125
Flow IAT Max	0.003579		Bwd Packet Length Min	0.000084

Table 4. According Attack and Benign Labels Feature Importance Weight List

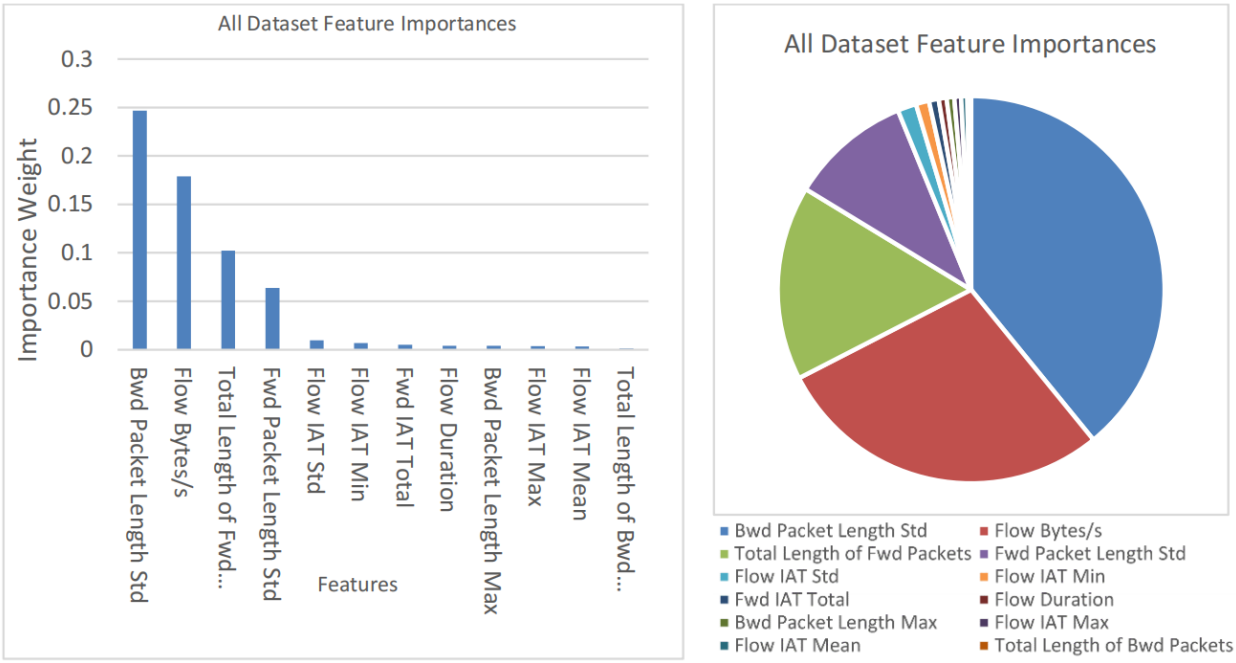


Figure 5. Graphs of Feature Importance Weights According to Attack and Benign Labels

3. METHODS EXPLAINATION

Machine learning is both a science and an art that empowers computers to learn from the provided data . During the machine learning process, computers are trained on a set of data (training set) and then tested on different data (test set) to demonstrate their performance. This allows problems to be solved with

minimal human intervention. Machine learning is particularly useful in situations where classical methods are inefficient. Its areas of application include:

- Handling large and complex data sets.
- Solving complex problems where traditional methods struggle to find solutions.
- Addressing situations where existing solutions require frequent updates or external interventions.
- Adaptability to variable environments by analyzing new data and applying learned knowledge.

Machine learning algorithms can be categorized into four groups based on whether they use labeled training data and the type of training supervision they receive:

1. Supervised Learning: The training data is correctly classified and labeled, and the algorithm's performance is evaluated based on its ability to predict the correct labels during the test phase. Examples include Decision Trees, K-Nearest Neighbours, and Random Forests.
2. Unsupervised Learning: There is no labeling process involved. The algorithm groups the data based on various properties and observes their associations with each other. It is commonly used for anomaly detection, relationship learning, and dimension reduction.
3. Semi-supervised Learning: This is a hybrid method that combines supervised and unsupervised learning. Only a small portion of the data is labeled, while the rest remains unlabeled. This method offers a balance between the performance of supervised learning and the lower cost of unsupervised learning.
4. Reinforcement Learning: Unlike the other three, this approach involves the algorithm receiving penalties for wrong choices and rewards for correct choices during training. The algorithm constructs its own rules based on these rewards and penalties.

In the project at hand, supervised learning methods will be used because there is access to a manually labeled dataset, which provides a high-performance advantage without incurring significant costs.

The machine learning methods used in the application phase include Random Forest, AdaBoost and K Nearest Neighbours. These algorithms were chosen to bring together popular methods with different characteristics, aiming to provide a comprehensive analysis and comparison of their performance.

