ENCODING IP ADDRESS AS A FEATURE FOR NETWORK INTRUSION DETECTION

by

Enchun Shao

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

December 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

Dr. Baijian Yang, Chair

Department of Computer and Information Technology

Dr. Wenhai Sun

Department of Computer and Information Technology

Dr. Tonglin Zhang

Department of Statistics

Approved by:

Dr. Eric T. Matson

Head of the Graduate Program

To my beloved family for their supports.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

TN	True Negative
TP	True Positive
FN	False Negative
FP	False Positive
IP	Internet Protocol
IPv4	Internet Protocol version 4
SVM	Supported Vector Machine
RAM	Random-access Memory
IDS	Intrusion Detection System
CIC	Canadian Institute for Cybersecurity

ABSTRACT

As machine learning algorithms take on more important roles in various areas of big data analysis, more accurate research is needed. Although machine learning techniques have been applied in network intrusion detection, encoding IP addresses as a feature of network intrusion detection has not been discussed. Since the IP address is strongly relevant to network intrusion detection, it cannot be ignored when predicting a network attack. Therefore, three machine learning algorithms - random forest, support vector machine (SVM), and decision tree have been applied for the present study to examine three IP address encoding methods for NetFlow data: converting into four individual numbers, converting into binary integers, and one hot encoding. The pivot of the study was to analyze the F-1, precision, recall, and accuracy scores of the machine learning algorithms and determine the best method of encoding IP addresses for network intrusion detection. In addition, 21 features of the data set related to the destination port, packets, destination IP, source port, flow duration, source IP, flags, and labels were also considered, as well as speed. The study shows that the best method of encoding an IP address was splitting the IP address into four numbers and the decision tree, random forest, and SVM accuracy scores for this method were 0.9562, 0.9631, and 0.9296, respectively.

CHAPTER 1. INTRODUCTION

This chapter reviewed the current study on encoding internet protocol (IP) addresses as a feature of machine learning algorithms. It firstly presented the statement of the problem, followed by introducing the research assumptions, significance, limitations, and delimitation, culminating in the research question regarding the best method of encoding IP addresses for network intrusion detection.

1.1 Statement of the Problem

An IP address is displayed as a numerical label, which is assigned to devices connected to a network. Although the IP address is a numerical label, it cannot be learned directly by machine learning algorithms. IP addresses have not been considered in the application of machine learning algorithms because they are not quantitative variables (Sharafaldin, Habibi Lashkari, & Ghorbani, 2018) and determining how to convert it has been difficult. However, the IP address is quite relevant in a network attack, and it must be considered in network intrusion detection. For example, IP packets are usually sent from a spoofed address so that attackers can disguise themselves. In fact, IP address spoofing is one of the most frequent ways to attack a network.

Network attacks are usually considered as unauthorized actions against private, corporate, or government information technology (IT) assets. The present research of network attack synthesis focused on an analysis of a NetFlow database. To evaluate the accuracy of predicting a network attack, IP addresses were encoded in three different ways. After that, SVM, random forest and decision tree were implemented to determine the best way of converting IP addresses.

1.2 Scope

The data sets used in the present study were collected by one of the most important defense tools against the ever-growing number of sophisticated network attacks: the Intrusion Detection Systems (IDS) from the Canadian Institute for Cybersecurity (CIC). The CIC works across disciplines to cultivate and create a national network of research talent from the user sector, industry, government, academia to address and examine internet privacy, security, and trust. The CICs program on the science of cybersecurity focused on enabling the network security solutions and development of information for the evolving data-intensive cyber domain by focusing on the meaning and value of security data (Sharafaldin et al., 2018). Moreover, the CIC data sets were well-labeled, which saved a great deal of time during the data pre-processing phase of the present study.

