HYBRID FEATURE SELECTION IN NETWORK INTRUSION DETECTION USING DECISION TREE

by

Chenxi Xiong

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer and Information Technology

West Lafayette, Indiana

August 2020

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Baijian Yang, Chair

Department of Computer and Information Technology

Dr. Robert Deadman, Co-Chair

Department of Computer and Information Technology

Dr. Wenhai Sun, Committee Member

Department of Computer and Information Technology

Approved by:

Dr. Eric Matson

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

FTP	File Transfer Protocol
SSH	Secure Shell
RDP	Remote Desktop Protocol
TCP	Transmission Control Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
RST	Reset
FIN	Finished
SYN	Synchronize
PSH	Push
ACK	Acknowledgment
SMTP	Simple Mail Transfer Protocol
IRC	Internet Relay Chat
DNS	Domain Name System
XSS	Cross-site scripting
SQL	Structured Query Language
HTML	Hypertext Markup Language
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocols.
IPFIX	Internet Protocol Flow Information Export
SVM	Support Vector Machine
DT	Decision Tree
BN	Naïve Bayes
CWR	Congestion Window Reduced
URG	Urgent
ECE	ECN-Echo
PDU	Protocol Data Unit
RFE	Recursive Feature Elimination

GLOSSARY

Network Intrusion detection system: An automated system that checks all inbound and outbound traffic on the network to detect network activity with malicious or abnormal patterns, and sends an alert message to users for attack notification.

Machine Learning: An application of artificial intelligence that applies statistical algorithms to interpret and learn the patterns of large amount of data, and automatically provides prediction based on new data.

Feature Selection: The process of selecting a subset of input variables or features that may contribute most to the prediction result from numerous variables to save computational costs and improve the performance of the machine learning model.

ABSTRACT

The intrusion detection system has been widely studied and deployed by researchers for providing better security to computer networks. The increasing of the attack volume and the dramatic advancement of the machine learning make the cooperation between the intrusion detection system and machine learning a hot topic and a promising solution for the cybersecurity. Machine learning usually involves the training process using huge amount of sample data. Since the huge input data may cause a negative effect on the training and detection performance of the machine learning model. Feature selection becomes a crucial technique to rule out the irrelevant and redundant features from the dataset. This study applied a feature selection approach that combines the advanced feature selection algorithms and attacks characteristic features to produce the optimal feature subset for the machine learning model in network intrusion detection. The optimal feature subset was created using the CSE-CIC-IDS2018 dataset, which is the most up-to-date benchmark dataset with comprehensive attack diversity and features. The result of the experiment was produced using machine learning models with decision tree classifier and analyzed with respect to the accuracy, precision, recall, and f1 score.

CHAPTER 1. INTRODUCTION

This chapter describes relevant background of intrusion detection and machine learning, problem existing in the current research of related topic, significance and purpose of the research. This chapter also states research question, assumptions, limitations and delimitations of the study.

1.1 Background

Nowadays, computer network has been applied to every aspect of people's life and daily production. People use desktop computers, laptop computers, or other types of Internet-enabled devices to access public information that has been published online. Different organizations, such as companies and schools, build up the networks to exchange the information within the organizations. According to statistics from the National Telecommunications and Information Administration, by November 2017, more than 240 million people in the United States used the Internet, accounting for about 77% of the US population. The line chart from National Telecommunications and Information Administration in Figure 1.1 in the next page shows that the number of Internet use in the U.S. fluctuated up and down for no more than 10% of the American population from 2010 to 2014, but it is steadily increasing at the whole [1].

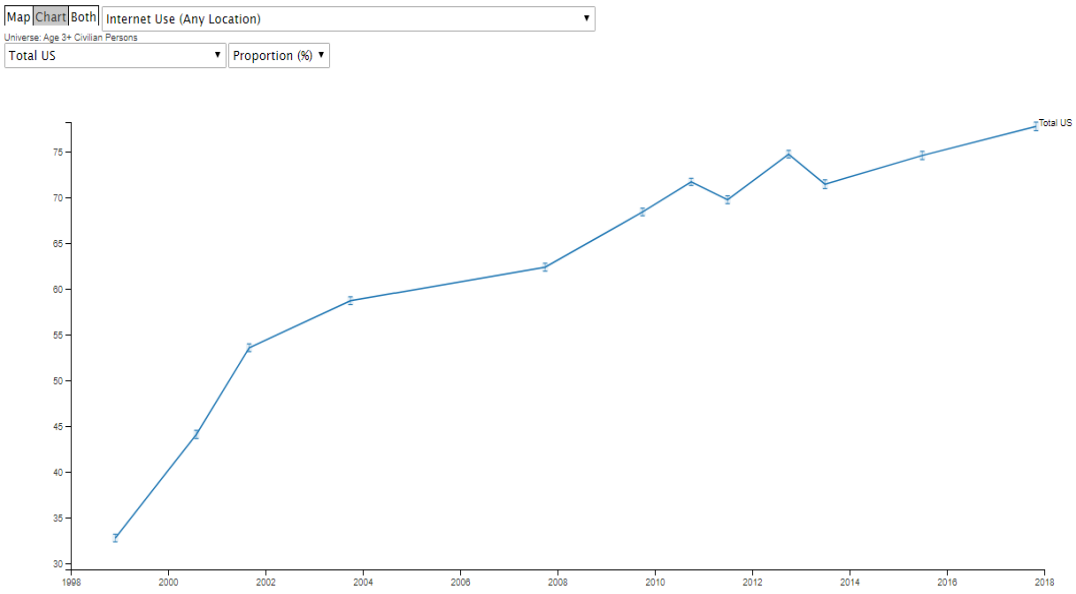


Figure 1.1 Internet Use in the U.S

Source: Adapted from [1]

However, the wide spreading and developing of computer network accompanies with the security challenges for people's privacy and assets that are stored or transmitted through the computer network. In the past few years, several critical data breach incidents have been reported, and caused sensations in the world. The latest incident of data breach that also called publics a great attention was the database breach of Capital One on March 2019. Forbes reported that Capital One misconfigured the Amazon server making it a vulnerable target. As a former employee of Amazon, the hacker in this case got the access to the login information of the company's computer and server, and then stole the data. Over 100 million credit card users of Capital One were influenced. And the hacking incident costed Capital One 100 million to 150 million US dollars [2]. From this case, it is not difficult to find that the network attack could cause extremely critical damage to people and organization's privacy and economic property. Thus, network security has become one of the major concerns of most people and organizations when using computers and other Internet-enabled devices to access online resources and store valuable data.

Generally, intrusion detection systems are used as powerful tools to protect network security by detecting network attacks from malicious users on the Internet. These systems examine the network traffic by analyzing the packet information at different layers of communication model [3]. In recent years, more and more researchers have started to use machine learning approaches to enhance the intrusion detection system and solve network security issues, including detecting and classifying network abnormal activities by making the system learn different attacks from the sample data [4]. Artificial intelligence is becoming more and more advanced and is used in various fields of technology. As part of artificial intelligence, machine learning also has great potential in the field of cybersecurity.

The general process of machine learning could be straightforward and comprehensible, but it is never limited to what is going to be presented in this research. When researchers apply the machine learning approaches, they need to identify the features of sample data that are summarized patterns of the sample data in the packet of network traffic. For example, IP address, protocol, port number, etc. The researchers use features to train models and reach the higher identification rate and accuracy, and then apply the trained machine learning models to further detect and classify the network attacks in the network traffic. However, when using machine learning to study different problems in the same field, the features used for training and testing may not necessarily be the same. For instance, detecting DDoS attacks may require investigating different packet

information from detecting spam. Hence, having solid domain knowledge and choosing the right features play an important role in machine learning.

1.2 The Problem

Applying machine learning in network intrusion detection is well studied. More than a hundred of research papers about machine learning in network intrusion detection system were published in various conferences every year. It is interesting to find that the researchers are so enthusiastic about trying various machine learning algorithms in network intrusion detection. However, at the same time, lots of researchers are stingy at an explanation about the reason why they choose some certain features or simply use up all features in the dataset except some researchers who are focusing on the feature engineering aspect of machine learning.

After taking a deeper look at the machine learning, it is easy to notice that using a large number of non-representative features could create the tough problem for creating an effective and accurate machine learning model. Feeding a large amount of data that could possibly create millions of combinations to a machine learning model could make it hard to distinguish the attack patterns during the monitoring process [5]. On the other hand, redundant or irrelevant feature is one of the most significant factors that cause the excessive training and classification time [6]. However, the importance of feature selection is often neglected or underrepresented by some researchers [7]. Feature selection plays a significant role on creating the machine learning model for classifying or predicting purposes. Each different kind of network attacks has its specific attack patterns that could be discovered in the data sorted in different features. The process of feature selection is to verify whether all features in the dataset play equally significant roles for identifying these network attacks in the certain machine learning model and tells if it is necessary to feed all the data to a machine learning algorithm to get a decent rate of identification and accuracy. One study indicated that since high dimensionality of data causes critical decrease of the performance of intrusion detection system, finding the best feature subset that could mostly increase the detection accuracy and eliminate the distraction of irrelevant features becomes a key task [8].

Regarding to the issues mentioned above, the problem addressed by this proposed study is that can domain knowledge-based features work with feature selection to boost the efficiency of the machine learning models in network intrusion detection.

1.3 Significance

In many researches about the machine learning in network security, the significance of feature selection solution has been emphasized repeatedly. Applying the advanced feature selection approaches could bring significant improvement to the performance of network intrusion detection system, Cai et al. indicated that "good feature selection results can improve learning accuracy, reduce learning time, and simplify learning results [9]." On one hand, using key features that are interrelated with each other with machine learning approach can effectively reduce the iterations of the experiment [10]. On the other hand, obtaining the optimal feature subset could help to effectively shrink the size of features that will be used for training the machine learning model, which could further lower the impact of over-fitting and provide better generalization of models [11]. Besides, less features fed into machine learning model save a lot of time on data processing and model training [12][11]. Eliminating the irrelevant features by applying the feature selection techniques improves the classification accuracy of machine learning model [13].

1.4 The Purpose

The purpose of the project is to design a new feature selection approach to enhance the machine learning models in network intrusion detection by creating the optimal subset of features.

By finishing this study, multiple deliverables would be presented as follows. The importance of different features in network intrusion detection would be identified based on the characteristics of each kind of network attack. The optimal combinations of features for each network attack in the study would be identified by comparing the detection rate of different combinations. And the statistical analysis would be conducted by the project to explain the improvement on the performance of the machine learning model using the optimal combination of features with respect to the detection rate, accuracy, training time, testing time, etc.

1.5 Research Question

- Is a hybrid feature selection approach that combines feature selection algorithms and attack characteristic features able to help machine learning model to produce better predictions in network intrusion detection?

1.6 Assumption

The assumptions for the research are:

- The proposed feature selection approach will create feature subsets that contain less than 15 features, which is the average size of feature subset in other related researches.
- The selected subsets of features will help the machine learning model achieve the same or better accuracy of network attack identification than that of using all features in the given dataset.
- The selected subset of features will help to shorten the training and testing time of the machine learning model that uses all features in the given dataset.

1.7 Delimitations

- All assumptions mentioned above are only dictated by the CSE-CIC-IDS2018 dataset.
- The attacks and features that are tested in this research are limited to those in the CSE-CIC-IDS2018 dataset.
- The proposed model only performs the tasks of intrusion detection. The prevention function is not studied in this research.
- Only statistical methods of machine learning are considered as evaluation models to examine the results of experiment. No neural networks or deep learning are used in this research.

1.8 Limitations

The data used in this project is limited to the published dataset. Due the time limit of project, there is not enough time to capture the data traffic myself.

1.9 Summary

This chapter introduced the background information of machine learning in network intrusion detection and feature selection. This chapter also established the problem statement that showed the lack of attention to feature selection in the current study about the machine learning in network intrusion detection, as well as the significance of the problems, the research question, goals, assumptions, delimitations, and limitations of this project.

CHAPTER 2. REVIEW OF LITERATURE

This chapter first introduces common network attacks including brute-force attack, botnet attack, web attacks, and infiltration attack. All listed attacks are explained in terms of characteristics and detection techniques. This chapter also discusses three feature selection methods: filter method, wrapper method, and embedded method. The differences between the three kinds of feature selection methods are explained in detail. In addition, the man-made features (derived features or constructed features) targeting various network attacks are introduced in this chapter.

2.1 Network Attacks

2.1.1 Brute-force

Since most people use passwords as a security measure to protect the sensitive information that is stored in the electrical devices or online cloud servers. The password has been targeted by the hackers for a very long time. The brute-force attack describes a network attack that the malicious actor tries to crack other users' password by guessing the password to gain the access to the private information of others or control of websites or servers. There are two common methods for performing brute-force attack: exhaustive key search attack (trial-and-error method) and password-spray attack (low-and-slow method).

The brute-force attack has been used frequently in the computer network. However, it is not difficult to detect this kind of attack. The traditional intrusion detection system could effectively detect the exhaustive key search-based brute-force attack by setting the maximum number of login attempts. The exhaustive key search attack requires the hackers to try all the potential combinations of passwords in the key space until the right key is found [14]. When the maximum login attempts are reached, the intrusion detection system would detect the abnormal activity and raises the alarm. And the account would be locked for preventing further attempts from hackers. However, the password-spray attack overcomes the drawback of the exhaustive key search. It allows the hackers to continually try the same password for different accounts to avoid alert from the maximum attempts policy that could be activated by multiple unsuccessful login attempts on the same account [15]. Although, the improvement of password-spray attack makes it

to be detected much more difficult. There is another approach that is introduced to detect the brute-force attack.

Instead of detecting the brute-force attack after several attempts from the hackers, verifying the fix patterns in traffic packet of brute-force attack becomes a more efficient approach to detect the brute-force attack. This approach is called signature-based approach. The signature-based approach basically focuses on verifying six flow features and statistics that are summarized by researchers observing a large amount of attack traffic [16]:

1. Port number of common services
2. Port number range of connected clients
3. TCP flags
4. Flow duration
5. Number of transferred packets (incoming and outgoing traffics)
6. Number of transferred bytes (incoming and outgoing traffics)

Firstly, based on many researches about the brute-force attack, it could be asserted that the hackers could launch the brute-force attack on various services of the connected devices on the Internet. The services that are targeted most by the brute-force attack include FTP, SSH, Telnet, RDP, HTTP, and HTTPS [17]. In order to attack the services, the hackers need to target on the respective ports of the services on the victims' devices. The corresponding port numbers of the services mentioned above are TCP 21, 22, 23, 80, 443, and 3389 [18].