3.1 Hyperpameter tuning

Grid search: A technique used in machine learning to systematically search for the optimal combination of hyperparameter values for a given model. Hyperparameters are settings or configurations that are not learned from the training data but need to be specified by the user before training the model.

Grid search works by systematically searching through a predefined grid of hyperparameter values to find the optimal combination for a given machine learning model.

In this report. Grid search systematically explores hyperparameter combinations using k-fold cross-validation with n_folds = 5 and prioritizes the accuracy for evaluation and selection.

Grid search's operation include:

1. Define the parameter grid: Specify the hyperparameters to be tuned and the possible values for each hyperparameter.

2. Create the training and validation sets: Split the available dataset into training and validation sets. To ensure reliable evaluation, use k-fold cross-validation, which divides the data into k equally sized folds and iteratively uses each fold as the validation set while training on the remaining data.
3. Training and evaluation loop: Iterate over all the hyperparameter combinations in the grid. For each combination:

- Train the model: Build and train the machine learning model using the current hyperparameter values on the training set.
- Evaluate the model: Use the trained model to make predictions on the validation set and calculate F1-score.
- Store the results: Keep track of the evaluation metric(s) obtained for each hyperparameter combination.

4. Select the best hyperparameters: After evaluating all the combinations, select the combination that yielded the highest or most desirable evaluation metric(s). This combination represents the optimal hyperparameter values for the model.
5. Optional step: Retrain the model: Once the best hyperparameters are identified, the final step may involve retraining the model using the entire dataset (including the validation set) and the chosen hyperparameters to obtain the final model.

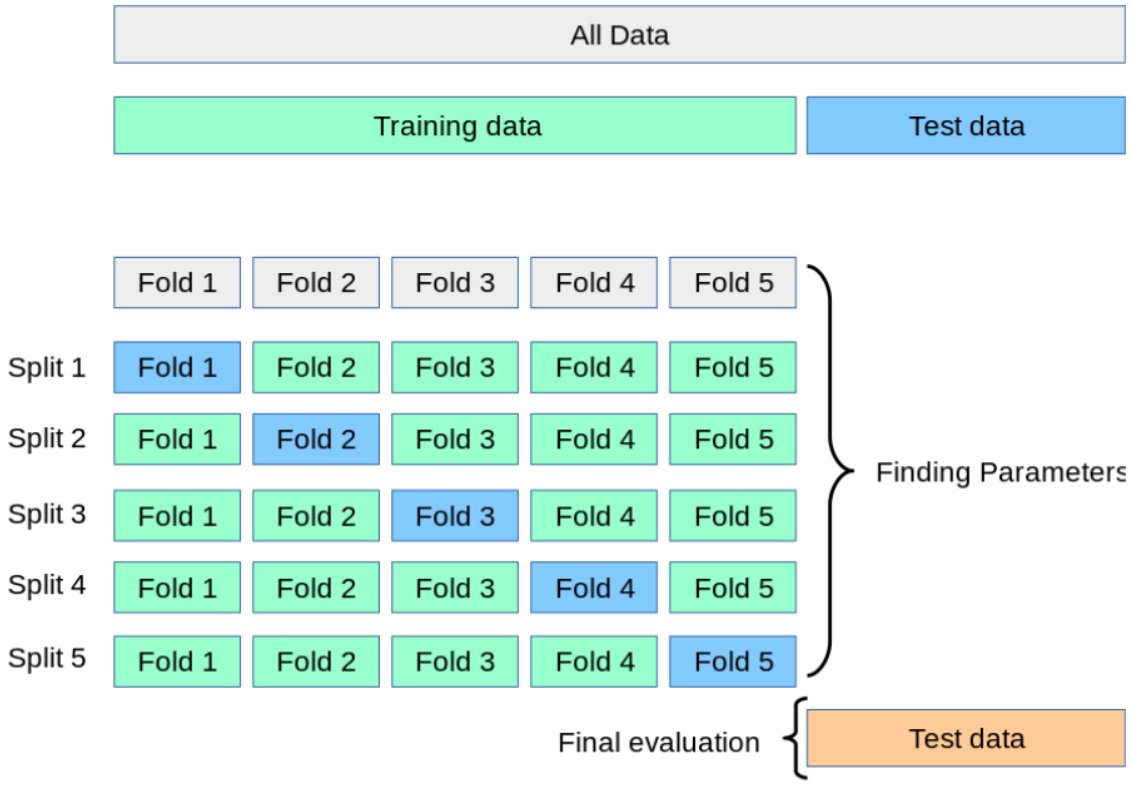


Figure 6. Cross Validation

Whether to use Grid Search for hyperparameter tuning with KNN (K-Nearest Neighbors), Random Forest (RF), and AdaBoost depends on the specific problem and the nature of the hyperparameters involved. Let's discuss each algorithm separately:

K-Nearest Neighbors (KNN):

- KNN is a simple and intuitive algorithm that classifies data points based on the majority class among their k-nearest neighbors in the feature space.
- The primary hyperparameter in KNN is 'k,' which represents the number of neighbors to consider. Other hyperparameters might include distance metrics (e.g., Euclidean, Manhattan) and weighting schemes.
- Since KNN has fewer hyperparameters and they are relatively straightforward to tune, Grid Search might be appropriate to find the optimal 'k' value and other hyperparameters.