According to the CIC team (Sharafaldin et al., 2018), the process of data capture lasted for a week. On the first day of the capture period, the system collected benign traffic data and stored it in a file. During the rest of the week, the system tried to collect attack traffic data. The network attacks included bruteforce attacks, DoS attacks, web attacks, infiltration attacks, botnet attacks, DDoS attacks, and PortScan attacks. The data for each attack was stored in one file, with every file containing hundreds of thousands of rows (Sharafaldin et al., 2018).

1.3 Research Question

To encode the IP addresses, three methods were applied: (1) converting the IP addresses into binary integers, (2) splitting the IP addresses into four numbers (since the IP addresses were collected as IPv4 addresses) and (3) one hot encoding. One hot encoding was used to convert the host (32-bit IP addresses) and subnets of 24-bit and 16-bit IP addresses, because it is a good method of converting categorical variables into a binary-like form for better machine learning algorithm predictions. After encoding the IP addresses, accuracy scores for predicting network attacks were compared.

- Which method of encoding IP address is the best for network intrusion detection among split IP, binary IP, and one hot encoding?

1.4 Significance of the Problem

While machine learning algorithms had been applied for the prediction of network attacks, the IP address had typically been ignored because it was a categorical value (Sharafaldin et al., 2018). In contrast, the present study aimed to apply three methods of encoding IP addresses, namely, one hot encoding, converting IP addresses into binary numbers, and splitting IP addresses into four numbers, to determine which method is the best one for network intrusion detection. Since an IP address would be normally converted into four numbers or a binary integer, the novelty of this study is that compares one hot encoding with the other two methods. In fact, few studies have applied supervised learning in using one hot encoding for IP addresses for intrusion detection.

1.5 Assumptions

- The original data sets were considered to be clean and well-labeled.
- Data was considered to resemble real-world situations.
- One-hot encoding typical offers best results for categorical data.
- The IP address in the data sets could be spoofed.

1.6 Limitations

- The work only considered the device including at least 16GB RAM.
- Only seven kinds of network attack traffic were collected in the original data sets.
- Too much benign network traffic was collected than network attacks from original resources.
- All data were collected in a week in 2017 and each day had been assigned for one malicious task.

1.7 Delimitation

- Not all of the features were considered for machine learning algorithms.
- Not all the data were considered for the method of one hot encoding.
- The IP address was not considered as the public IP or private IP.

1.8 Summary

The statement of the problem and the research question of this thesis were contained in this chapter, and the chapter also covered the study's scope, limitations, and delimitation. Relevant works from the literature are discussed in the next chapter.

CHAPTER 2. REVIEW OF LITERATURE

This chapter included an overview of machine learning, ensemble learning, and feature selection for malicious network traffic detection. It also discussed studies on the performance of machine learning algorithms and ensemble learning methods, as well as algorithms and features that were relevant to the present research.

2.1 Overview of Machine Learning for Network Attack Detection

Analyzing flow data related to network security and machine learning algorithms are typical areas of focus for most researchers. The research group from Aalborg University (Stevanovic & Pedersen, 2014) explored how botnet detection could be achieved by using supervised machine learning. To figure out this problem, a novel flow-based detection system based on supervised machine learning was built to identify botnet network traffic. The system, which consisted of two main elements, the classifier entity and the pre-processing entity, classified network traffic as non-malicious or malicious using algorithms. Eight algorithms were considered, and the best one was indicated. Moreover, the classifier entity implemented supervised machine learning algorithms to categorize traffic flows as non-malicious or malicious, and the network traffic on flow level was analyzed by the pre-processing entity (Stevanovic & Pedersen, 2014, p. 798). The results demonstrated that a novel flow-based detection system could detect the botnet, and random tree classifiers and simple flow features were proven to accurately detect botnet traffic (Stevanovic & Pedersen, 2014).

The research group from Chongqing University of Posts and Telecommunications (Deng, Luo, Liu, Wang, & Yang, 2014) considered that machine learning was a feasible approach to address P2P traffic identification, which was an important problem in internet traffic analysis. In their study, feature weighted Naive Bayes and random forest were integrated into P2P traffic identification, and the Hadoop pseudo-distributed test platform was used to test the effectiveness and stability of the models. The prediction for each model was processed by calculating the scores, and the weighted majority voting was applied to achieve the final output. The results showed that these models performed better than other methods, and P2P traffic could be solved in an alternative way. However, the performance of the ensemble learning model should be enhanced by studying other combination and classification methods and analyzing useful features (Deng et al., 2014).