Secondly, if the hackers want to establish a data connection between their machines and the target machines, they must use one of the available short-lived ports, ephemeral ports, from the TCP/IP stack on their currently used machines as a source port. According to the study conducted by other researchers, different operating systems use different ranges for the ephemeral ports, but overall the ephemeral ports are falling between 1024 to 65535 [20]. And the destination ports of the attack traffic are usually the ports of the target services, for example port 22 for SSH and port 23 for Telnet.

Thirdly, since the brute-force attacks frequently target to the services that use TCP protocols, the TCP flags could be one of the significant features to distinguish the normal traffic and the brute force attacks. Based on the observations in other studies, the brute force attack could be launched after a successful TCP connection being established. And a set of TCP flags would always show in the traffic packet of brute force attacks, including RST, FIN, SYN, PSH, and ACK

flags [21]. However, the normal TCP connection usually contains a part of TCP, SYN, and ACK flags.

Fourthly, the flow duration of brute force attack is noticeably short than the normal data transmission. Since the brute force attack creates multiple login attempts that establish the data flows containing very few data inside the traffic packets. The flow duration could be limited to a very short period of time from 100ms to 5s [22].

Lastly, the data flow of brute force attack has its specific characteristics in number and size of the packets inside of each flow.

2.1.2 Botnet

With the development of network technology, more and more devices are connecting to the Internet. However, since a lot of users lack the awareness of cyber threats, their devices may be infected by viruses, for example Trojan horse, and used as a part of botnet for bad intentions. Botnet attack defines such kind of network attack that the hackers obtain the control of other users' computers by infecting their computers with the malware over the Internet and connect all the infected computers as bots with command and control server to form a malicious botnet [23]. This malicious botnet is commonly used by hackers to launch many other kinds of network attacks, including Denial of Service attack, phishing attack, and web attack [24].

A common approach for detecting botnet attack is to analyze the network traffics. The typical attributes of network traffic include the source and destination address, the protocols, the related port numbers, the duration of the session, the number of incoming and outgoing packets, and the cumulative size of the transmitted packets [25]. The detector usually checks the IP addresses of network traffic to see if the IP addresses have been identified as suspicious addresses with abnormal activities before. The bots, the compromised computers, usually use the specific protocol and associated port to establish the communication with each other or infect other target computers [26]. For example, port 25 for SMTP, port 80 for HTTP, port 6667 for IRC, and port 53 for DNS [27]. Counting the packets transmitted in the network and their size shows the significant meaning for detecting the communications between bots or between bot and target. It could indicate the unusual behavior of the botnet attack by showing huge amount of data transmission within a relatively short of time, multiple packets with identical size, etc. [28].

DNS based approach is another popular botnet detection technique. This approach focuses on verifying the information from DNS queries. The hackers create malicious website or other servers with the domain name that is similar to the legitimate domain name. For example, amazcn.com instead of amazon.com. Some users may mistype the domain name to access the malicious servers and get infected by malware from hackers. The hackers also conceal the communication between the bot master and bot by using DNS service to hide the IP address of the bot master. However, DNS based approach targets to identify the compromised computers by examining the traffic from the suspicious domains or traffics from various machines to the same domain repeatedly in the similar manner within a short period of time [29].

2.1.3 Web Attacks

The development of the web applications makes the Internet more functional and productive. People could access many software applications through the Internet and store the data online. The common web applications that people may use every day are online office suite, online video streaming platform, webmail, e-commerce platform, etc. But the sensitive data of users handled by these web applications may be threatened by the web attacks. Web attacks aim to break through the security of web services by using the weakness of the programs and hijack the data traffic between the client and server. XSS attack and SQL injection are two types of common web attacks [30].

XSS attack is targeting to exploit the communication between the end users and web applications though the malicious script. The XSS attack could be launched to obtain users' cookie information by injecting malicious scripts into the HTML content of a web page. When the user accesses this website, the malicious script would be activated and steal the user's session cookie. The cookie information may reveal users' sensitive information, like username, password, etc. SQL injection allows the hackers to control the SQL database that is connected to the web application by injecting a SQL query though a user input. The hackers discover the vulnerable user input of a web page that directly uses user input in SQL query. After changing the SQL code, the hackers may retrieve, modify, even delete the information in the database [31].

Some significant components of web request gain a lot of attention for detecting the web attacks. These elements include Source IP, HTTP Method, forwarded and responded HTTP data, and requested URI.

2.1.4 Infiltration Attacks

By looking at the software released on the Internet, none of them is perfectly secured without any vulnerability. However, the hackers may use the weaknesses the software to break into victims' computer and launch the attacks from the inside of the internal network. The hackers may target to various software, including document viewer, web browser, program development tools, etc. [32]. The malware will infect users' computer through the vulnerable software. Then the hackers launch the port scanning attack to discover other potential victims in the internal network. The port scanning attacks identify the potential victims by sending packets to the hosts in the network through every possible port. The status of the ports will be acknowledged by checking the responses, they are either open or closed. The response packets also contain the target information, for example the IP address, protocol, etc. Leaked information may be used by hackers to launch more network attacks [33].

Even though there are many kinds of port scanning attack, they are attacking the similar targets. The common features that are used to detect the port scanning attacks contain protocols, port numbers, IP addresses, and TCP flags [34]. The port scanning attacks can be employed on transport layer protocols including TCP, UDP, ICMP and IP. They can be performed against a single port from multiple IPs, and multiple ports can be scanned by one IP as well. Since TCP connections are established during the attack and different TCP flags are used for different types of port scanning attack, TCP flags become one of the special characteristics of the port scanning attack [35].

2.2 Feature Selection

2.2.1 Filter Method

The filter method of feature selection is to choose the highly related features from the entire feature pool in the dataset without the involvement of machine learning algorithms [36]. This method solely depends on the test scores of various statistical approaches for the dependence or correlation between different features. These different features may have linear or non-linear relationships. The common statistical approaches include Pearson correlation coefficients, Chi-square test, Mutual Information, ANOVA test, etc. The statistical approaches rank the features by the degree of correlation or joint distribution. The closer the degree of correlation between two

features is to 1, the more these two features are related; the smaller the joint distribution between two features is, the lower the relationship between these features would be. On the reasoning of independence from other complicated mining and validating methods, the filter method is relatively easy to implement and is able to rule out the irrelevant features efficiently.

Filter method of feature selection is used commonly in the data preprocessing stage, because the machine learning model has not been applied yet with the sample data of the selected features that are decided in the feature selection stage. This characteristic creates another important advantage of filter method, which makes the whole process of building a machine learning model much faster. The features were only selected once using filter method to create the subset of highly related features. Since the data dimension is reduced dramatically before the data is fed into the machine learning algorithm, it is less prone to over-fitting [37].

2.2.2 Wrapper Method

The wrapper feature selection method shows the significant difference from the filter method. The wrapper method evaluates the goodness of the features by considering the prediction results through the machine learning models [37]. The commonly used machine learning algorithms in wrapper method include SVM, DT, BN, k-means, RF, etc. The training process of machine learning model must be iterated for multiple times to evaluate the predictive results of each subset of selected features, for example accuracy rate and error rate.

The wrapper method usually tries out all the combinations of different features in order to find the subset with the highest accuracy rate with low error rate. However, the wrapper method may suffer from the over-fitting and become time-consuming. Also, the wrapper method has less transitivity because of the differences in the concept of various machine learning algorithms [36]. If the machine learning algorithm is changed after the learning process has been finished, the feature selection should be done again to make sure that the selected features are suitable with the new algorithm.

There are three major techniques in wrapper method of feature selection: forward feature selection, backward feature elimination, and Bi-directional elimination.

Forward feature selection initiates selecting process when there is no feature in the feature subset and a new feature that could best improve the prediction results of the machine learning model is added into the feature subset in each selecting iteration until the result could not be

improved anymore [38]. The features selected by this method represent the best subset of features that could achieve the highest accuracy rate during the classification.

Backward feature selection is completely opposite to the forward selection, which starts the selecting process with all of features in the dataset and remove one feature in every iteration that makes the largest decrease in the accuracy of the model. The reducing process would repeat until the accuracy could not be further improved or all of the feature have been exhausted [39].

Bi-directional elimination can be seemed as combining the forward selection with the backward elimination. This method first sets thresholds of significance level for the forward selection as well as the backward elimination. Then the forward selection is applied by adding one feature and examining the significance level of the feature in each selection round. After the forward selection has been finished, the backward elimination will be performed by removing the feature that has the higher significance level than the elimination threshold in each reducing round. These two methods will be repeated until the optimal feature subset is found [40].

2.2.3 Embedded Method

Embedded method of feature selection overcomes the disadvantages of both filter and wrapper methods and combines their merits by applying the machine leaning algorithms that have built-in feature selection and regularization function [41]. Unlike the wrapper method that the selection process is separated from the learning process and uses exhaustive search for high dimensional data, the embedded method allows to pick the features in the training process of machine learning algorithms and reduces the data dimension internally. Therefore, the embedded method is less likely to consume high computational power. Comparing with filter method, the embedded method reaches higher overall detection performance by interacting with the machine learning algorithms instead of solely depending on the feature rank. The common feature selection algorithms in embedded method are regression and tree-based algorithms, for example, LASSO regression is one of the regression algorisms, Decision Tree and Random Forest are tree-based algorithms, . [42]

The embedded method optimizes the objective functions with the regularization penalty terms and measures the feature importance [43]. L1 regularization used by LASSO regression penalizes on the weight of less important features to zero. The features that have the least coefficient will be eliminated automatically for achieving better detection results. The feature

importance is examined by the tree-based algorithms. The features are permuted based on the importance score, and the most important feature will keep close to the root of the tree.

2.3 Feature Construction

Dataset is a key component of machine learning and is used as the data source for training the machine learning algorithms. Dataset contains all the information that the machine learning may use to recognize and classify the patterns of the objective, for example the network activities for intrusion detection. Dataset is composed by various features, each represents a piece of data that indicates some information about the objective. Some of these features are directly extracted from the raw data, called basic features, for example the IPs, port numbers and TCP flags are originally contained in the network packet. There are also a lot of features in the dataset that are the statistical information created by analyzing the raw data [44]. This kind of feature is called derived features. These statistical features are manually created by the researchers to better describe the characteristics of the objective. Onut and Ghorbani divided the derived features into two major groups: single connection dependent and multiple connection dependent [45].

The single connection dependent derived features are created using the flow information from a single connection. The functionality of the single connection dependent features is to verify whether the current connection has malicious intent or not. The single connection dependent derived features could be used to detect the bursty and stealthy attacks based on the packet data within a certain time interval and the whole lifetime of a single connection. The examples of the single connection dependent derived feature include number of packets, packet length, number of TCP flags per packet, etc.

The multiple connection dependent derived features are used to represent the relationships between multiple connections. These features are mostly used to detect any kinds of network attack that was launched through multiple network connections, for example worm attacks, DDoS attacks, etc.

In the research of the machine learning in intrusion detection, some derived features are created to better detect various kinds of network attacks.

2.3.1 Brute-force Characteristic Features

Najafabadi et al. introduced three derived features that present the packet information extracted from the network flow following the IPFIX standard to provide better detection of the brute-force attacks: the number of packets, packet size, initial flags, and session flags [20].

- The number of packets describes how many packets are captured in the flow. Since the attackers need to frequently guess the password of the users' accounts until they obtain the correct one, the packet number of brute force attacks would be much larger than that of normal login activities.
- Packet size describes the total size of the packets in the flow. On the reasoning of the small size of network flow for the failed logins, the normal login activities would have apparently larger byte size.
- Flow flags describe the flags of the all packets seen in the flow. This feature can recognize the attack traffic that contains the complete set of TCP flags including FIN, SYN, PSH, and ACK flags, which is not normal for the common TCP connections that have only SYN and ACK flags in general.

Constructed features mentioned above were applied in 5- Nearest Neighbor, C4.5 Decision Trees, and Naïve Bayes algorithms to predict SSH brute force attacks by Najafabadi et al. The results showed that the constructed feature significantly improved the performance of weak algorithm like Naïve Bayes and achieved 99% accuracy using 5- Nearest Neighbor and C4.5 Decision Trees.

2.3.2 Botnet Characteristic Features

The botnet attack can be detected using the derived features extracted from the network traffic mentioned in the last section. Since the bots need to contact with command and control server to obtain the instruction for further malicious activities, the TCP connections are required between bots and command and control server. And this kind of TCP connections shows a periodic pattern according to the observations by Wang et al [46]. Under this situation, the interval time between request and response flow could be an important feature.

On the other hand, Wang et al. found that the TCP connections between bots and C&C server usually follow the periodical DNS query from bots to C&C server. Therefore, the

information extracted from the DNS query can be used as effective feature to detect the botnet attacks. Three derived features based on the DNS query were introduced: interval time of DNS queries, the total number of DNS responses and the failed DNS responses [46].

Jin et al constructed the similar features as mentioned above to predict the botnet attack using Adaboost, C4.5 Decision Trees, and Naïve Bayes algorithms. The result showed that using contracted features, Adaboost and C4.5 Decision Trees classifiers achieved over 90% for precision, recall, f1, and ROC area, and Naïve Bayes reached over 70% for all scores as well [46], which proved that the constructed features were critical for detecting botnet attack.

2.3.3 Web Attacks Characteristic Features

In order to pick up the web attacks more efficiently, eight derived features were introduced by Qin et al. using the information extracted from the web server logs: diffReqPercent, stutas200Percent, avgBytePerRequest, urlLevelRate, maxFrequency, FrequencyTimes, requestTimeDistrubution, and avgInterval [48]. Each of these features can be computed using the statistical information based on the web server records [49], including the total number of requests, different requests, and successful requests, and the total length of requests from the same user within a certain time interval. The detailed descriptions of eight derived features are listed as follows:

- diffReqPercent describes the percentage of requests from the same user for different services in the total requests. The normal users would not frequently access the same page within a short period. But the malicious programs from the attackers will repeatedly attack the same page.
- stutas200Percent describes the percentage of successful requests from the same user in the total requests. The normal users usually access the functioning web pages that rarely fail in connection. The malicious programs will try to randomly access any vulnerable web pages and fail sometimes, because some of web pages may be dysfunctional or forbidden for public access.
- avgBytePerRequest describes the average length of requests from the same user. The normal users will consistently transmit larger packets when browsing the web content. However, the attack packets may change in the packet size dramatically to consume the bandwidth and processing power of the web server.