Random Forest (RF):

- RF is an ensemble learning method that builds multiple decision trees and combines their predictions to improve accuracy and robustness.
- RF has more hyperparameters to tune, such as the number of trees (n_estimators), the number of features to consider at each split (max_features), and the depth of the trees (max_depth), among others.
- Grid Search can be used to explore different combinations of these hyperparameters to find the optimal configuration.

AdaBoost:

- AdaBoost is another ensemble learning method that combines multiple weak learners (typically decision trees) to create a strong learner.
- The main hyperparameters in AdaBoost include the number of estimators (n_estimators) and the learning rate (learning_rate).
- Like with RF, Grid Search can be used to tune these hyperparameters effectively.

In conclusion, while KNN might require fewer hyperparameters and could be easier to tune manually, Random Forest and AdaBoost tend to have more hyperparameters, making Grid Search a helpful tool to find the best combination.Figure 7 and 8 show the results of applying Grid Search on Random Forest and AdaBoost.

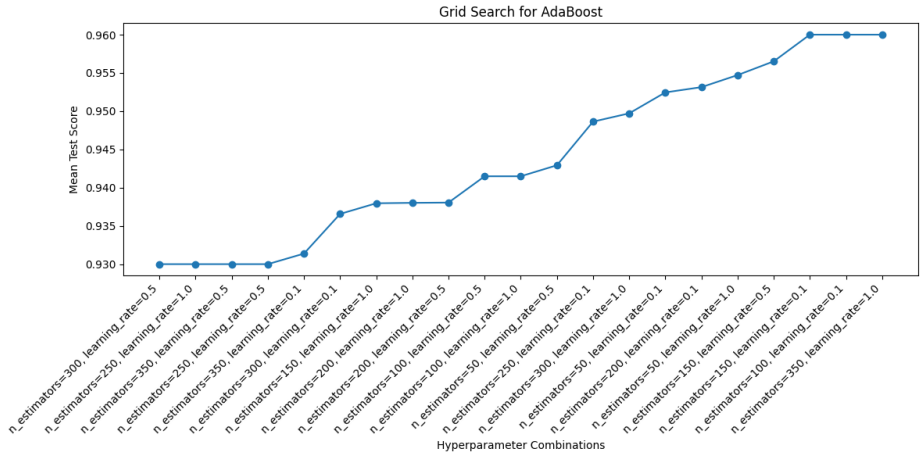


Figure 7. AdaBoost GridSearch result

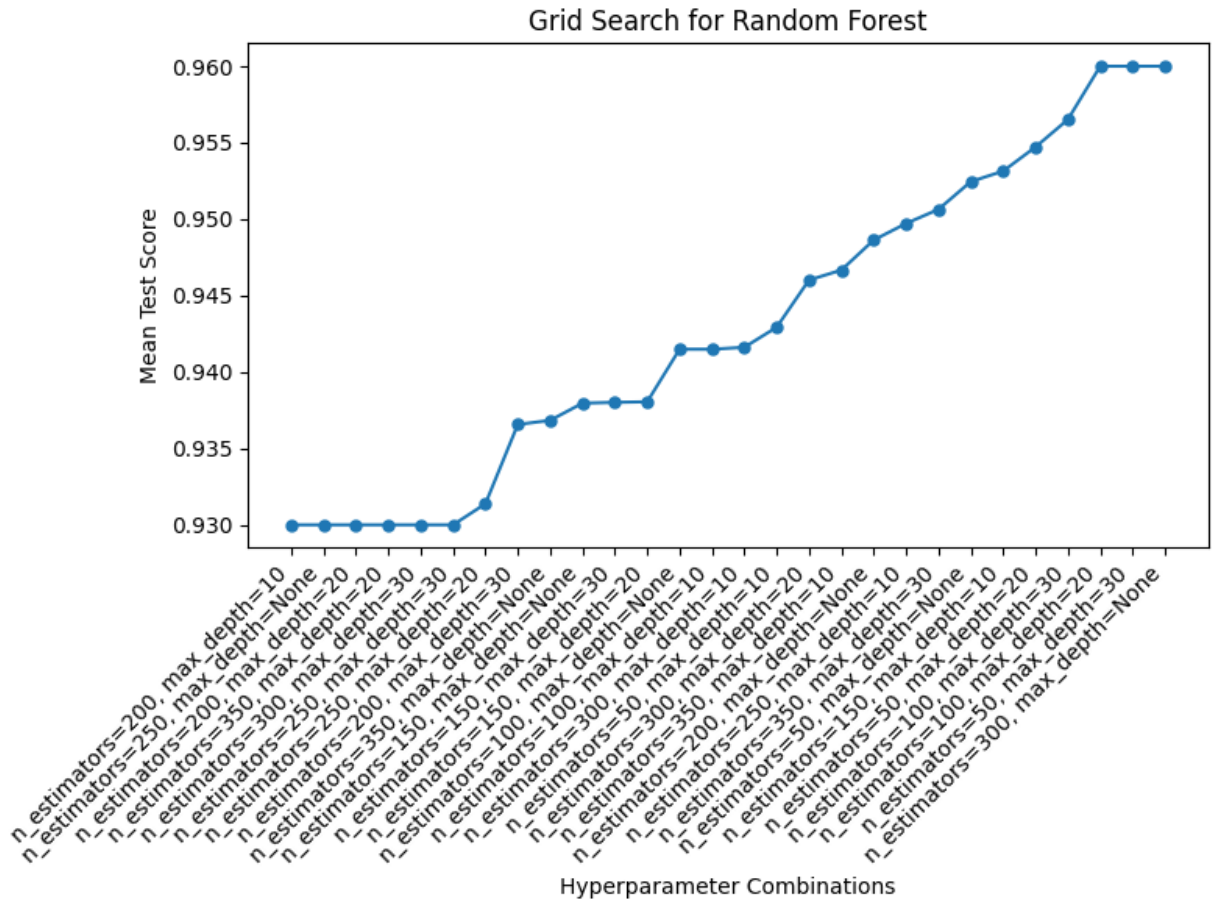


Figure 8. AdaBoost GridSearch result

After using GridSearch and manualy finding best parametters for KNN, we conclude that the hyperparametter of each model is:

KNN: Neighbor=5, p=1 , weigh=distance

Random Forest: Number of eatimators=300, Max Depth=3

AdaBoost: Number of estimators=350, Learning rate=0.1

3.1 Machine learning models

3.1.1 Random Forest

Random forest is a machine learning approach that uses decision trees. In this method, a "forest" is created by assembling a large number of different decision tree structures which are formed in different ways.

Random forest generates N decision trees from training set data. During this process, it randomly resamples the training set for each tree. Thus, n decision trees are obtained, each of which is different from the other. Finally, voting is performed by selecting new estimates from estimates made by N trees. The value with the highest rating is determined as the final value.

The random forest has many advantages. These can be listed as follows :

- This algorithm can work well with very large and complicated datasets.
- The overfitting problem frequently encountered by decision trees is very rare in this algorithm.
- It can be applied to a lot of kind of machine learning problem.
- It is also good to deal with missing values in the data set. It replaces these lost values with their own created values.
- In addition, it calculates and uses the importance level of the variables when making the classification. Thanks to this, it is also used for feature selection in the machine learning.

On the other hand, the structure of Random forest is quite complicated because it is made up of many decision trees. Another disadvantage is that It is pretty difficult to understand its functioning.