The research group from the Internet Work Research Department of BBN Technologies (Livadas, Walsh, Lapsley, & Strayer, 2008, p. 970) tried to identify IRC-based botnets and command and control (C2) traffic. Their research was split by distinguishing real IRC traffic and botnet and distinguishing non-IRC traffic and IRC. In part one, the effectiveness of machine learning-based classification was used to identify chat traffic in three dimensions: the subset of characteristics, the classification scheme, and features used to describe the size of the training set and the flows. In part two, real IRC flows were applied for training, and and more testbed experiments were run and collected testbed botnet. Finally, an evaluation of whether machine learning-based classifiers were applied to distinguish between real IRC flows and botnet was described.

According to (Miller & Busby-Earle, 2016), machine learning algorithms played an important role in botnet detection. Real-time online detection approaches of both effectiveness and efficiency have been developed from robust models. Their research focused on how different approaches based on machine learning used various machine learning methods to detect botnet activity. The combination of machine learning method features and the best technique for addressing the botnet detection problem were determined. As a result, supervised learning was observed to be a command trend for detection. Given specific malicious traffic, supervised learning was more effective and accurate against bot traffic that sought to camouflage itself among legitimate traffic. In contrast, unsupervised learning was used to capture group activity by bots in a botnet and was not unique to any type of bot (Miller & Busby-Earle, 2016).

2.2 Overview of Machine Learning in Different Areas

The choice of machine learning area is an important one and is directly to the results of any research. Thus, it is necessary to review the implementation of machine learning algorithms in other areas. For example, text categorization is important for many organizations and for information management. Besides feature coding, classifier design is one of the major problems for text categorization. Multiple statistical classification and machine learning methods, such as probabilistic Bayesian models, multivariate regression models and nearest neighbor classifiers have been applied in text categorization. In addition, (Yang & Liu, 1999) reported k-nearest neighbor as one of the top-performing methods for text categorization. K-nearest neighbor (KNN) is a classification method contingent on statistic theory (Mitchell & Schaefer, 2001), and it was usually implemented in the data mining classification algorithm. The research group from Hebei University (Zhan, Chen, Zhang, & Zheng, 2009) used a KNN algorithm based on feature weight learning via text categorization, proving the validity of these algorithms. Moreover, the traditional neighbor algorithm was improved by feature weight learning.

Many studies have tried to classify student attentiveness; however, a large amount of the researches focused on qualitative analysis and lacked quantitative analysis. A research group from North Carolina A&T State University (Ross, Graves, Campbell, & Kim, 2013) used machine learning algorithms to classify student attentiveness. Within machine learning, there are two main tasks: supervised and unsupervised learning. (Ross et al., 2013) used SVM, which is a supervised learning method. In addition, a consumer RGB-D sensor collected data from students, who were classified as attentive or inattentive. The results illustrated the final classification by SVM and successfully determined a particular student's learning style.

Many insurance companies use SVM in the underwriting process (Tan & Zhang, 2005) to classify applicants. Before SVM, the data mining association rule algorithm (Oyama & Ishida, 2003) was used to mine insurance company databases. The results gave the idea of the improvement of this algorithm. SVM demonstrated good performance in the experiment and made sense in lowering the premium and reducing the risk of the insured pool as the application of the algorithm went (Tan & Zhang, 2005).

Statistical machine learning concerns the framework of machine learning related to statistics. The application of statistical machine learning had been used in an integrated anti-spam system (Zhang, Su, & Wang, 2007) to filter spam in an flexible, intelligent, self-adaptive and precise way. The algorithms included k-means clustering, linear regression on optimal separating hyperplane, and the improved Naive Bayes. The improved Naive Bayes was used in content analysis layer, and k-means clustering and linear classification were used in the action recognition layer. The applications of these algorithms helped advance the performance of the integrated anti-spam system (Zhang et al., 2007).