- urlLevelRate describes how different the requested URLs are from the same user. The normal users may randomly access multiple URLs, but the malicious programs may target certain URLs in the stable or periodic sequence.
- maxFrequency describes the maximum number of requests from the same user. The attackers would send out the requests more frequently than the normal users.
- x FrequencyTimes describes the number of requests that is more than predefined threshold number X from the same user. The previous research conducted by Qin et al. indicated that the normal users usually send out one request every five minutes, but the request number of the attackers would reach once per minute minimum.
- requestTimeDistrubution describes the difference between time intervals of each pair of requests from the same user. The normal users may not have certain regularity in the request frequency, while the malicious programs may be set to launch the attacks with certain frequency.
- avgInterval describes the average time interval of requests from the same user. As mentioned above, the normal users may show the randomness for browsing web content, they could stay in some web pages that interest them for a long time. However, the attackers would try to access as many web pages as possible within a short period of time for maximizing the performance of the web attacks.

After applying the constructed features in Naïve Bayes, Radial Basis Function Network and C4.5 decision tree, Qin et al achieved accuracy and detection rate for more than 98% and false positive rate for lower than 2%.

2.3.4 Infiltration Attack Characteristic Features

The port scanning attacks take advantage the TCP, UDP, and ICMP responses to detect the accessible open ports of any hosts existing in the network, and these scanning attacks usually get involved in the flows either to various ports in the single host or the same port in the multiple hosts. However, the network event associated with these protocols can be also used to identify the port scanning attacks. Ring et al. created two derived features that target different kinds of port scanning attacks: ICMP-Error count and RST count [50].

- ICMP-Error count describes how many ICMP error messages are received. Under the attack of UDP port scanning, due to the nature of UDP protocol, the victims would either

accept the forwarding packets or return the ICMP error message. Counting the ICMP message may help to reveal the attack pattern of the UDP scanning.

- RST count describes the number of received RST flags from different targets. The RST response is usually sent to the attackers who launch various TCP scanning attacks, when the TCP ports of the normal host are closed.

Decision tree and Support Sector Machine were used to test the performance of constructed features. The results indicated that both classifiers reached 100% detection rate and 10% false alarm using the constructed features.

2.4 Summary

This chapter elaborated on the common network attacks that are threating the current computer networks. The important characteristics of each network attack were explained in terms of flow information. This section also listed out three feature selection methods in machine learning. The advantages and disadvantages of each method were discussed as well. In addition, the derived features for targeting various network attacks were introduced and the experiment results were provided to prove the effectiveness of the derived features.

CHAPTER 3. RESEARCH METHODOLOGY

This chapter introduces the method that was used to conduct the research in the project, including the workflow of experiments, the method of collecting data, and evaluation parameters.

3.1 Introduction

In order to mitigate or even avoid the damage brought by the dramatically increasing cyber threats, the intrusion detection system is widely deployed to improve the security strength of the network for either personal use or enterprise level. Recently, the machine learning has become more and more popular and well developed, the combination of machine learning and intrusion detection becomes a promising solution for making the network more robust. When the researchers are trying different machine learning algorithms, some of them neglect the importance of preparing the good quality of the data that fits the specific machine learning model they use or the target issue they want to solve. The problem that lack of an effective method to select the most effective features is a critical barrier that prevents the machine learning model to reach its best predictive capability [51]. On the reasoning of the existence of irrelevant data and excessive amount of data, the machine learning model may suffer from the overfitting and endless learning process. The research question of this study is that what are the features that could significantly bring up the detection performance for each kind of network attacks? The goal of the project is designing a feature selection approach that combines the advanced feature selection algorithms and the characteristic features for different kinds of network attacks. By the end of this study, the optimal feature subset that best improved the performance of the machine learning model in network intrusion detection was provided, and the analysis about the result of machine learning model using proposed solution was explained in detail.

3.2 Hypotheses

The Hypotheses for this study were as follows:

- H_0: The proposed feature selection approach of combining feature selection algorithms and attack characteristic features does not improve the detection performance when compared with detection approach using all features in the given dataset.

- H_a: The proposed feature selection approach of combining feature selection algorithms and attack characteristic features improves the detection performance when compared with detection approach using all features in the given dataset.

3.3 Research Approach

The research focused on the study of feature construction and feature selection in machine learning for detecting network attacks. The research required the understanding of the machine learning process and programming technique and was mainly based on the experiment for analyzing the effectiveness of the proposed solution. The research was conducted in a series of steps as listed below.

1. Understand the functionality and process of machine learning.
2. Study and analyze the current approaches of feature selection for network intrusion detection.
3. Set up machine learning model.
4. Design a new feature selection approach and apply the proposed approach to obtain optimal feature subset.
5. Generate findings of the tests, analyze and document the performance of the proposed solution.

3.4 Experiment Design

The following section elaborated the workflow of building the machine learning model that was used as a testbed for the proposed solution. The machine learning model contained three major stages: preprocessing data, training model, and classifying target data. The detailed activities in each stage were depicted in Figure 3.1.

1. Preprocessing data aimed to organize the raw training data in an acceptable data structure and converted the dataset into the proper format permitted by the machine learning models [52]. Data preprocessing dealed with the tasks including handling null values, categorical variables, standardization, one-hot encoding, and multicollinearity [53].

2. Training machine learning model allowed it to fit with the training data and tune model parameters for classification need. This stage repeatedly adjusted the hyperparameters of machine learning algorithm to find function that best described the data [54].
3. Classifying target data was to apply the trained machine learning model to perform detection functionality on the test data that had never been fed into the model before.

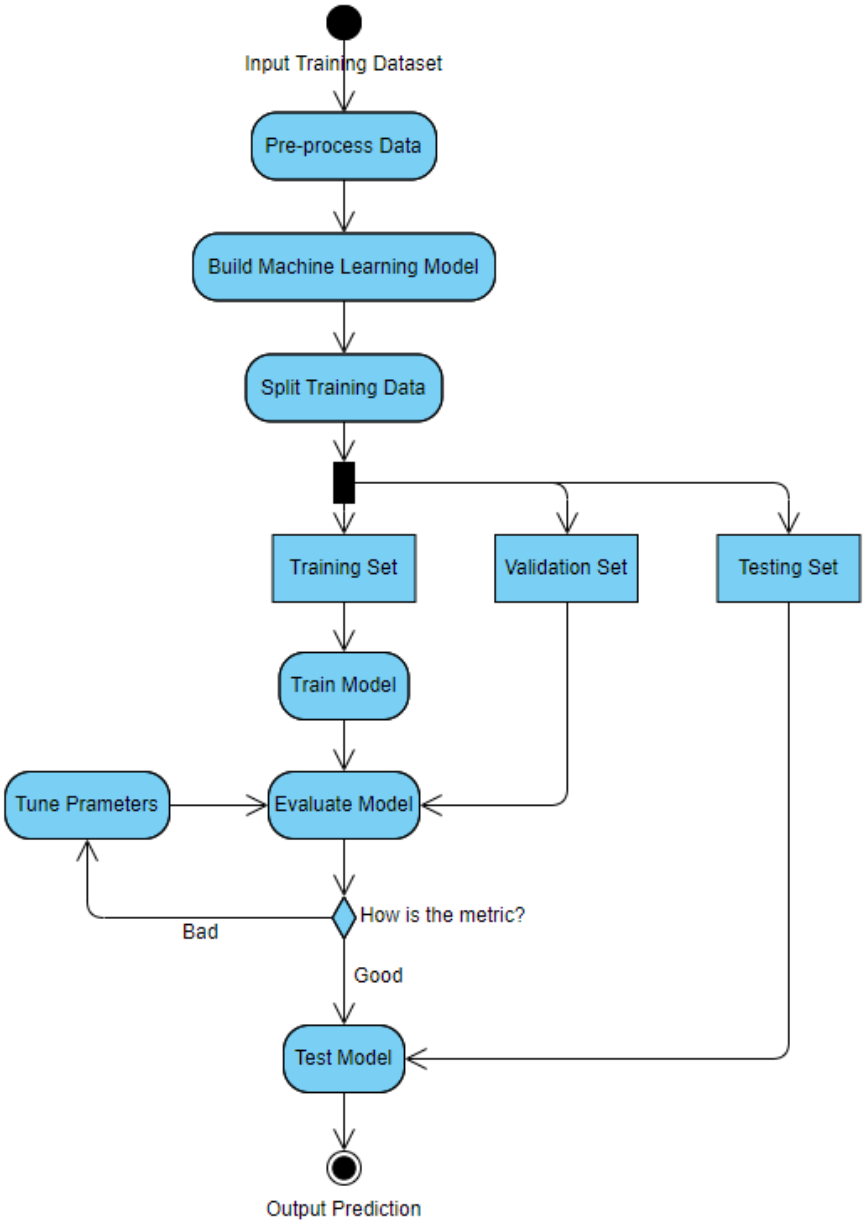


Figure 3.1 Machine Learning Workflow

The proposed approach of feature selection was divided into six steps shown in Figure 3.2. Firstly, the CSE-CIC-IDS2018 dataset with raw features was input. Secondly, attack characteristic features that might indicate the attack patterns were added into the dataset. Thirdly, all data was preprocessed to make sure all features were processible by machine learning model. Fourthly, all features were scaled to make the data normalized in the magnitude. Fifthly, all features were calculated through multiple feature selection algorithms for filtering out the majority of irrelevant features. The final step was to output the feature subset for training the machine learning model and producing the prediction result.

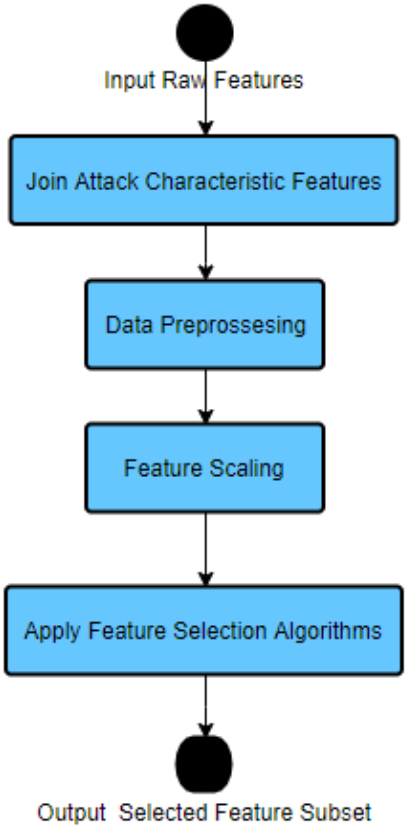


Figure 3.2 Proposed Solution Process

Four experiments were conducted to create the comparison on the prediction results between machine learning model using raw features, constructed features, and two feature selection techniques respectively. The result of the machine learning model that used raw features was used as a baseline.

3.5 Data Collection

The dataset that was used to conduct the experiments in the project was the CSE-CIC-IDS2018 dataset, a comprehensive dataset for network intrusion detection collected by Canadian Institute for Cybersecurity (CIC) and Communications Security Establishment (CSE). The dataset fulfilled the key requirements of dataset used for supervised machine learning, for example the complete labels for attack and normal traffic.

3.5.1 Variables

The CSE-CIC-IDS2018 dataset contained benign background traffic and malicious traffic based on seven kinds of network attacks, including brute-force attack, Heartbleed attack, botnet attack, DoS attack, DDoS attack, web attacks, and infiltration attack. The attacks studied in the project were brute-force attack, botnet attack, web attacks, and infiltration attack. The dataset included seven features extracted from the raw data flow, for example protocol, timestamp, IP address, etc. [44].

3.5.2 Instrumentation and Methods

According to the dataset description given by Canadian Institute for Cybersecurity, the data of CSE-CIC-IDS2018 dataset was generated using the CIC-BeignGenerator and simulation network on the AWS computing platform shown in Figure 3.3. The attacking infrastructure includes 50 PCs using either Ubuntu or Windows Operating System, and the victim organization has five departments with five subnets as well as a server room. The machines of victim organization included 420 PCs and 30 servers. The computers of all departments except IT department used Windows Operating Systems, for example Windows 8.1 and 10 Operating Systems; The computers of IT department used Ubuntu Operating System; The servers used Windows Server 2012 and 2016 Operating Systems [55].

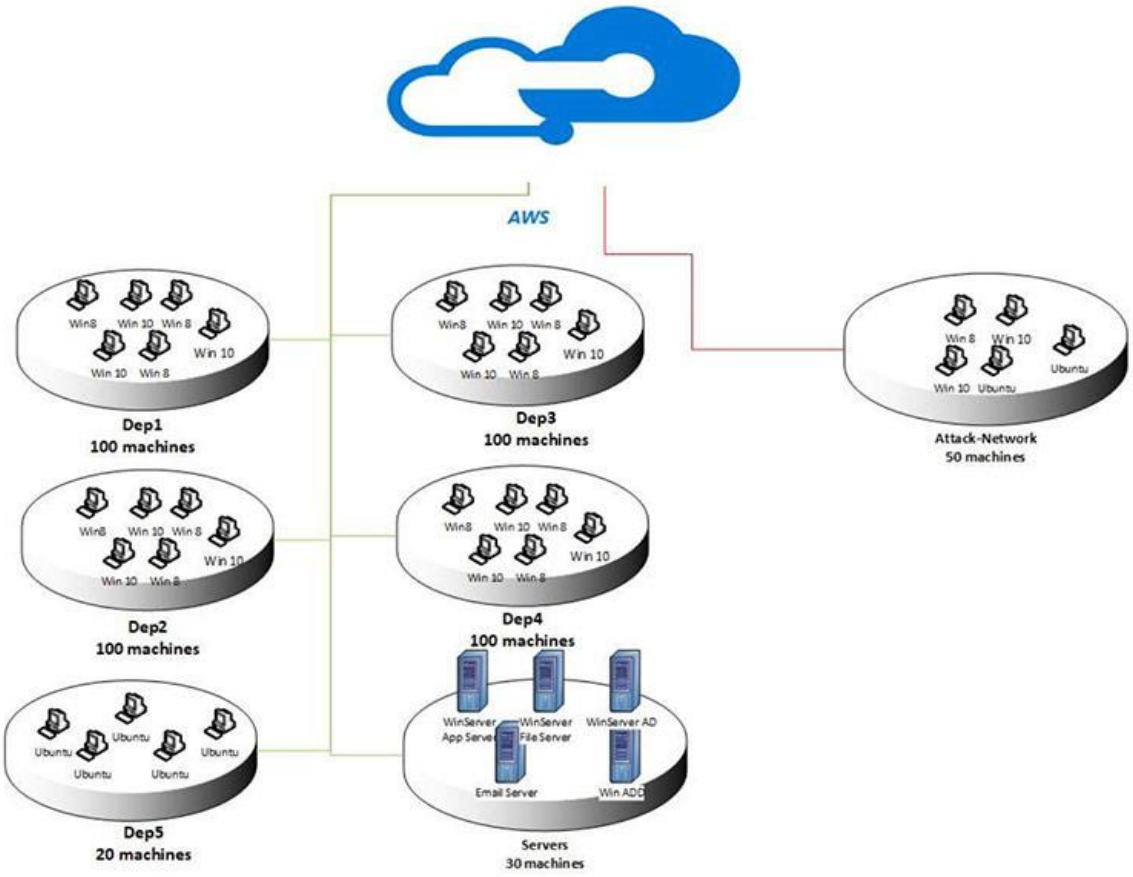


Figure 3.3 Simulation network on AWS Computing Platform

Source: Adapted from [55]

The various attacks were launched targeting the machines of victim organization, and the attacking process started from 2/14/2018 to 3/2/2018. The detailed attack records included attacker IP, victim IP, attack name, attack date, start time and finish time.