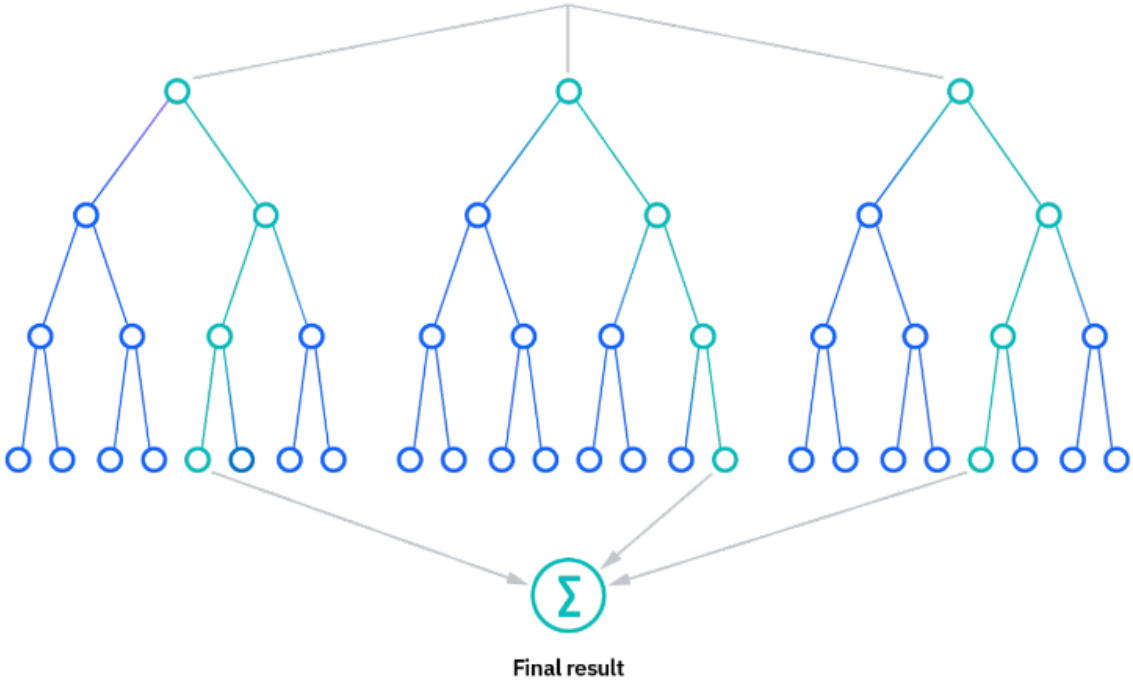


Figure 9. Random Forest

3.1.2 KNN (K-Nearest Neighbor)

(K Nearest Neighbour), which is a sample-based method, is one of the most used machine learning algorithms with its simple and fast structure. This algorithm depends on the assumption that the examples in a dataset will exist close to the examples with similar properties in another dataset.

In this context, KNN identifies the class of new data that is not classifiable by using training data of known class type. This determination is made by observing the nearest neighbours of the new sample, for which no classifications are specified .

In a plane with N properties, the number of neighbours to be looked at for an unclassified sample is specified by the number K. For the unknown sample, the distances to the neighbours are calculated and the smallest K numbers are chosen from these distance values. The most repeated property within the K values is assigned as the unknown instance property. In Figure 10, a visual is created for the values 2 and 5 of K in a 3-dimensional plane (N = 3).

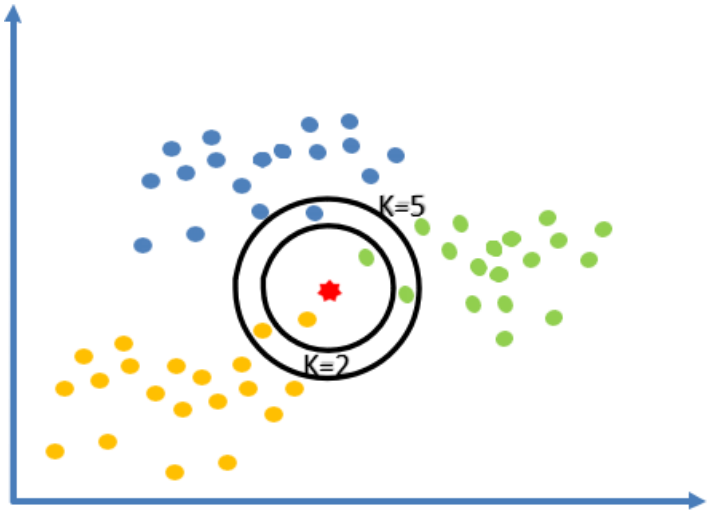


Figure 10. Operation of KNN algorithm for K = 2 and K = 5 values.

KNN, which provides good performance over multidimensional data and is a fast algorithm during the training phase, is relatively slow in the estimation stage.

3.1.2 AdaBoost

AdaBoost (Adaptive Boosting), a boosting method, is a machine learning algorithm developed to improve classification performance. The basic working principle of Boosting algorithms can be explained as follows: The data are first divided into groups with rough draft rules. Whenever the algorithm is run, new rules are added to this rough draft rules. In this way, many weak and low performance rules called "basic rules" are obtained .

Once the algorithm has been working many times, these weak rules are combined into a single rule that is much stronger and more successful. During this process, the Algorithm assigns a weighting coefficient to each weak rule, giving the highest coefficient value to the lowest error rate. These weight values come into play when final rules are selected. The final rule is created by giving priority to the high scored weak rules .

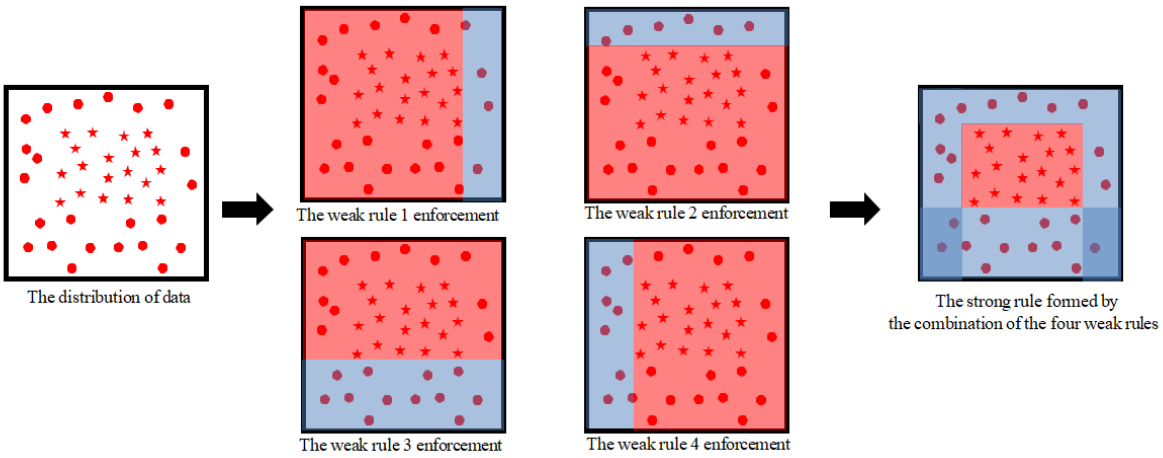


Figure 11. Demonstration of the operation of the AdaBoost algorithm.