When the data was collected, a data analytics platform was necessary to implement the algorithm. A research team from the University of Georgia (Assefi, Behravesh, Liu, & Tafti, 2017) implemented five algorithms to compare Apache Spark machine learning library (MLlib) and Weka. Six data sets (Zhan et al., 2009) were used for the platform test experiment. Two of the data sets were less than 100 megabytes, while the others were 3 gigabytes on average. Each of them were analyzed using the five algorithms, and the running times between the two platforms were recorded for comparison. As a result, Apache Spark MLlib showed better performance and lower processing times (Assefi et al., 2017).

The machine learning methodology was not only used in the field of technology but also in financial analysis. Financial data mining was an key research topic in the field of data mining. Two popular techniques, SVM and ensemble learning, were applied to classify financial data (Lei, Xinming, Lei, & Xiaohong, 2010). The researchers from this paper used two data sets in their experiment: German credit and credit approval data sets. First, accuracy was adopted to analyze classification performance. Then, after SVM and ensemble learning algorithms were implemented, the predictions of different techniques in terms of the accuracy values of the German credit and the credit approval data sets were calculated , and the results showed an obvious improvement in performance (Lei et al., 2010).

2.3 Overview of Ensemble Learning for Big Data Analysis

The use of ensemble methods is expected to enhance the accuracy and reduce the variances of algorithms. However, when comparing ensemble learning and normal machine learning algorithms, several parameter changes sometimes do not affect the results. According to research from Oregon State University (Dietterich, 2000), there were three types of ensemble learningvoting, bagging, and boostingas described below:

- Voting: Voting is the easiest way to create an ensemble, and there are multiple types of voting classifiers. In the training step, all models were trained separately with whole training data, and average posterior probabilities were calculated by each model in the recognition step (Dietterich, 2000, p. 5).

- Bagging: Bagging, also known as bootstrap aggregation, manipulates which training data is used to generate multiple models. It was not necessary to train the model for the whole training data set. Instead, bagging randomly samples the training set from the total training data to make sub-models (Dietterich, 2000, p. 6).

- Boosting: Boosting samples training data like bagging does; however, it maintains a set of weights on the data, especially AdaBoost, which uses the weighted errors of each model update weight on the training data to place more weight on data with lower accuracy levels and less weight on data with higher accuracy levels (Dietterich, 2000, p. 7).

The research group from Hebei University (Li & Hao, 2009) considered the stability of Naive Bayes algorithm by evaluating an ensemble learning algorithm for Naive Bayesian classifiers based on oracle selection and the truth of the limitation of the attributes independence assumption. Naive Bayes classifiers, which generate ensemble methods based on the oracle selection strategy, were presented, and both the selection of the better classifier and the double random process were applied in the algorithm. The experiment showed that generating the ensemble method resulted in better performance than normal Naive Bayes learning and better classification accuracy than AdaBoost and bagging (Li & Hao, 2009).

The research group from (Liu & Wang, 2010) indicated a new class of learning algorithms for a single hidden layer feedforward neural network (SLFN), which was named an extreme learning machine (ELM). Training error was reduced by implementing ELM to achieve good generalization performance. However, neural network classifiers could suffer from overtraining, which might affect the generalization performance. Therefore, an ensemble method based on ELM (EN-ELM) algorithm was proposed, including cross-validation and ensemble learning, and the predictive stability was enhanced and the overtraining problem was alleviated. The experimental results of several benchmark databases indicated that EN-ELM was efficient, although it took more time to train EN-ELM than ELM (Liu & Wang, 2010).