By analyzing the current available datasets for intrusion detection, the CSE-CIC-IDS2018 dataset shows its high reliability compared with other similar datasets and fulfills eleven key criteria for evaluating the published dataset for intrusion detection system stated by Sharafaldin et al, including “attack diversity, anonymity, available protocols, complete capture, complete interaction, complete network configuration, complete traffic, feature set, heterogeneity, labeled dataset, and metadata [56].”

3.6 Analysis Methods

The project evaluated the prediction results of different experiments using the test scores produced from confusion matrix. Confusion matrix is a two-dimensional matrix that represents the correlation of true conditions and predictive results showing in Table 3.1.

Table 3.1 Confusion Matrix

	Positive Prediction	Negative Prediction
Positive Condition	True Positive (TP)	False Negative (FN)
Negative Condition	False Positive (FP)	True Negative (TN)

TP describes the number of abnormal samples being accurately classified. TN defines the number of normal samples being accurately classified. FP specifies the number of normal samples being falsely classified as abnormal samples. FN specifies the number of abnormal samples being falsely classified as normal samples.

Various test scores were calculated using confusion matrix in the project: Accuracy, Precision, Recall, and F1 Score.

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

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

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

$$\text{F1} = \frac{2 * Recall * Precision}{Recall + Precision}$$

CHAPTER 4. EXPERIMENTS

This chapter provides the experiment implementation by elaborating the instruments and technologies used in the project, the architecture of selected machine learning model, and the code structure to achieve the test environment and proposed solution.

4.1 Instruments and Technologies

In order to build the experiment environment, various instruments, software and applications were employed and eased the implementation.

The experimental environment was implemented on JupyterLab hosting on a Jupyter docker container that contained various Jupyter applications and interactive computing tools. For the physical device, all experiments were conducted on the server having 2.93 GHz 6 cores Intel Xeon X5670 CPU with 96 GB RAM.

Since Python programming language showed its advantages of easy-to-use and fast-to develop, lots of researchers and learners have widely used Python in machine learning. Under this situation, the project decided to use Python to compile the code for the machine learning model. Python extensively supported superior-quality data science libraries to reduce the length of code and provide more powerful functions. In the project, a large amount of multidimensional data required to be processed. Therefore, Numpy and Pandas libraries were used to manipulate the multidimensional data metrices and arrays. Comparing with Python core library, Numpy library provided better performance in terms of data structure. Numpy data structure consumed less memory by saving the space for element object for each element in the data structure [56]. Faster speed is another significant advantage of Numpy. The faster speed of Numpy is due to multiple reasons. Firstly, Numpy is written in C programming language that has less execution time. Secondly, Numpy array contains a collection of homogeneous datatypes which are stored in contagious memory locations. Thirdly, NumPy is designed to breakdown a task into multiple smaller fragments, and then processes all the fragments parallelly [56]. Pandas is a powerful data analysis and manipulation library that improves the structure and operation for data analysis in Python Programming Language. Pandas library extracted the data from the .CSV files and converted the data into data-frame structure that was required for processing in the machine

learning model. Pandas offered various useful functions to explore and process the dataset, including calculating the statistics about the data, visualizing the data, and cleaning the data [59]. The project employed scikit-learn library, which contains many efficient helper tools for building machine learning models to simplify the construction of machine learning models. The helper tools of scikit-learn library could be applied in various tasks including data preprocessing, model selection, classification, etc. [60]. In the project, scikit-learn provided functions to split train set and test set, normalize the data, select relevant features, build classifiers, evaluation matrix and reports.

4.2 Model Architecture and Specification

As depicted in the chapter 3, the experiment depended on the machine learning model that was used as a testbed for the proposed solution. In the project, the selected machine learning classifier was decision tree that continuously splits the data according to the defined parameters. The structure of decision tree is presented like a tree, showing in the Figure 4.1.

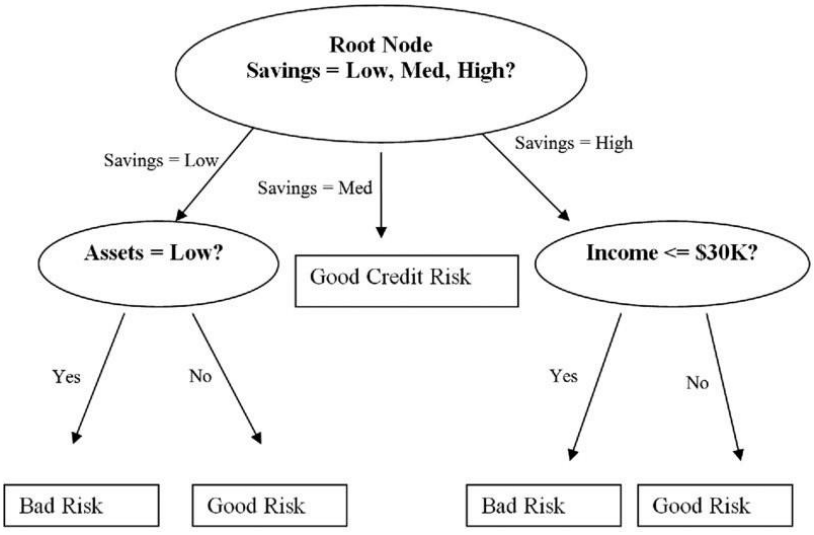


Figure 4.1 Architecture of Decision Tree

Source: Adapted from [61]

Decision tree contains three major components: Nodes, Branch, and Leaves. Node represents a test for the data of a certain feature. Branch contains the result of a node and connects to the next

node or leaf. Leaf is the final node of a tree that provides the prediction corresponding to the label of the sample data [61].

Decision tree algorithm was specified with some parameters that helped to optimize the performance. Decision tree employed the Gini impurity to evaluate the quality of data splitting. Gini impurity calculated the probability of a sample being randomly misclassified regarding to the distribution of different labels. The algorithm also controlled the data selection to use different random value for each run of the classification by setting random_state to none. Decision tree classifier was set to use best splitter that split the data on the most relevant feature instead of randomly shuffling the feature [62]. Other hyperparameters of the selected architecture is shown in Table 4.1.

Table 4.1 Sample Hyperparameters of Decision Tree Algorithm

Hyperparameter	**Value**
Criterion	gini
Random_state	None
Splitter	Best
Class_weight	None
Max_depth	None
Max_feature	None
Max_leaf_nodes	None
Min_impurity_decrease	0.0
Min_impurity_split	None
Min_samples_leaf	1
Min_samples_split	2
Min_weight_fraction_leaf	0.0
Presort	False

4.3 Code Structure

As mentioned in chapter 3, the project involved three major stages: Preprocessing data, Training model, and Classifying target data. However, in order to present the technical details of model construction, when writing the code using Python programming language, the model was expended to six phases: Data Loading & Presentation, Data preprocessing, Feature Scaling, Feature Selection, Building the model, and Prediction & Evaluation. Table 4.2 in the next page summarizes functions of all stages. The modeling stages were explained with detailed methods and the code listed in APPENDIX A.

Table 4.2 Modeling Stages and Descriptions

Stage	Description
Data Loading & Presentation	This stage was to import the CSE-CIC-IDS2018 dataset and provide the sample view and statistical summary of the dataset.
Data Preprocessing	This stage was to laundry the CSE-CIC-IDS2018 dataset.
Feature Scaling	This stage was to split train & test set and normalize the data of the CSE-CIC-IDS2018 dataset.
Feature Selection	This stage was to use the feature selection algorithms to produce a subset of features.
Building the model	This stage was to build the decision tree classifier.
Prediction & Evaluation	This stage was to predict the network attacks and evaluate the results of experiments.

4.3.1 Data Loading

The Data Loading & Presentation stage was to import the proper Python libraries and the CSE-CIC-IDS2018 dataset and provided the sample view and statistical summary of the dataset.

The CSE-CIC-IDS2018 dataset was originally separated into 10 csv files based on different types of network attack, including FTP-Brute Force, SSH-Brute Force, Brute Force -Web, Brute Force -XSS, SQL Injection, Infiltration, Bot, DoS-Golden Eye, DoS-Slowloris, DoS-Slow HTTP Test, DoS-Hulk, DDoS attacks-LOIC-HTTP, DDoS-LOIC-UDP, and DDOS attack-HOIC. Due to the scope of the project, network attacks related to Dos and DDOS weren't considered and used in the project. Therefore, the first step of Data Loading & Presentation stage was to remove the .csv files containing data of Dos and DDOS attack from the dataset. All other csv files were required to import as a data-frame, a two-dimensional data structure containing labeled axes. Data-frame of the dataset was essential in order to convert the .csv file into Numpy array format containing all numerical values of dataset for further training the machine learning model. Pandas library provided functions to concatenate and import multiple .csv files into one data-frame, showing in the lines of code below.

In code I, glob module and Pandas library were imported first. The glob module found all separate .csv files of CSE-CIC-IDS2018 dataset whose names started with `Traffic` and made a list of filenames of .csv file called `files`. Then within a for loop of the list of `files`, `pd.read_csv` function was used to read the each .csv file and imported the data of .csv file as a data-frame called `df`. Each new created data-frame was added to a list called `L` using `append` function. After creating data-frames for all .csv files, `pd.concat` function was applied to concatenate all Pandas data-frames along index axis.

4.3.2 Data Preprocessing

This stage was to laundry the CSE-CIC-IDS2018 Dataset and create train & test set. Before feeding the data to the machine learning model, the dataset was preprocessed to make sure that all values were fit into the proper data type and processable without any errors. The following steps were adopted to laundry the CSE-CIC-IDS2018 Dataset.

- Delete obviously insignificant features;
- Convert data into standard datatype;
- Remove rows containing “Infinity” and “NaN” value;
- Reduce the long decimal digits of the float numbers;
- Rename attack labels;

As shown in the sample view of The CSE-CIC-IDS2018 dataset in Figure 4.2, the original dataset contained feature that didn’t have significant influence on verifying whether the traffic contains the network attack or not, for example Timestamp. Since Timestamp was a list of time and date that each network traffic occurred, there is not much help to train the machine learning model. In code II, Timestamp column was removed from the data-frame by executing `df.drop` function.

After exploring the data type for all columns in the dataset, the output showed that there were mixed datatypes within most columns of the dataset, which might cause problems for the further data normalization. In order to make sure the consistency of datatype for each column, the data was converted into standard datatype except Label column. `pd.to_numeric` function was applied to convert the value of columns into float64 datatype. And invalid parsing triggered the function to automatically set error values to NaN [63]. By scanning the dataset, it was easy to find

that there were some NaN and Infinity values that couldn't be processed by data scalers. Therefore, all rows of the dataset that contained NaN and Infinity values were deleted from the dataset. The function that were used to remove rows containing NaN was `df.dropna`. All rows containing Infinity were excluded from the data-frame by applying a condition and reconstructing the same data-frame, for example `df[(df != 'Infinity')]`. Dropping rows caused discontinuous index, so that resetting index was required with `df.reset_index` function. Since some float numbers had extremely long digits after decimal, in order to avoid the memory issue with these long digits during the computing process, the number of digits after decimal was reduced to 1 for all float numbers. As written in the code II, the number value of whole data-frame was rounded to one digit after decimal using `df.round(1)` function.

Label column contained label values for all instances. Labels were divided into eight categories: seven attack labels, mentioned in Data Loading section, and a benign label. However, string value couldn't be processed by machine learning model. Under this situation, all seven label values were renamed with numeric values from 0 to 7.

In code III, Label column was taken up from the data-frame and replaced the label values with numbers respectively using `labeldf.replace()` function, then new Label column was put back to the data-frame.

4.3.3 Feature Scaling

This stage was to split train set & test set and normalize the data of the CSE-CIC-IDS2018 dataset. After finishing the data laundry for the CSE-CIC-IDS2018 dataset, the next step was to randomly create train set and test set by separating the dataset with certain percentage. As discussed in the experimental design section of chapter 3, train set was used to help the machine learning algorithms fit the parameters that best described the sample data. Test set was a set of data that was never used in train set and was employed to produce the prediction and evaluate the final machine learning model. In the project, the train set and test set were 80% and 20% of the processed CSE-CIC-IDS2018 dataset respectively.

In code IV, the data-frame was separated into x and y. x was assigned as a data-frame of all features and y was a series of labels in the original data-frame. Then x and y were split into train set and test set respectively using `train_test_split()` function. The parameter

`test_size` defined the proportion of the data-frame and series that was assigned to test set, and the rest of data-frame and series became the train set. The parameter `random_state` was set to 42 to freeze the randomness of sampling of train set and test set [64]. So that every time the dataset was split, the values of train & test set were always the same. Since train & test set for x data-frame were still containing Label column that was duplicated in y series, Label column was dropped from the train & test set for x data-frame using commands `xTrain[:,:-1]` and `xTest[:,:-1]`.

The CSE-CIC-IDS2018 dataset contained features that highly vary in magnitudes, units, and range, for example, some features counted the number of data packets, some counted the seconds of data flow, some had huge numbers, some had negative numbers. However, some of machine learning models were highly sensitive to the feature scaling due to the optimization techniques and calculating mechanisms [65]. Therefore, the feature scaling played significant role to keep the machine learning model training properly. In the project, `MinMaxScaler` was used to normalize the train & test set.