3.3 Implementation of Machine Learning Algorithms

Two distinct approaches have been adopted to apply machine learning algorithms to the dataset. In the first method, separate files are created for each attack type and the benign data, with the attack data accounting for 30% and benign data 70% of each file. Each file is labeled according to the type of attack it contains. The seven chosen machine learning methods are applied ten times to each of these files, resulting in distinct outcomes for each attack type. This approach aims to assess the effectiveness and performance of various machine learning methods on different types of attacks.

In the second approach, the entire dataset is utilized as a single file. All attacks present in this file are combined under a common label, namely "attack," while the benign data retains its original label. For feature selection, the four attributes with the highest importance weight achieved for each attack type in the first approach are aggregated. Consequently, 4 features are obtained from each of the 12 attack types, forming a pool of features comprising 48 attributes. After eliminating duplicates, the number of features is reduced to 18. The list of these 18 features can be found in Table 5.

In summary, the first approach analyzes the performance of machine learning methods on individual attack types, while the second approach aggregates the most important features from each attack type to create a unified set of features for further analysis.

Bwd Packet Length Max	Flow IAT Mean	Fwd Packet Length Min
Bwd Packet Length Mean	Flow IAT Min	Fwd Packet Length Std
Bwd Packet Length Std	Flow IAT Std	Total Backward Packets
Flow Bytes/s	Fwd IAT Total	Total Fwd Packets
Flow Duration	Fwd Packet Length Max	Total Length of Bwd Packets
Flow IAT Max	Fwd Packet Length Mean	Total Length of Fwd Packets

Table 5. The feature list created for all attack types.

An alternate way of implementing this approach is to use features with high weighting according to the importance scores obtained for the entire dataset in the Feature Selection According to Attack or Benign section, without using all of these eighteen features above. The threshold value for feature weight is set at 0.8%. In this way, 97% of the total feature importance weight will be covered by selecting only 7 features. The remaining 13 features constitute only 3% of the total weight of significance. If the features with a weight of 0.8% and above are selected, the following features are used:

Feature Name	**Importance Weight**	**Percentage**
Bwd Packet Length Std	0.246627	38.97%
Flow Bytes/s	0.178777	28.25%
Total Length of Fwd Packets	0.102417	16.18%
Fwd Packet Length Std	0.063889	10.10%
Flow IAT Std	0.009898	1.56%
Flow IAT Min	0.006946	1.10%
Fwd IAT Total	0.005121	0.8 %

Table 6. The importance weights obtained in "Feature Selection According to Attack or Benign" section.

4. RESULTS

In this section, the results of the studies done in the implementation section are presented. In this context, in the assessment carried out, the evaluation criteria are presented via the data of the F-measure. However, all the evaluation data obtained can be accessed from the Appendix.

The performance evaluation procedures are repeated 10 times for each machine learning algorithm. The numbers given in the tables are the arithmetic mean of these 10 processes. Box and whisker graphs are created to illustrate the consistency of the results and the change between them.

4.1 Approach 1 - Using 12 Attack Types

The performance evaluation procedures are repeated 10 times for each machine learning algorithm. The numbers given in the tables are the arithmetic mean of these 10 processes. Box and whisker graphs are created to illustrate the consistency of the results and the change between them.

Three different machine learning methods are applied to 12 different attack types and the obtained results are presented in Table 7.

Among the results presented in Table 7, the biggest and smallest achievements are emphasized as follows: The greatest scores are bold, the smallest scores are underlined and italics. In the results of the algorithms, if there is an equality in F-measure, the following values are examined in order to eliminate equality, respectively: accuracy, precision, recall, and time.

Attack Names	F-Measures		
	RF	KNN	AB
Bot	0.96	0.95	**0.97**
DDoS	0.96	0.92	0.96
DoS GoldenEye	0.99	0.98	0.99
DoS Hulk	0.93	0.96	0.96
DoS Slowhttptest	0.98	0.99	**0.99**
DoS slowloris	0.95	0.95	0.95
FTP-Patator	1.00	1.00	1.00
Heartbleed	0.99	1.00	0.93
Infiltration	0.92	0.88	**0.92**
PortScan	1.00	1.00	1.00
SSH-Patator	0.96	0.95	0.96
Web Attack	0.97	0.93	0.97

Table 7. Distribution of results according to type of attack and machine learning algorithm.