The research group from (Ting & Zheng, 2003) investigated the boosting Naive Bayesian classification to determine whether the boosting method could improve accuracy. The experiments included 25 domains from the UCI machine learning repository (Newman & Merz, 1998). In addition, each domain used 10 threefold cross-validations. Leveled Naive Bayesian Tree (LNBT) and Boosting Leveled Naive Bayesian Trees (BLNBT) were tested in the experiment. The error rate of each algorithm was the main concern in order to evaluate the accuracy of the learning algorithms. According to the results, although boosting improved the accuracy of the Naive Bayesian classifier in 14 out of the 25 domains, accuracy decreased in the other 11 domains. Moreover, the mean relative error reduction of boosting over 25 domains was only 4 percent, which indicated only a slight improvement for boosting. As a result, boosting did not perform very well for Naive Bayesian classification, although it achieved great success with decision trees (Ting & Zheng, 2003).

The faculty from the Department of Bioresource Engineering at McGill University (Belayneh, Adamowski, Khalil, & Quilty, 2015) predicted drought conditions in the Awash River Basin of Ethiopia through reliable artificial neural network (ANN), SVR models with wavelet transform, and the boosting and bootstrap ensemble methods. The models were compared using coefficient of determination (R2), and root mean squared error (RMSE), and mean absolute error (MAE). The performance of wavelet analysis improved drought predictions, and the boosting ensemble method was shown to improve the correlation between predicted and observed parameters. As a result, the wavelet boosting SVR (WBS-SVR) and wavelet boosting ANN (WBS-ANN) models showed better performance in terms of predictions than the other models (Belayneh et al., 2015).

2.4 Overview of Feature Selection

According to (Ryu & Yang, 2018), botnet could be detected by IDS and various machine learning could be applied by IDS. However, ensemble learning for botnet detection had not yet been resolved. Thus, not only were decision tree, Naive Bayes, and neural network evaluated in their study, but ensemble learning based on bagging, voting and boosting were also tested. Based on (Garca, Grill, Stiborek, & Zunino, 2014), the following features were configured in the study: start time, end time, destination port, protocol, number of packets, destination IP, duration, flags, type of services, source port, number of flows, direction, source IP, number of bytes, and label. The confusion matrix was run for the accuracy of classifiers, and Matthews Correlation Coefficient (MCC) and F1 score were evaluated. Precision or recall were used to measure accuracy by MCC and F1 score was used for the imbalanced data. As a result, random tree, an ensemble of multiple decision trees, was determined to be the most reliable approach.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.1}$$

$$MCC = \frac{TP \times TN + FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{2.2}$$

The paper which was written by (Sharafaldin et al., 2018) described different kinds of intrusion detection data sets, and each data set included around 80 features. The data was generated by IDSs and Intrusion Prevention Systems (IPSs) and well-labeled by pre-processing. However, not all of the features were related to each kind of intrusion detection. Thus, the RandomForestRegressor class of the Scikit-Learn library (Abraham et al., 2014) was configured to find the most related feature set to detect each attack, and the features related to each attack were mentioned in this paper. For example, subflow forward bytes, backward packet length, total length of forward packets, and forward packet length mean were related to botnet detection, which is relevant to the present research.

2.5 Summary

This chapter summarized related studies on machine learning algorithms in different areas, including intrusion detection. It also demonstrated that ensemble learning methods enhance predictions. However, the unexpected results still existed where the accuracy would be better if ensemble methods were not used. Furthermore, research on feature selection for network intrusion detection was mentioned. Based on relevant literature, machine learning had achieved great success in big data, and ensemble learning had enhanced predictions in some situations. However, the IP address was usually ignored in network intrusion detection. Therefore, encoding IP address as a feature has not been fully studied. The main focused on my research would be the accuracy scores when encoding IP addresses as a feature of machine learning algorithms for network intrusion detection.

CHAPTER 3. RESEARCH METHODOLOGY

The purpose of this paper was to determine the best method of encoding IP addresses as a feature while predicting different kinds of network attacks by applying machine learning algorithms. The methodology for encoding IP addresses and applying machine learning algorithms were described in this chapter.