In code V, preprocessing module was imported from sklearn library. Preprocessing module provided standardization, normalization, transformation functions that converted dataset into suitable representation for machine learning models. `Preprocessing.MinMaxScaler()`, one of the data scalers provided by preprocessing module, translated values of each feature of train set and test set into the range between 0 and 1.

4.3.4 Feature Selection

This stage describes the method of using feature selection techniques to obtain highly relevant features. The project applied two feature selection techniques, ANOVA F-test and Recursive Feature Elimination.

In code VI, ANOVA F-test and Recursive Feature Elimination were imported with `f_classif` and `RFE` functions from sklearn library. Floating-point error was handled by ignoring the division by zero and invalid floating-point operation. `SelectPercentile` function, called `selector`, passed the `f_classif` function and defined the highest scoring percentage of features to 10%. The returned values of `selector`, `x_f_train`, were the numbers of selected instances and features from the train set. Since Recursive Feature Elimination required a machine learning

classifier to evaluate the feature importance, the decision tree classifier was created first and named with `clf`. `n_features_to_select` parameter, the number of features selected from the train set, was set to 5, which corresponded to the number of features that could reached best accuracy based on feature ranking from RFECV function. `x_rfe_train` variable was the returned values of selected instances and features.

4.3.5 Building the model

This stage was to build the decision tree classifier. Machine learning model was the key component of the project. Thanks to sklearn library, the decision tree classifier was directly called from the sklearn.tree module without complex and tedious coding process. In code VII, The function calling decision tree classifier was `DecisionTreeClassifier()` and named `clf_all`. The detailed parameters of the decision tree model were discussed in the model architecture and specification section. `Fit()` function fed the train set and corresponding labels into the machine learning model for training.

4.3.6 Prediction & Evaluation

This stage was to predict the network attacks using test data and evaluate the machine learning model along with the proposed feature selection method. Since the CSE-CIC-IDS2018 dataset was divided into train set and test set, and the machine learning model was built and trained using train set, the model was ready to produce the final prediction based on the test set.

In code VIII, `predict()` function provided by sklearn library was passed to trained machine learning model, `clf_all`, and predicted the label value for all samples in the test set, `xTest`. The predicted output returned by `predict()` function was stored into the new variable called `Y_all_pred`.

In code VIIII, various test scores were calculated with cross validation strategy. `make_scorer` module made scorers based on the performance metric. Multiple scoring functions, including `accuracy`, `precision_score`, `recall_score`, and `f1_score`, were imported to compute corresponding scores. Cross validation was employed to avoid the potential impact of small samples for some targets, for example, there were only 8 instances of SQL Injection in the

test set. `cross_validate` function split the dataset into four smaller sets with same label distribution. Among four equal-sized sets, three sets were used to train the model and the remaining one was used to calculate the performance metrics. A customized function, `average_score_on_cross_val_classification`, was defined to evaluate the machine learning model using `cross_validate` function and return the absolute mean value for all four scores.

CHAPTER 5. RESULTS

This chapter introduces the results of the study. Dataset presentation section was provided to describe the dimension of the CSE-CIC-IDS2018 dataset and sample view of the dataset. Feature construction section introduced the features that were constructed using the data provided in the original dataset. Feature selection section described the feature selection methods used in the project and listed the selected features. Model evaluation explained the performance metrics based on the different feature selection methods.

5.1 Feature Construction

Feature construction was completed by using CICFlowMeter, a feature extractor that extracted information from the bidirectional flows. CICFlowMeter provided functions to create time-related features from Pcap files of both forward and backward flows. In the original Pcap files, there were seven raw features that presented the sequence of the data flows and all sort of packet information. Seven raw features included FlowID, Timestamp, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol. However, SourceIP and DestinationIP were removed due to the possibility of data leak during the model training and the difficulty of feature encoding. Using the feature construction functions of CICFlowMeter, extra 76 features were created targeting the attack patterns of different kinds of network attacks based on the packet number, packet length, time interval between packets, TCP flags, header length, segment size, bulk rate, initial window, active time, and idle time in the forward and reversed flows. Some samples of derived features and descriptions were shown in Table 5.1.

Table 5.1 Samples of Derived Features

Feature Name	Description
Flow Byte/s	Number of bytes per second in the flow
Flow Pkts/s	Number of packets per second in the flow
Flow IT Mean	Mean time between packets in the flow
Flow IT Max	Maximum time between packets in the flow
Flow IT Min	Minimum time between packets in the flow
Fwd IT Min	Minimum time between packets in the forward flow
Fwd IT Max	Maximum time between packets in the forward flow
Fwd IT Mean	Mean time between packets in the forward flow
Fwd IT Total	Total time between packets in the forward flow
Bwd IT Min	Minimum time between packets in the reversed flow
Bwd IT Max	Maximum time between packets in the reversed flow
Bwd IT Mean	Mean time between packets in the reversed flow
Bwd IT Total	Total time between packets in the reversed flow
FIN Pkts	Number of FIN packets in the flow
SYN Pkts	Number of SYN packets in the flow
RST Pkts	Number of RST packets in the flow
PSH Pkts	Number of PSH packets in the flow
ACK Pkts	Number of ACK packets in the flow
URG Pkts	Number of URG packets in the flow
CWR Pkts	Number of CWR packets in the flow
ECE Pkts	Number of ECE packets in the flow
Fwd Sgmt Size Avg	Average PDU segment size in the forward flow
Bwd Sgmt Size Avg	Average PDU segment size in the reversed flow
Fwd Byte Bk Avg	Average bytes bulk in the forward flow
Fwd Pkt Bk Avg	Average packets bulk in the forward flow
Bwd Byte Bk Avg	Average bytes bulk in the reversed flow
Bwd Pkt Bk Avg	Average packets bulk in the reversed flow

5.2 Dataset Presentation

After loading the CSE-CIC-IDS2018 dataset, Executing `df.shape` function presented the dimension of the imported dataset. The output showed that the created data-frame contained 5138535 instances and 80 columns, which included 79 features and one label column. Figure 5.2 below is a snapshot of the first 20 rows of CSE-CIC-IDS2018 dataset with the structure of the data-frame format. The statistical summary of CSE-CIC-IDS2018 dataset included the count of values, unique values, the top value and frequency of occurrence in each column, showed in Figure 5.3.

	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	443	6	02/03/2018 08:47:38	141385	9	7	553	3773	202	0	...	20	0	0	0	0	0	0	0	0	Benign
1	49684	6	02/03/2018 08:47:38	281	2	1	38	0	38	0	...	20	0	0	0	0	0	0	0	0	Benign
2	443	6	02/03/2018 08:47:40	279824	11	15	1086	10527	385	0	...	20	0	0	0	0	0	0	0	0	Benign
3	443	6	02/03/2018 08:47:40	132	2	0	0	0	0	0	...	20	0	0	0	0	0	0	0	0	Benign
4	443	6	02/03/2018 08:47:41	274016	9	13	1285	6141	517	0	...	20	0	0	0	0	0	0	0	0	Benign
5	443	6	02/03/2018 08:47:41	250	2	0	0	0	0	0	...	20	0	0	0	0	0	0	0	0	Benign
6	80	6	02/03/2018 08:47:41	5964033	3	1	0	0	0	0	...	20	0	0	0	0	0	0	0	0	Benign
7	49690	6	02/03/2018 08:47:46	144	2	0	0	0	0	0	...	20	0	0	0	0	0	0	0	0	Benign
8	443	6	02/03/2018 08:48:17	90828	8	8	1748	3898	1078	0	...	20	0	0	0	0	0	0	0	0	Benign
9	443	6	02/03/2018 08:48:17	152	3	0	31	0	31	0	...	20	0	0	0	0	0	0	0	0	Benign
10	445	6	02/03/2018 08:48:35	234228	3	1	0	0	0	0	...	20	0	0	0	0	0	0	0	0	Benign
11	443	6	02/03/2018 08:47:40	60860062	15	13	870	3306	535	0	...	20	130202	148832	434003	69408	1e+07	16330.7	1e+07	9.96839e+06	Benign
12	49687	6	02/03/2018 08:48:41	68899	2	0	0	0	0	0	...	20	0	0	0	0	0	0	0	0	Benign
13	0	0	02/03/2018 08:47:31	119250445	72	0	0	0	0	0	...	0	4.64e+07	2.09e+07	6.11e+07	3.16e+07	1.33e+07	9.72057e+06	2.01e+07	6.38138e+06	Benign
14	443	6	02/03/2018 08:47:38	116185749	21	18	749	4567	389	0	...	20	467602	890267	329454	20847	9.63538e+06	1.27947e+06	1e+07	5.57287e+06	Benign
15	443	6	02/03/2018 08:48:36	60602600	13	14	338	3264	207	0	...	20	106008	92771	295376	68039	9.9944e+06	23887	1e+07	9.95915e+06	Benign
16	80	6	02/03/2018 08:47:40	116690356	17	16	883	1576	436	0	...	20	226997	26978	88621	11223	9.70055e+06	1.01454e+06	1e+07	6.47979e+06	Benign
17	80	6	02/03/2018 08:47:38	119731397	17	16	597	768	293	0	...	20	22859	26715.6	82791	11324	9.9538e+06	138400	1e+07	9.52041e+06	Benign
18	443	6	02/03/2018 08:47:38	118281864	36	83	1022	108156	250	0	...	20	134521	79961	191062	77980	5.9e+07	44453	5.9e+07	5.89e+07	Benign
19	443	6	02/03/2018 08:50:29	192	3	0	77	0	46	0	...	20	0	0	0	0	0	0	0	0	Benign

Figure 5.1 Sample View of CIC-CSE-2018 Dataset

	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
count	5138535	5138534	5138534	5138534	5138534	5138534	5138534	5138534	5138534	5138534	...	5138534	5138534.0	5138534.0	5138534.0	5138534.0	5138534.0	5138534.0	5138534.0	5138534.0	5138534
unique	79416	7	193607	1918239	2160	2869	16418	58640	2817	473	...	20	429159.0	353228.0	371603.0	179227.0	530832.0	369956.0	381964.0	454574.0	9
top	53	6	28/02/2018 01:48:05	1	1	1	0	0	0	0	...	20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign
freq	1189731	3504598	3055	101459	1488004	1667533	1573463	1865189	1573463	3556764	...	3051994	4361320.0	4476642.0	4361320.0	4361320.0	4123010.0	4424884.0	4123010.0	4123010.0	4308469

Figure 5.2 Statistical Summary

In data preprocessing and scaling process, timestamp information was removed from the dataset due to the insignificance for model training; all remaining data was converted to float datatype; all instances that contained Infinity and NaN values were dropped from the dataset; long decimal digits were reduced to one decimal digit for saving training time and memory ; the attack labels were renamed with numbers as shown in Table 5.2.

Table 5.2 Label Names and Corresponding Numbers

Label	Number
Benign	0
Bot	1
FTP-BruteForce	2
SSH-Bruteforce	3
Infiltration	4
Brute Force -Web	5
Brute Force -XSS	6
SQL Injection	7

After all works above, the CSE-CIC-IDS2018 dataset became clean and normalized in terms of datatype, magnitude, and integrity. Then train set and test set were created using the dataset. The train set was created with 1,347,733 instances and 78 features, and the test set had 337,263 instances with 78 features. The label distribution in train set and test set was showed in Figure 5.4. Showing in Figure 5.4, there was huge amount of benign traffic in the CSE-CIC-IDS2018 dataset. Compared with benign traffic, the number of attack traffic was quite small. Lack of balance between different kinds of network attack and benign traffic became the biggest weakness of CSE-CIC-IDS2019 dataset.

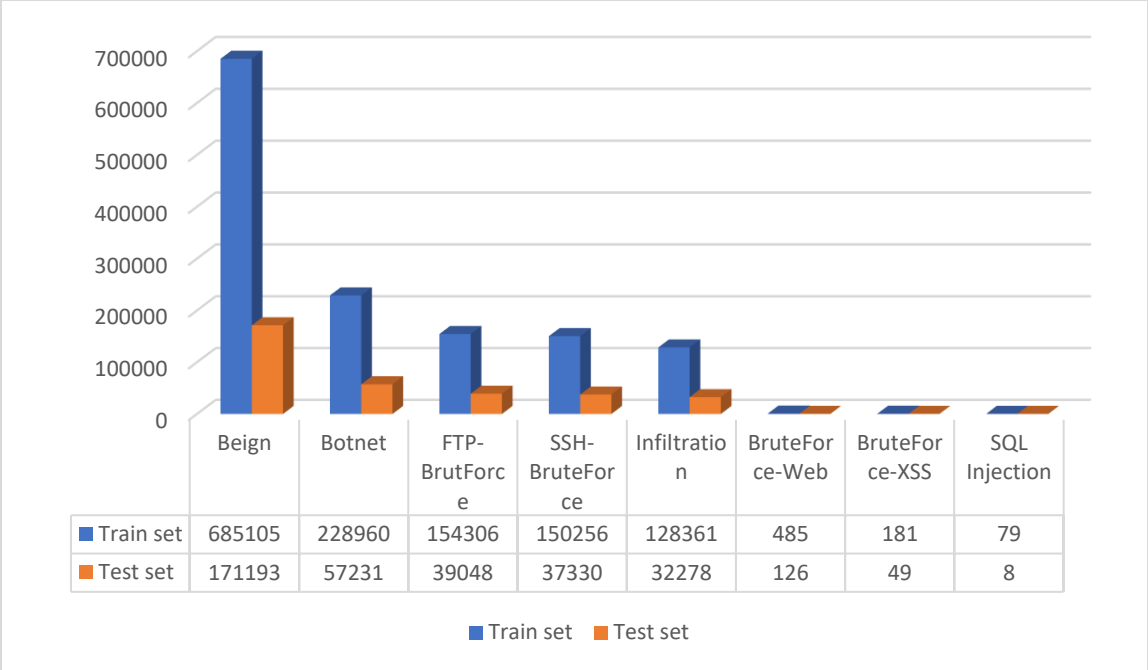


Figure 5.3 Label Distributionn of Train Set and Test Set

5.3 Feature Selection

The project used two different feature selection methods, ANOVA F-test and Recursive Feature Elimination. ANOVA F-test was a filter selection method that solely depended on the test scores to evaluate whether each feature had relationship with the label. Recursive Feature Elimination was a wrapper selection method that used machine learning algorithm, for example decision tree classifier in this case, to remove the features based on the contribution of each feature to the prediction results.