After analyzing the F-Measure results for different machine learning methods on various attack types, we can draw the following conclusions:

- Detection of Bot, DDoS, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, FTP-Patator, PortScan, and SSH-Patator attacks: The Random Forest (RF), K Nearest Neighbours (KNN), and AdaBoost (AB) methods exhibited outstanding performance, achieving high F-Measure values close to 1.00 for these attack types. This indicates that these methods are highly effective in accurately detecting and classifying these attacks.
- Heartbleed attack: The Random Forest and AdaBoost methods achieved high F-Measure values, while K Nearest Neighbours showed a slightly lower F-Measure value of 0.93. Nonetheless, all three methods demonstrated promising results in identifying the Heartbleed attack.
- Infiltration and Web Attack: The Random Forest method outperformed the other methods, achieving the highest F-Measure values of 0.92 and 0.97, respectively. Although K Nearest Neighbours and AdaBoost showed slightly lower F-Measure values, they still exhibited reasonable performance in detecting these attack types.

In summary, the Random Forest method consistently displayed effectiveness and robustness across various attack types. While K Nearest Neighbours and AdaBoost also demonstrated strong performance, the combination of these three machine learning methods can be a reliable approach for detecting a wide range of attacks in the dataset.

4.2 Approach 2 - Using Two Groups: Attack and Benign

In this section, the entire data set is used as a single dataset file. All attacks contained in this file are collected under a single common name, "attack". Seven different machine learning methods are applied to this dataset. In this approach, two methods will be used, the first one, the features created for attack files

in approach 1 are used. In the second method, the 7 features obtained in the Feature Selection According to Attack or Benign section are used.

4.2.1 Using Features Extracted for Attacks Files

Table 8 shows the results obtained by using 18 extracted from the first approach.

Machine Learning Algorithms	**Evaluation Criteria**				
	F-Measure	Precision	Recall	Accuracy	Time[5]
Random Forest	0.94	0.95	0.94	0.94	24.739
AdaBoost	0.95	0.95	0.95	0.95	391.804
K Nearest Neighbours	**0.96**	0.96	0.97	0.97	*1967.054*

Table 8. Application of the features obtained in the first approach.

The F-Measure, Precision, Recall, and Accuracy are all performance metrics used to assess the effectiveness of the algorithms. A higher value indicates better performance in detecting and classifying attacks. In terms of F-Measure, K Nearest Neighbours achieved the highest score of 0.96, followed closely by AdaBoost with a score of 0.95, and Random Forest with a score of 0.94.

All three algorithms demonstrated high precision and recall values, indicating their ability to correctly classify both positive (attack) and negative (benign) instances. K Nearest Neighbours showed the highest recall of 0.97, indicating it effectively identified true positive instances.

Accuracy, which measures overall correctness, was also high for all three algorithms, with K Nearest Neighbours achieving the highest accuracy of 0.97.

However, it's essential to consider the trade-off between performance and time taken. Random Forest is the fastest among the three algorithms, while K Nearest Neighbours took the most time for processing.

Overall, K Nearest Neighbours appears to be the top-performing algorithm based on the evaluation criteria, as it demonstrated high F-Measure, Precision, Recall, and Accuracy scores. However, the choice of the best algorithm may also depend on other factors, such as computational resources and specific application requirements.

4.2.2 Using Features Selection for All Dataset

An implementation that uses an alternative feature selection method can be seen in Table 9. In this method, the 7 features obtained in the Feature Selection According to Attack or Benign section are used. The changing parts are highlighted in red. When Table 8 and Table 9 are compared, there is no significant change in the algorithms of Random Forest, AdaBoost on F-measure.

From speed perspective, the running times of all algorithms are noticeably reduced. The reason for this reduction in execution time is that 18 attributes are used in the method applied in Table 8, whereas only 7 attributes are used in Table 9. This reduction in feature count has reduced the running time of machine learning algorithms.

Machine Learning Algorithms	Evaluation Criteria				
	F-Measure	Precision	Recall	Accuracy	Time
Random Forest	0.94	0.947	0.94	0.94	20.511
ID3	0.95	0.95	0.95	0.95	11.552
K Nearest Neighbours	0.97	0.97	0.97	0.97	1038.253

Table 9. Implementation of features obtained using Random Forest Regressor for All Dataset.

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this study, two different approaches were utilized to apply machine learning algorithms to the dataset. Approach 1 involved using 12 distinct attack types, and Approach 2 focused on combining attacks into two groups: "attack" and "benign." The performance of various machine learning methods was evaluated using F-Measure, Precision, Recall, and Accuracy metrics.