3.1 Framework

- Since the databases were collected separately for each attack, they were combined using the concat function because the results focused on detecting different kinds of attacks instead of a unique attack.
- All IP addresses would be converted in three ways and multiple different data sets should be created.
- Three different machine learning algorithms including random forest, SVM, and decision tree were implemented to analyze for the research.
- Python 3.7 would be used throughout the whole research. Additionally, Pandas was imported in the code and mainly used for handling the databases and implementing machine learning algorithms.
- The following specifications were used: hardware environment: an Intel Core i7 CPU, 16GB of RAM, and the macOS operating system.

3.2 Data Pre-processing

3.2.1 Loading and Merging Databases

Each network attack traffic was detected and recorded in separate comma-separated values (CSV) files. Each file contained hundreds of thousands of rows. The CSV files were loaded one at a time using the `read_csv` function. There were seven CSV files, representing different kinds of network attack traffic, so seven individual databases were created.

The structure of each database was the same because the feature numbers and names were exactly the same among all of the databases. To predict different kinds of network attack traffic, the databases had to be combined. Since the structure of each CSV file was the same, the `concat` function could be used to combine the databases. In the end, a large data set was created with 86 features and more than 2 million rows. The features included destination IP, source port, flow duration, source IP, destination port, etc

3.2.2 Converting Attack Label to Numbers

Seven types of attacks were recorded in the databases: DDoS attack, brute force, infiltration, botnet attack, web attack, DoS attack, and PortScan attack. Since the attack type was recorded in the text as a label, the original labels were transformed into numbers. Table 3.1 shows the process of converting the labels.

Table 3.1. *Label*

Label	
Attack Name	**Number**
Benign Network	0
Brute Force	1
Web Attack	2
DoS	3
Infiltration	4
Botnet Attack	5
DDoS	6
PortScan	7
Unknown	-1

3.3 Imbalanced Data Set

After the data sets were combined according to attack and non-attack data, it was evident that the amounts of attack and non-attack data were not balanced. Additionally, too much benign network traffic data had been collected from the original database. The amount of the attack and non-attack data were equated to balance the data set. First, the total attack and non-attack data amounts were calculated separately. The sample function was used to randomly select non-attack data, and the parameter of this function was set to be equal to the total amount of the attack data. Finally, the selected non-attack data was combined with the attack data set.

3.4 Handling the IP Address

The IP address information was one of the most important features in the databases; it is strongly related to network attacks and is collected as an IPv4 address. An IPv4 address is a 32-bit number that uniquely identifies a network interface on a machine, and it is displayed as four numbers and split by three dots. The IPv4 address is one of the core protocols of standards-based internetworking methods in the Internet and other packet-switched networks.

Even though it is displayed as numbers split by three dots, an IPv4 address cannot be easily operated by machine learning algorithms. Therefore, three ways of encoding IP addresses were introduced. Each IP address was split into four numbers and converted into one binary number. Most importantly, one hot encoding of the IP address was applied.

This research involved two kinds of IP addresses: source IPs and destination IPs. When a device initiates communication with servers or other devices, its IP address is called a source IP, which also sends IP packets. The device's IP address that receives the packets is called a destination IP.

3.4.1 Converting the IP Addresses to Binary Numbers

IPv4 uses 32-bit addresses, which consist of four 8-bit addresses displayed separately by dots. The first method used to convert the IP addresses was to split the 32-bit address into four 8-bit binary numbers and then directly combine them as one feature in the data set. Both the destination and source IP and were converted. Figure 3.1 shows the process of converting IP addresses to binary integers.