ANOVA F-test feature selection tested if each feature had any impact on classifying the attack categories. Based on the statistical technique of one-way ANOVA, the p-value was calculated to verify the probability of an attack category being classified using the values of each feature exclusively. The bigger the p-value was, the higher the correlation coefficient between this feature and the specific attack category was. All features were ranked with descending order based on the p-values. As the project decided to select the top 10% of the ranked features, the first eight features were selected as the feature subset. Table 5.3 showed eight features selected by ANOVA F-test. The eight features selected by ANOVA F-test were all constructed features, which proved

that the constructed features had better statistical relationship with the attack categories compared with raw features.

Table 5.3 Selected Features by ANOV F-test

Label Index	**Feature Name**
16	Flow Pkts/s
37	Fwd Pkts/s
38	Bwd Pkts/s
46	RST Pkts
49	URG Pkts
51	ECE Pkts
66	Init Fwd Win Byts
69	Fwd Sgmt Size Min

Recursive Feature Elimination created the feature subset by evaluating the feature importance through the machine learning estimator. Recursive Feature Elimination required to train the decision tree estimator with all features first for calculating the importance of each feature. Due to the sample splitting mechanism, decision tree algorithm introduced built-in function, for example Gini Impurity, for calculating the feature importance in terms of the misclassification rate. The feature with lower Gini Impurity was preferred and significant because the misclassification rate of this feature was lower. Then, the feature with the lowest importance was removed from the current feature set. The training and eliminating process was iterated until the required number of features was reached, which was five in the project. The top five features selected by Recursive Feature Elimination were listed in Table 5.4. The five features selected by Recursive Feature Elimination showed the low misclassification rate during the training process of the decision tree model.

Table 5.4 Selected Features by Recursive Feature Elimination

Label Index	Feature Name
0	Dst Port
26	Bwd IT Tot
28	Bwd IT Std
38	Bwd Pkts/s
69	Fwd Sgmt Size Min

5.4 Model Evaluation

As mentioned in Prediction & Evaluation section in chapter 4, the project calculated the test scores based on confusion matrix with cross-validation strategy for different experiments. Precision score described the correctness of positive samples being classified by machine learning model. Recall score described the ability of classifying all positive samples. F1 score was weighted average of the precision and recall, which considered both precision and recall score equally significant.

5.4.1 Classification Reports of Four Experiments

This section presented the classification report of machine learning model in four comparison experiments. Classification report presented the precision, recall, and f1 score for each kind of network attack and benign traffic individually on the top, and the macro average and weight average of test scores as well as overall accuracy on the button. There were 8 numbers from 0 to 7 on the top left of the report. Number 0 was the benign traffic and number 1 to 7 represented seven kinds of network attacks respectively.

Table 5.5 showed the performance metrics of decision tree model that used 7 raw features to classify each kind of network attack and benign traffic with cross-validation strategy.

Table 5.5 Classification Report Using 7 Raw Features

	precision	recall	f1-score	support
0	0.83	0.92	0.87	861502
1	0.81	0.01	0.03	57221
2	0.00	0.00	0.00	38616
3	0.00	0.00	0.00	37614
4	0.03	0.04	0.03	32547
5	0.00	0.00	0.00	128
6	0.04	0.04	0.04	50
7	0.00	0.12	0.00	17
accuracy			0.77	1027695
macro avg	0.21	0.14	0.12	1027695
weighted avg	0.74	0.77	0.73	1027695

Table 5.6 showed the performance metrics of decision tree model that used all raw features and constructed features to classify each kind of network attack and benign traffic with cross-validation strategy.

Table 5.6 The Classification Report Using All Raw and Constracted Features

	precision	recall	f1-score	support
0	0.94	0.76	0.84	855969
1	1.00	0.99	0.99	57231
2	1.00	1.00	1.00	39048
3	1.00	0.50	0.67	37330
4	0.05	0.28	0.08	32278
5	0.83	0.40	0.54	126
6	0.00	0.94	0.01	49
7	0.00	0.00	0.00	8
accuracy			0.76	1022039
macro avg	0.60	0.61	0.52	1022039
weighted avg	0.92	0.76	0.83	1022039

Table 5.7 showed the performance metrics of decision tree model that used ANOVA F-test to obtain the feature subset from all raw and constructed features and classified each kind of network attack and benign traffic with cross-validation strategy.

Table 5.7 The Classification Report Using ANOVA F-test

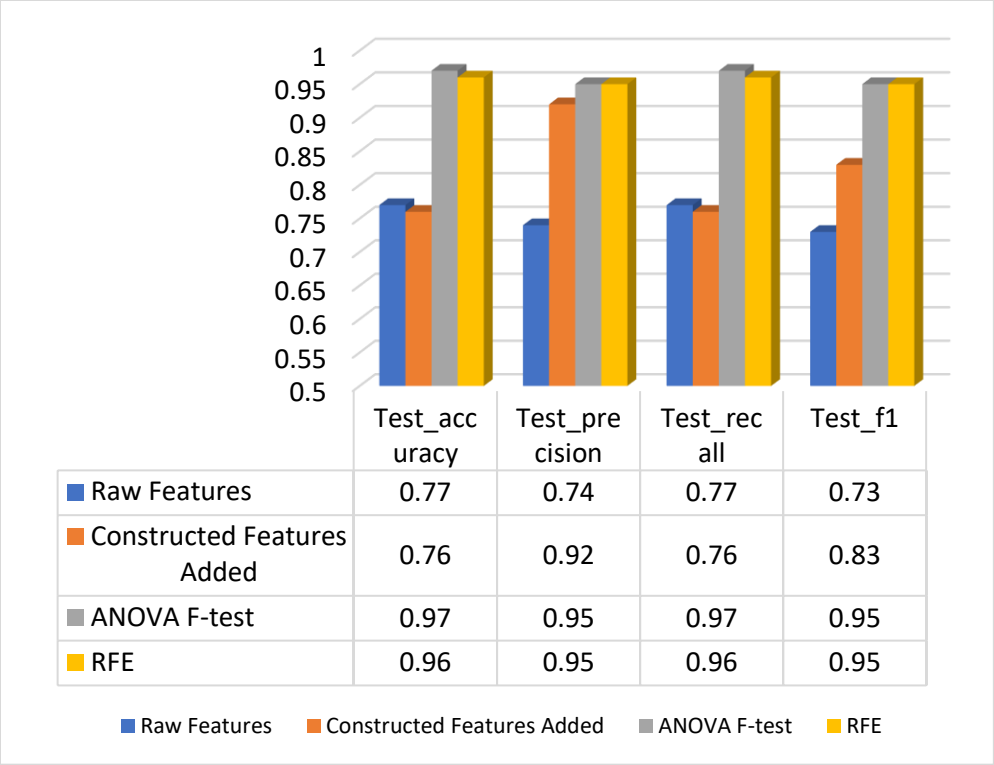
	precision	recall	f1-score	support
0	0.96	1.00	0.98	855969
1	0.99	0.97	0.98	57231
2	1.00	1.00	1.00	39048
3	1.00	1.00	1.00	37330
4	0.35	0.03	0.06	32278
5	0.89	0.27	0.41	126
6	0.75	0.31	0.43	49
7	0.00	0.00	0.00	8
accuracy			0.97	1022039
macro avg	0.74	0.57	0.61	1022039
weighted avg	0.95	0.97	0.95	1022039

Table 5.8 showed the performance metrics of decision tree model that used Recursive Feature Elimination to obtain the feature subset from all raw and constructed features and classified each kind of network attack and benign traffic with cross-validation strategy.

Table 5.8 The Classification Report Using RFE

	precision	recall	f1-score	support
0	0.97	0.99	0.98	855969
1	1.00	1.00	1.00	57231
2	1.00	1.00	1.00	39048
3	1.00	1.00	1.00	37330
4	0.24	0.10	0.14	32278
5	0.98	0.33	0.49	126
6	0.96	0.47	0.63	49
7	0.07	0.12	0.09	8
accuracy			0.96	1022039
macro avg	0.78	0.63	0.67	1022039
weighted avg	0.95	0.96	0.95	1022039

In Figure 5.4, the overall test scores of decision tree model using constructed features and feature selection techniques were much better than using raw feature. Especially combing feature selection and constructed features reached more than 95% on all test scores. But using raw features only got lower than 80% for all test scores.



	Test_accuracy	Test_precision	Test_recall	Test_f1
Raw Features	0.77	0.74	0.77	0.73
Constructed Features Added	0.76	0.92	0.76	0.83
ANOVA F-test	0.97	0.95	0.97	0.95
RFE	0.96	0.95	0.96	0.95

Figure 5.4 Weighted Average of Accuracy, Precision, Recall, and F1 Score for Four Experiments

5.4.2 Test Scores of Four Experiments Based on Different Network Attacks.

This section provided comparison on the precision, recall, and f1 scores for predicting different kinds of network attacks in four experiments.

By looking at Figure 5.5, 5.6, and 5.7, the decision tree model that used ANOVA F-test and Recursive Feature Elimination to obtain the feature subset presented good performance on detecting botnet attack, FTP-BruteFrce attack, and SSH-BruteForce attack. The precision, recall, and f1 score of detecting botnet attack, FTP-BruteFrce attack, and SSH-BruteForce attack when using feature selection techniques and constructed features reached more than 97%, which were way better than the test scores using raw features.

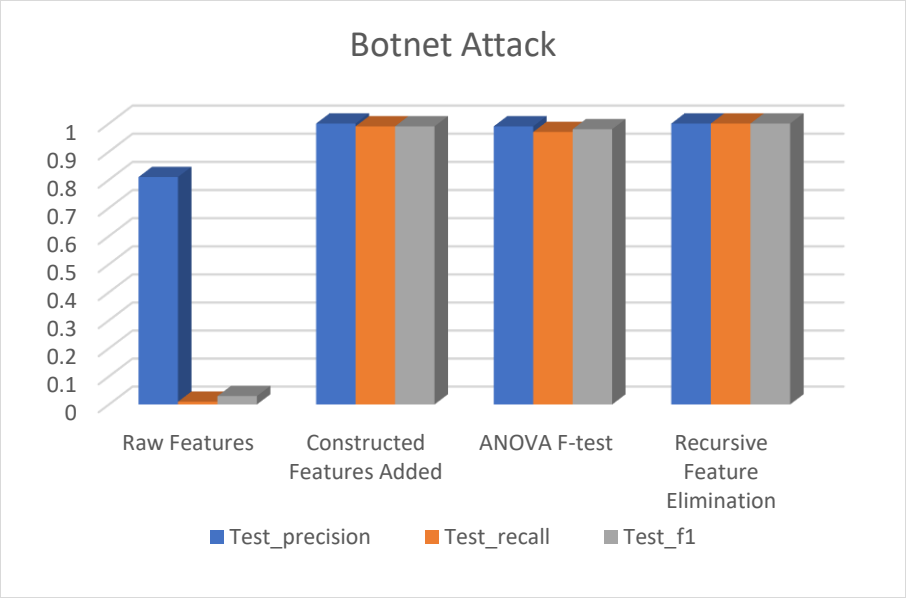


Figure 5.5 Test Scores of Four Experiments on Botnet Attack

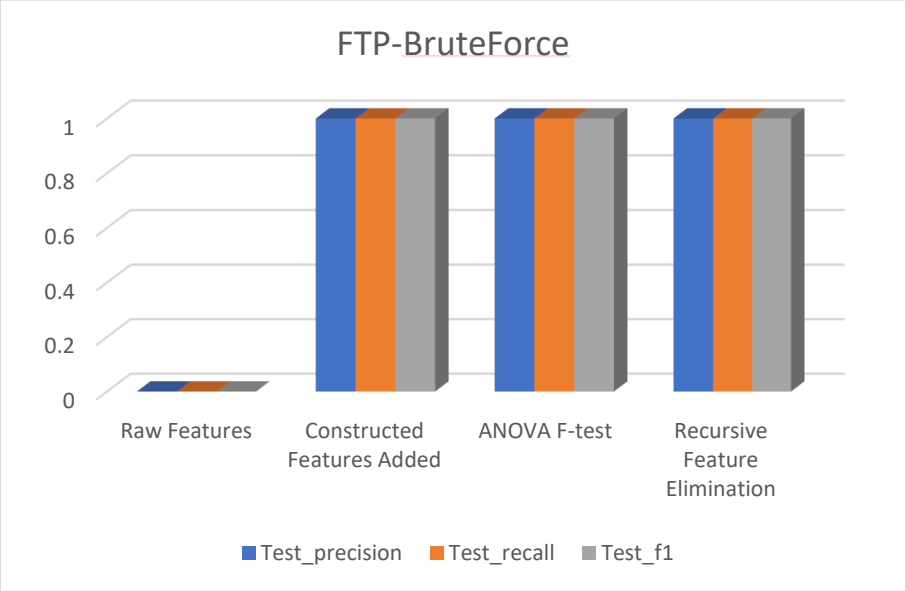


Figure 5.6 Test Scores of Four Experiments on FTP-BruteForce

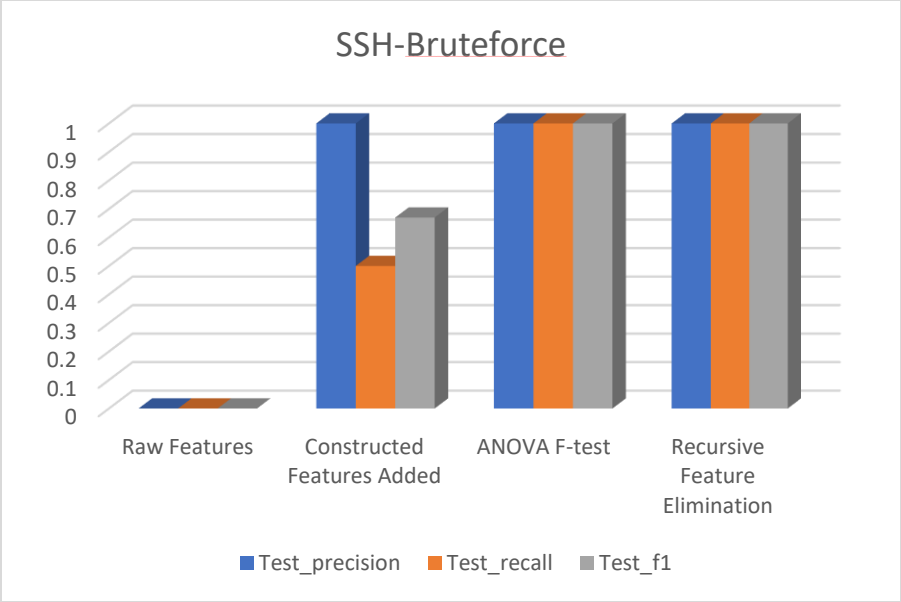


Figure 5.7 Test Scores of Four Experiments on SSH-BruteForce

However, when detecting infiltration attack, web attack, and SQL injection, the precision, recall, and f1 score were not increased dramatically with feature selection and constructed features shown in Figure 5.8, 5.9, and 5.10. But still, the test scores using feature selection and constructed features was slightly better than using raw features.