In Approach 1, the results demonstrated that Random Forest, K Nearest Neighbours, and AdaBoost consistently achieved high F-Measure scores for most attack types. These methods showed excellent performance in accurately detecting and classifying different attacks. Notably, Random Forest exhibited consistent effectiveness across various attack types, making it a reliable choice for this scenario.

In Approach 2, using features extracted from the first approach, K Nearest Neighbours emerged as the top-performing algorithm with the highest F-Measure, Precision, Recall, and Accuracy scores. This method showcased its ability to effectively detect both attack and benign instances, making it a promising approach for broader classification tasks.

An alternative feature selection method was employed in the second part of Approach 2, reducing the feature count from 18 to 7 attributes. The results demonstrated that Random Forest and AdaBoost algorithms maintained their high F-Measure performance, while ID3 and K Nearest Neighbours also delivered promising results.

Overall, K Nearest Neighbours consistently displayed strong performance in both approaches, making it a preferred choice for this specific dataset. However, the selection of the optimal machine learning algorithm may vary depending on factors like computational resources and specific application requirements.

In conclusion, this study provides valuable insights into the effectiveness of different machine learning methods for intrusion detection tasks using the CICIDS2017 dataset. The combination of Random Forest, AdaBoost, and K Nearest Neighbours algorithms appears to be a reliable approach for detecting a wide range of attacks. By utilizing these findings, cybersecurity practitioners can enhance their ability to detect and defend against various threats in real-world scenarios.

5.2 Future work

As we forge ahead in the quest for enhanced intrusion detection systems, several promising avenues beckon us to explore and refine our approaches. By incorporating these cutting-edge techniques, we aim to construct more robust and accurate models to safeguard our networks against potential security threats in real-world environments.

- Feature Engineering: Let us delve deeper into the realm of feature engineering, meticulously investigating and exploring additional network features. Through this in-depth exploration, we seek to uncover novel insights that can potentially propel the performance of our models to greater heights. The adoption of advanced feature engineering techniques holds the promise of yielding more accurate and reliable intrusion detection systems.
- Ensemble Methods: Embracing the power of synergy, we shall embark on a journey to combine the strength of multiple models through ensemble techniques like stacking, blending, or bagging. By doing so, we anticipate that our intrusion detection systems will exhibit a significant boost in accuracy, ensuring a more formidable defense against cyber intrusions.
- Anomaly Detection: We are inspired to augment the prowess of traditional supervised learning models with the complementary capabilities of anomaly detection techniques. This fusion will result in a more comprehensive and adaptive approach to intrusion detection, enabling our systems to detect and thwart even the most elusive threats.
- Deep Learning: In the realm of deep learning, we envision leveraging cutting-edge architectures such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs). These deep learning models possess the capability to autonomously learn hierarchical representations from data, showcasing remarkable potential in various domains. By applying them to intrusion detection, we strive to achieve unparalleled accuracy and robustness.
- Real-Time Monitoring: As the stakes of network security grow higher, we recognize the significance of real-time monitoring. Deploying our intrusion detection systems in live network environments will allow us to assess their performance and reliability in real-world scenarios. Through meticulous optimization, we aspire to ensure timely and precise detection and prevention of intrusions.
- Imbalanced Data Handling: A critical concern that demands our attention is the issue of imbalanced data in the dataset. To address this challenge, we shall implement techniques like oversampling, undersampling, or using class weights. This concerted effort will lead to improved performance on both normal and intrusion classes, striking a balance in our detection system.
- Dynamic Updating: In the face of constantly evolving threats and attack patterns, the need for adaptability becomes paramount. We shall explore and devise methods for dynamic updating and continuous learning of our intrusion detection system. This dynamic nature will empower our models to stay resilient and proactive in countering emerging cybersecurity threats.

By charting these future directions, we embark on a transformative journey towards building intrusion detection systems that transcend the limits of today's technology. Together, let us forge ahead in fortifying our networks, ensuring a safer and more secure digital landscape for all.

ACKNOWLEDGEMENT

In the pursuit of scientific inquiry, we wish to acknowledge and extend our deepest gratitude to the individuals who have played a crucial role in the successful completion of this project.

Foremost, we express our profound appreciation to Professor. Nguyen Linh Giang for his invaluable contribution. His gracious permission to undertake this project, along with her unwavering support and meticulous guidance, have been pivotal in our scientific journey. Under his expert tutelage, we have gained profound insights into Applied Statistics and Experimental Design, augmenting not only our present knowledge but also equipping us with essential skills for our future scientific endeavors.

Furthermore, we would like to extend our sincere thanks to our classmates, whose selfless dedication and arduous efforts have been instrumental in our research. Their willingness to devote their time and expertise to answer our queries and provide support for our report has significantly bolstered the quality and comprehensiveness of our findings.

We are acutely aware of the invaluable contributions made by these esteemed individuals, and it is with great humility and gratitude that we recognize their unwavering commitment to our project. Their assistance has undoubtedly propelled our scientific exploration to new heights and enriched our understanding of the subject matter.