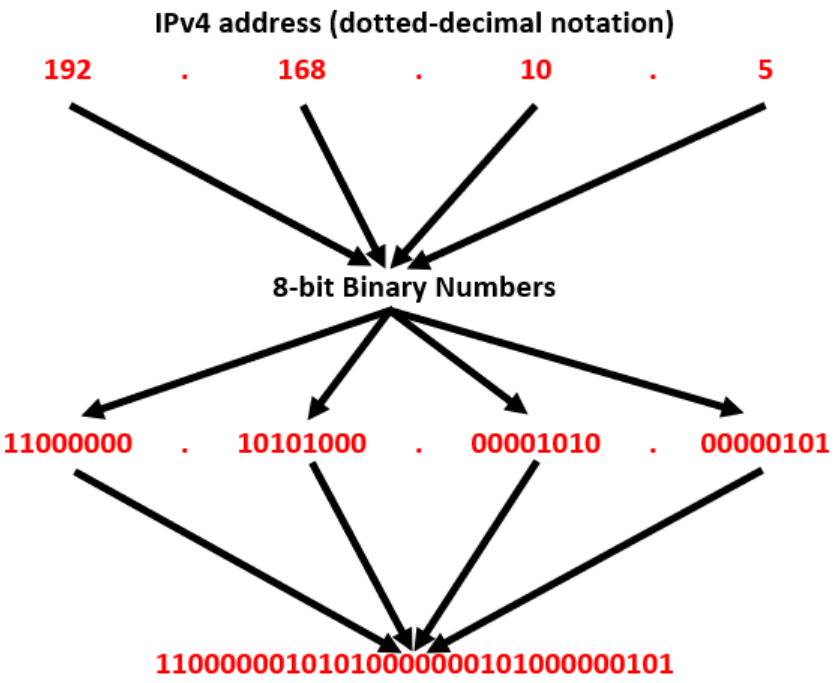

Figure 3.1. IP Address to Binary Numbers

3.4.2 Splitting an IP Address to Four Numbers

The second method used to convert the IP addresses was to split the 32-bit address into four separate numbers. These four numbers were considered as four individual features. After converting the source IP and the destination IP, these IP address became eight different features in the database. Figure 3.2 shows the process of splitting an IP address.

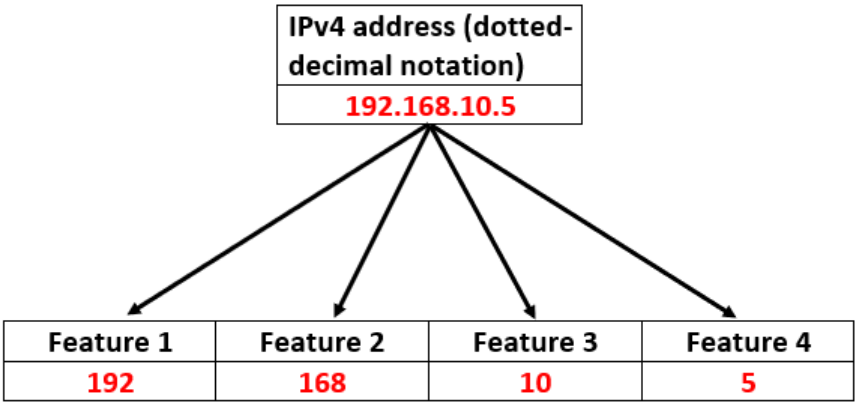

Figure 3.2. Split IP Address to 4 Numbers

3.4.3 One Hot Encoding for IP Addresses

One hot encoding is a method of converting categorical variables into a binary matrix format, which can provide better predictions for machine learning algorithms. In digital circuits and machine learning, one hot encoding consists of legal combinations of a single high bit and all other low bits. High bits and low bits are binary-like numbers, which are 0 and 1. Table 3.2 gives an example of the results of one hot encoding.

Both source IPs and destination IPs were considered as categorical values for which one hot encoding was applied separately. IP addresses were created using 32-bit numbers; the first 16-bit number was a network address and the second was a host address. One hot encoding was applied for three IP address sizes (16-bit, 24-bit, and 32-bit).

There were many ways to apply one hot encoding. In this research, the function of `LabelBinarizer` from `Scikit-Learn` library was applied.