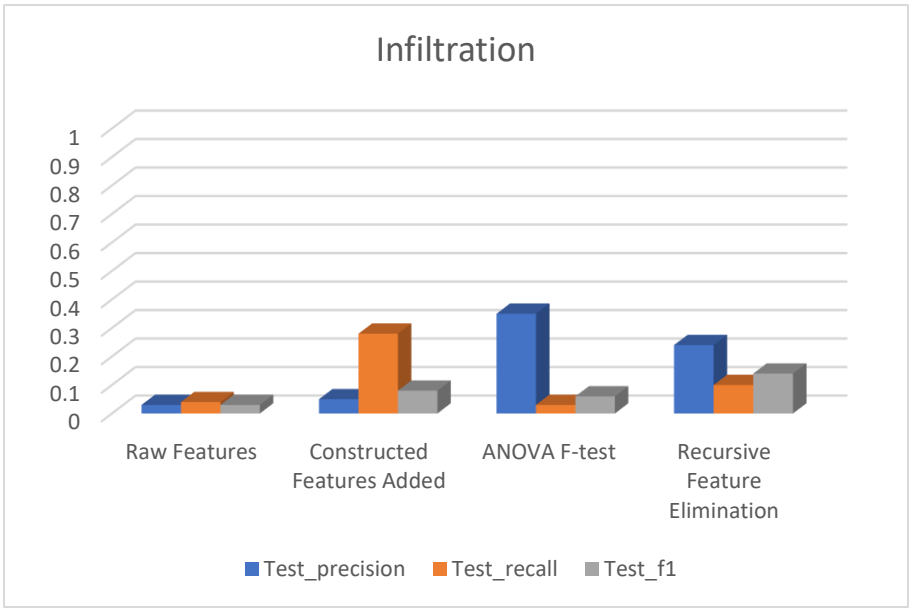


Figure 5.8 Test Scores of Four Experiments on Infiltration Attack

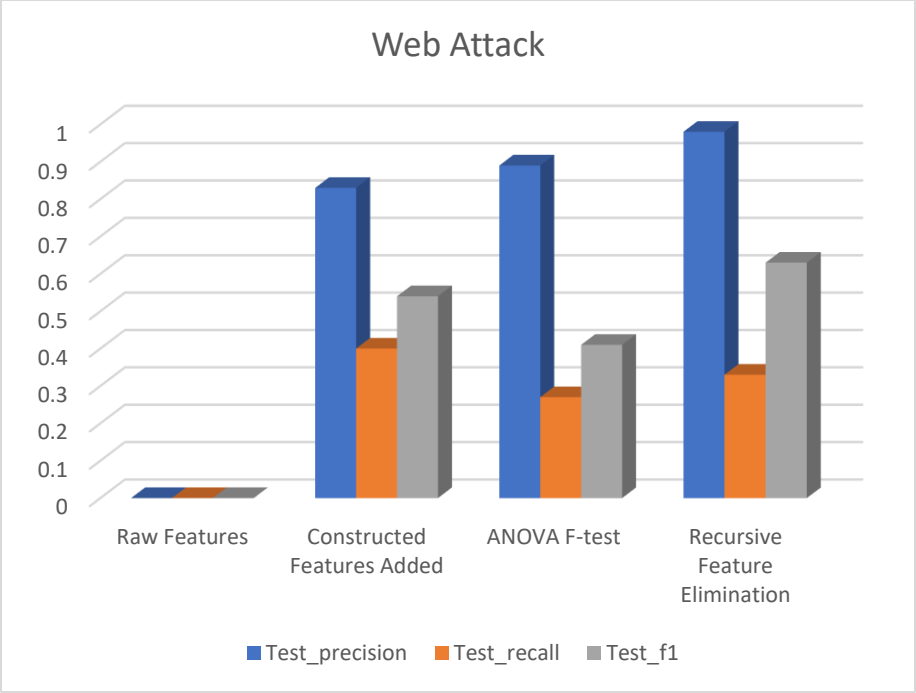


Figure 5.9 Test Scores of Four Experiments on Web Attack

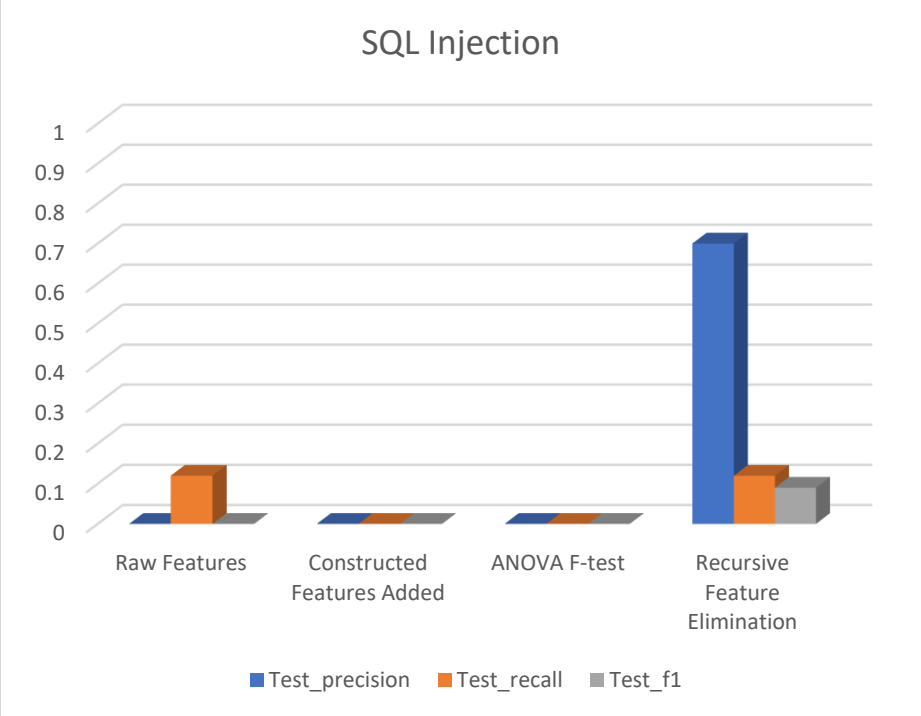


Figure 5.10 Test Scores of Four Experiments on SQL Injection

5.4.3 Training and Testing Time of Four Experiments

This section compared the training time and testing time of four experiments.

In Figure 5.11, adding constructed features to raw features increased the training time dramatically and took over 327 seconds to train the decision tree model, because more data was added into the dataset that required decision tree model to spend more time to process them. However, after using ANOVA F-test and Recursive Feature Elimination, the training time was reduced to around 29 seconds, even lower than the time used for training with raw features.

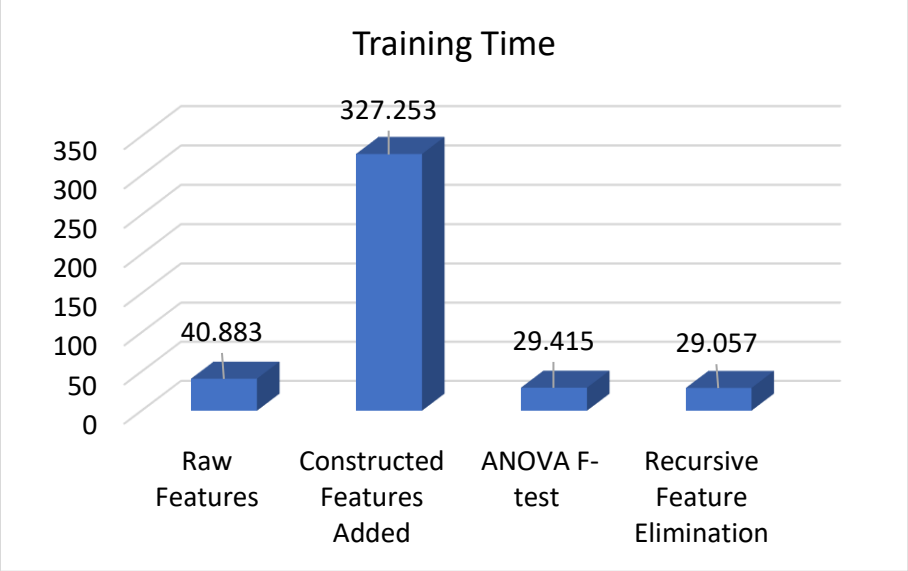


Figure 5.11 Training Time of Four Experiments

In Figure 5.12, the testing time presented the similar situation as training time shown in Figure 5.11. The combination of constructed features and raw features made the decision tree model spend much more time to produce the prediction result than only using raw features or employing two feature selection techniques. However, the testing time of using feature selection techniques was slightly longer than using raw features.

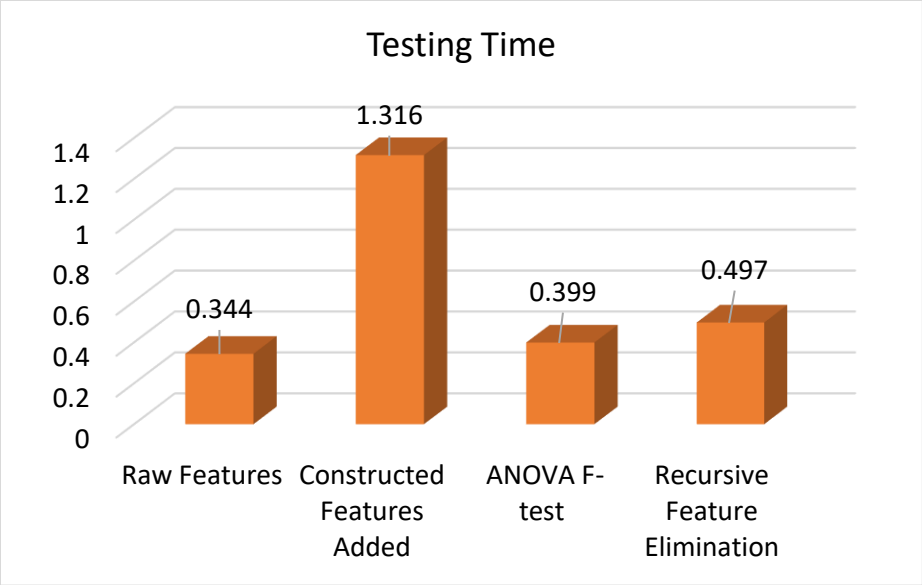


Figure 5.12 Testing Time of Four Experiments

CHAPTER 6. CONCLUSION

The goal of this project was to combine feature selection and derived features to achieve the better performance of the machine learning model in network intrusion detection. The objective was finished by constructing attack characteristic features using CICFlowMeter and creating feature subset using ANOVA F-test and Recursive Feature Elimination. The project chose decision tree as the machine learning model to detect the network attacks according to the CSE-CIC-IDS2018 dataset from Canadian Institute for Cybersecurity and Communications Security Establishment. The machine learning model and the test environment were developed using Python programming language on Jupyter Notebook. This study showed that the combination of feature selection algorithms and derived features could effectively bring up the prediction accuracy, precision, recall, and f1 score of decision tree model, and slightly reduce the training and testing time.

The CSE-CIC-IDS2018 dataset used in the project was the most recent benchmark intrusion detection dataset. The dataset provided a large number of samples for various kinds of popular network attacks in the Internet, including Brute Force, Web attack, Infiltration, and Botnet attack. However, the only weakness of the CSE-CIC-IDS2018 dataset was that the sample distribution of network attacks wasn't quite balanced, which may influence the performance of the machine learning model in some degree. In order to reduce the impact of imbalanced samples, the project employed the cross-validation technique to split the dataset into multiple folds that contained same distribution of samples from different classes.

In the future work, more samples for different kinds of network attacks will be collected to enrich and balance the current CSE-CIC-IDS2018 dataset. Since this project only tested ANOVA F-test and Recursive Feature Elimination, more feature selection techniques will be applied to explore potential combinations for better prediction in network intrusion detection.

APPENDIX A. CODE

```
import glob
import pandas as pd

path = r'/IDS2018'
files = glob.glob( "Traffic*.csv")
L = []                                                                  I
for filename in files:
    df = pd.read_csv(filename, index_col=None, header=0)
    L.append(df)
df = pd.concat(L, axis=0, ignore_index=True)
```

```
df = df.drop(['Timestamp'], axis=1)

cols = df.columns.drop('Label')
df[cols] = df[cols].apply(pd.to_numeric, errors='coerce')

df = df.dropna()                                                        II
df = df[(df != 'Infinity').all(axis=1)]
df = df.reset_index(drop=True)

df = df.round(1)
```

```
Labeldf = df['Label']
newlabeldf = labeldf.replace({ 'Benign':0, 'Bot':1,
                               'FTP-BruteForce':2,
                               'SSH-Bruteforce':3,
                               'Infilteration':4,                     III
                               'Brute Force -Web':5,
                               'Brute Force -XSS':6,
                               'SQL Injection':7})
df['Label'] = newlabeldf
```

```
y = df['Label']
xTrain, xTest, yTrain, yTest = train_test_split(df.values, y,
                                                test_size = 0.2,
                                                random_state = 42)
                                                                        IV
xTrain = xTrain[:,:-1]
xTest = xTest[:,:-1]
```

```
from sklearn import preprocessing

scaler1 = preprocessing.MinMaxScaler().fit(xTrain)
xTrain=scaler1.transform(xTrain)                                    V

scaler2 = preprocessing.MinMaxScaler().fit(xTest)
xTest=scaler2.transform(xTest)
```

```
from sklearn.feature_selection import SelectPercentile, f_classif
from sklearn.feature_selection import RFE

np.seterr(divide='ignore', invalid='ignore');
selector=SelectPercentile(f_classif, percentile=10)
x_f_train = selector.fit_transform(xTrain, yTrain)                  VI

clf = DecisionTreeClassifier(random_state=0)
rfe = RFE(estimator=clf, n_features_to_select=5, step=1)
rfe.fit(xTrain, yTrain)
x_rfe_train=rfe.transform(xTrain)
```

```
from sklearn.tree import DecisionTreeClassifier

clf_all = DecisionTreeClassifier(criterion='gini',                 VII
                                 random_state=0)
clf_all.fit(xTrain, yTrain)
```

```
Y_all_pred = clf_all.predict(xTest)                                VIII
```

```
from sklearn.metrics import make_scorer
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.model_selection import cross_validate

scoring = {'accuracy': 'accuracy',
           'precision': make_scorer(precision_score, average='weighted'),
           'recall': make_scorer(recall_score, average='weighted'),
           'f1': make_scorer(f1_score, average='weighted'),
           }                                                      VIIII

def average_score_on_cross_val_classification(clf_all, xTest, yTest,
                                              scoring=scoring, cv=8):
    scores_dict = cross_validate(clf_all, xTest, yTest, scoring=scoring,
                                 cv=4, n_jobs=-1)
    return {metric: round(np.mean(scores), 5) for metric,
                    scores in scores_dict.items()}

average_score_on_cross_val_classification(clf_all, xTest, yTest)
```

REFERENCES

[1] National Telecommunications and Information Administration, “Digital Nation Data Explorer,” *Digital Nation Data Explorer | National Telecommunications and Information Administration*, 2018. [Online]. Available: https://www.ntia.doc.gov/data/digital-nation-data-explorer#sel=internetUser&demo=&pc=count&disp=chart. [Accessed: 28-Nov-2019].

[2] T. Brewster, “Capital One Hacker 'Breached 30 Organizations And Mined Cryptocurrency,' Claims DOJ,” *Forbes*, 30-Aug-2019. [Online]. Available: https://www.forbes.com/sites/thomasbrewster/2019/08/29/alleged-capital-one-hacker-breached-30-organizations-and-mined-cryptocurrency/. [Accessed: 28-Nov-2019].

[3] M. K. Asif, T. A. Khan, T. A. Taj, U. Naeem, and S. Yakoob, “Network Intrusion Detection and its strategic importance,” *2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*, 2013.

[4] T. Mehmood and H. B. M. Rais, “Machine learning algorithms in context of intrusion detection,” *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, 2016.

[5] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, “Deep Abstraction and Weighted Feature Selection for Wi-Fi Impersonation Detection,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 621–636, 2018.

[6] O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, “Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, Oct. 2016.

[7] F. Iglesias and T. Zseby, “Analysis of network traffic features for anomaly detection,” *Machine Learning*, vol. 101, no. 1-3, pp. 59–84, Apr. 2014.

[8] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee, “An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection,” *Applied Soft Computing*, vol. 12, no. 10, pp. 3285–3290, 2012.

[9] J. Cai, J. Luo, S. Wang, and S. Yang, “Feature selection in machine learning: A new perspective,” *Neurocomputing*, vol. 300, pp. 70–79, 2018.

[10] P. V. Balachandran, D. Xue, J. Theiler, J. Hogden, J. E. Gubernatis, and T. Lookman, "Importance of Feature Selection in Machine Learning and Adaptive Design for Materials," *Materials Discovery and Design Springer Series in Materials Science*, pp. 59–79, 2018.

[11] A. R. Yusof, N. I. Udzir, A. Selamat, H. Hamdan, and M. T. Abdullah, "Adaptive feature selection for denial of services (DoS) attack," *2017 IEEE Conference on Application, Information and Network Security (AINS)*, 2017.

[12] J.-H. Woo, J.-Y. Song, and Y.-J. Choi, "Performance Enhancement of Deep Neural Network Using Feature Selection and Preprocessing for Intrusion Detection," *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2019.

[13] Y.-L. Wan, J.-C. Chang, R.-J. Chen, and S.-J. Wang, "Feature-Selection-Based Ransomware Detection with Machine Learning of Data Analysis," *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, 2018.

[14] S. Vaithyasubramanian, A. Christy, and D. Saravanan, "An analysis of Markov password against Brute force attack for effective web applications," *Applied Mathematical Sciences*, vol. 8, pp. 5823–5830, 2014.

[15] Cybersecurity and Infrastructure Security Agency, "Brute Force Attacks Conducted by Cyber Actors: CISA," *Brute Force Attacks Conducted by Cyber Actors | CISA*, 27-Mar-2019. [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA18-086A##targetText=During a password-spray attack,rapid or frequent account lockouts. [Accessed: 28-Nov-2019].

[16] J. Vykopal, "A Flow-Level Taxonomy and Prevalence of Brute Force Attacks," *Advances in Computing and Communications Communications in Computer and Information Science*, pp. 666–675, 2011.

[17] S.-L. Peng, S.-J. Wang, V. E. Balas, and M. Zhao, *Security with Intelligent Computing and Big-data Services*, vol. 733. Cham: Springer International Publishing, 2018.

[18] S. Wilkins, "Pearson IT Certification," *TCP/IP Ports and Protocols | Pearson IT Certification*, 30-Apr-2012. [Online]. Available: http://www.pearsonitcertification.com/articles/article.aspx?p=1868080. [Accessed: 28-Nov-2019].Karrick, S. (2001). FTP Port 21 "Friend or Foe" Support for the Cyber Defense Initiative.

[19] S. Karrick, “FTP Port 21 "Friend or Foe" Support for the Cyber Defense Initiative”, *SANS Institute*, 2001

[20] M. M. Najafabadi, T. M. Khoshgoftaar, C. Kemp, N. Seliya, and R. Zuech, “Machine Learning for Detecting Brute Force Attacks at the Network Level,” *2014 IEEE International Conference on Bioinformatics and Bioengineering*, 2014.

[21] M. Vizváry and J. Vykopal, “Flow-based detection of RDP brute-force attacks,” *2013 7th International Conference on Security and Protection of Information, SPI*, vol. 13, pp. 131-138, 2013.

[22] Y. Aleksieva, H. Valchanov, and V. Aleksieva, “An approach for host based botnet detection system,” *2019 16th Conference on Electrical Machines, Drives and Power Systems (ELMA)*, 2019.

[23] L. Zhang, S. Yu, D. Wu, and P. Watters, “A Survey on Latest Botnet Attack and Defense,” *2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, 2011.

[24] D. Plohmann, E. Gerhards-Padilla, and F. Leder, “Botnets: Detection, measurement, disinfection & defence,” *European Network and Information Security Agency (ENISA), 1*(1), pp.1-153, 2011

[25] M. R. Thakur, D. R. Khilnani, K. Gupta, S. Jain, V. Agarwal, S. Sane, S. Sanyal, and P. S. Dhekne, “Detection and prevention of botnets and malware in an enterprise network,” *International Journal of Wireless and Mobile Computing*, vol. 5, no. 2, p. 144, 2012.

[26] G. Khehra and S. Sofat, “Botnet Detection Techniques: A Review,” *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2018.

[27] I. Ullah, N. Khan, and H. A. Aboalsamh, “Survey on botnet: Its architecture, detection, prevention and mitigation,” *2013 10th IEEE International Conference on Networking, Sensing and Control (ICNSC)*, 2013.

[28] K. Alieyan, A. Almomani, A. Manasrah, and M. M. Kadhum, “A survey of botnet detection based on DNS,” *Neural Computing and Applications*, vol. 28, no. 7, pp. 1541–1558, Dec. 2015.

[29] G. K. Varadarajan and M. Santander Peláez, “Web application attack analysis using bro ids,” *SANS Institute, 90*, 2012.

[30] C. Obimbo, K. Ali, and K. Mohamed, "Using IDS to prevent XSS Attacks," *International Conference on Security and Management (SAM)*, pp. 233-239, 2017.

[31] Levinec, "Exploits and exploit kits," *Microsoft Docs*, 2019. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/exploits-malware. [Accessed: 28-Nov-2019].

[32] A. K. Kaushik, E. S. Pilli, and R. Joshi, "Network forensic system for port scanning attack," *2010 IEEE 2nd International Advance Computing Conference (IACC)*, 2010.

[33] M. Hasanifard and B. T. Ladani, "DoS and port scan attack detection in high speed networks," *2014 11th International ISC Conference on Information Security and Cryptology*, 2014.

[34] M. S. Kumar, J. Ben-Othman, K. Srinivasagan, and G. U. Krishnan, "Artificial Intelligence Managed Network Defense System against Port Scanning Outbreaks," *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, 2019.

[35] S. Visalakshi and V. Radha, "A literature review of feature selection techniques and applications: Review of feature selection in data mining," *2014 IEEE International Conference on Computational Intelligence and Computing Research*, 2014.

[36] P.-S. Tang, X.-L. Tang, Z.-Y. Tao, and J.-P. Li, "Research on feature selection algorithm based on mutual information and genetic algorithm," *2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP)*, 2014.

[37] G. Manikandan, E. Susi, and S. Abirami, "Feature Selection on High Dimensional Data Using Wrapper Based Subset Selection," *2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM)*, 2017.

[38] C. Gupta, "Feature Selection and Analysis for Standard Machine Learning Classification of Audio Beehive Samples*," Utah State University*, 2019, [Doctoral dissertation].

[39] S. Cateni, V. Colla, and M. Vannucci, "A Hybrid Feature Selection Method for Classification Purposes," *2014 European Modelling Symposium*, 2014.

[40] G. Smith, "Step away from stepwise," *Journal of Big Data*, vol. 5, no. 1, 2018.

[41] B. Sahu, S. Dehuri, and A. Jagadev, "A Study on the Relevance of Feature Selection Methods in Microarray Data," *The Open Bioinformatics Journal*, vol. 11, no. 1, pp. 117–139, 2018.

[42] L. Ladha and T. Deepa, "Feature selection methods and algorithms," *International journal on computer science and engineering*, *3*(5), 1787-1797, 2011.

[43] J. Peltonen, "Lecture 2: Feature selection," *Dimensionality Reduction and Visualization*, 2014. [PowerPoint slides]. [Accessed: 28-Nov-2019].

[44] Q. Zhou and D. Pezaros, "Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection--An Analysis on CIC-AWS-2018 dataset," *arXiv preprint arXiv:1905.03685*, 2019.

[45] I.-V. Onut and A. Ghorbani, "Toward A Feature Classification Scheme For Network Intrusion Detection," *4th Annual Communication Networks and Services Research Conference (CNSR06)*, 2007.

[46] K. Wang, C.-Y. Huang, S.-J. Lin, and Y.-D. Lin, "A fuzzy pattern-based filtering algorithm for botnet detection," *Computer Networks*, vol. 55, no. 15, pp. 3275–3286, 2011.

[47] J. Jin, Z. Yan, B. Yan, and G. Geng, "Botnet Domain Name Detection based on Machine Learning," *6th International Conference on Wireless, Mobile and Multi-Media (ICWMMN 2015)*, 2015.

[48] Q. Liao, H. Li, S. Kang, and C. Liu, "Feature extraction and construction of application layer DDoS attack based on user behavior," *Proceedings of the 33rd Chinese Control Conference*, 2014.

[49] C. Torrano-Gimenez, H. T. Nguyen, G. Alvarez, and K. Franke, "Combining expert knowledge with automatic feature extraction for reliable web attack detection," *Security and Communication Networks*, vol. 8, no. 16, pp. 2750–2767, 2012.

[50] M. Ring, D. Landes, and A. Hotho, "Detection of slow port scans in flow-based network traffic," *Plos One*, vol. 13, no. 9, 2018.

[51] X. Kong and P. S. Yu, "Multi-label Feature Selection for Graph Classification," 2010 IEEE International Conference on Data Mining, Sydney, NSW, 2010, pp. 274-283. doi: 10.1109/ICDM.2010.58.

[52] Google Cloud, "Data preprocessing for machine learning: options and recommendations | Solutions | Google Cloud," *Google*, Apr-2019. [Online]. Available: https://cloud.google.com/solutions/machine-learning/data-preprocessing-for-ml-with-tf-transform-pt1. [Accessed: 28-Nov-2019].

[53] S. B. Kotsiantis, D. Kanellopoulos and P. E. Pintelas, "Data preprocessing for supervised leaning," *International Journal of Computer Science*, *1*(2), 111-117, 2006.

[54] T. M. Mitchell, "Machine learning," *Amazon*, 2017. [Online]. Available: https://docs.aws.amazon.com/machine-learning/latest/dg/training-parameters.html. [Accessed: 28-Nov-2019].

[55] Canadian Institute for Cybersecurity, "CSE-CIC-IDS2018 on AWS," *University of New Brunswick est.1785*, 2018. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html. [Accessed: 28-Nov-2019].

[56] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a Reliable Intrusion Detection Benchmark Dataset," *Software Networking*, vol. 2017, no. 1, pp. 177–200, 2017.

[57] University of Central Florida, "Python Lists vs. Numpy Arrays - What is the difference?," *IST Advanced Topics Primer*. [Online]. Available: https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference. [Accessed: 01-Jun-2020].

[58] Riturajsaha, "Why Numpy is faster in Python?," *GeeksforGeeks*, 26-Feb-2020. [Online]. Available: https://www.geeksforgeeks.org/why-numpy-is-faster-in-python/. [Accessed: 01-Jun-2020].

[59] G. McIntire, B. Martin, and L. Washington, "Python Pandas Tutorial: A Complete Introduction for Beginners," *Learn Data Science - Tutorials, Books, Courses, and More*. [Online]. Available: https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/. [Accessed: 01-Jun-2020].

[60] J. Brownlee, "A Gentle Introduction to Scikit-Learn," Machine Learning Mastery, 21-Aug-2019. [Online]. Available: https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/. [Accessed: 01-Jun-2020].

[61] R. S. Brid, "Decision Trees - A simple way to visualize a decision," *Medium*, 26-Oct-2018. [Online]. Available: https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb. [Accessed: 31-May-2020].

[62] scikit-learn, "sklearn.tree.DecisionTreeClassifier¶," scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decision tree#sklearn.tree.DecisionTreeClassifier. [Accessed: 15-Jun-2020].

[63] pandas, "pandas.to_numeric¶," *pandas.to_numeric - pandas 1.0.4 documentation*. [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_numeric.html. [Accessed: 16-Jun-2020].

[64] scikit-learn, “sklearn.model_selection.train_test_split¶,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html?highlight=train_test_split#sklearn.model_selection.train_test_split. [Accessed: 16-Jun-2020].

[65] A. Bhandari, “Feature Scaling: Standardization Vs Normalization,” *Analytics Vidhya*, 14-Apr-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/. [Accessed: 16-Jun-2020].