Table 3.2. *One Hot Encoding*

One Hot Encoding								
Value 1	**Value 2**	**Value 3**	**Value 4**	**Value 5**	**Value 6**	**Value 7**	**Value 8**	**...**
1	0	0	0	0	0	0	0	...
0	1	0	0	0	0	0	0	...
0	0	1	0	0	0	0	0	...
0	0	0	1	0	0	0	0	...
0	0	0	0	1	0	0	0	...
0	0	0	0	0	1	0	0	...
0	0	0	0	0	0	1	0	...
0	0	0	0	0	0	0	1	...
...	...	...	...	...	...	...	...	...

3.4.3.1 Encoding a 32-bit IP Address

When applying one hot encoding to a 32-bit IP address, the entire IP address was encoded. The 32-bit IP address was considered as a categorical value representing the numerical value in the database. The representation of the 32-bit IP address began from 1 to N-1 categories. Finally, the binary matrix was created by applying the `LabelBinarizer` function. Figure 3.3 gives an explanation of how to use one hot encode for a 32-bit IP address.

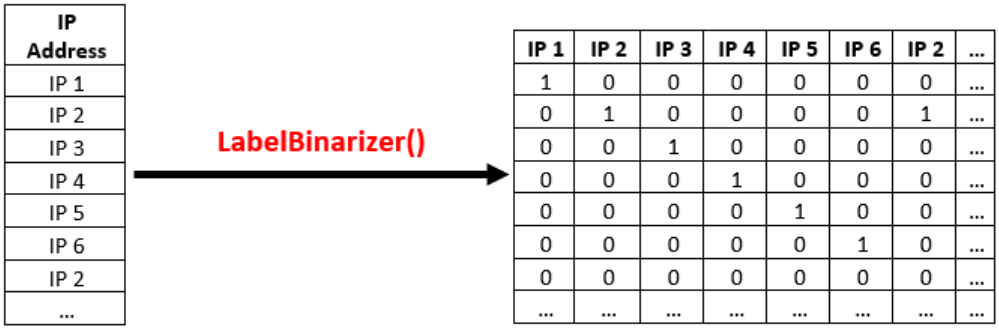

Figure 3.3. Encoding 32-bit IP address

3.4.3.2 Encoding a 24-bit IP Address

As each IP address was collected as an IPv4 address, it consisted of four 8-bit numbers. The last 8-bit number was ignored, leaving a 24-bit address, which indicated the network and the subnet. The process for encoding the 24-bit IP address was similar to the 32-bit IP address except for the encoding object. Figure 3.4 explains what the 24-bit IP address looks like and figure 3.5 explains the process for encoding 24-bit IP addresses.

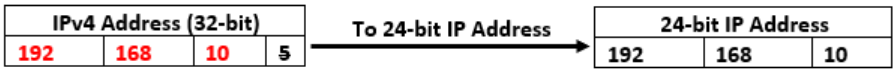

Figure 3.4. 24-bit IP Address

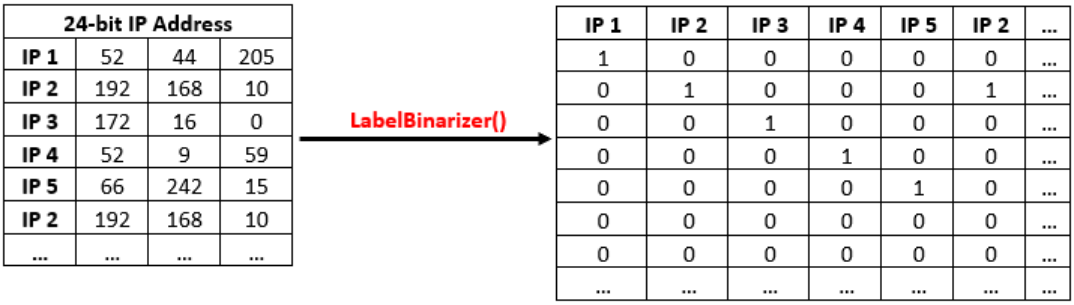

Figure 3.5. Encoding 24-bit IP address

3.4.3.3 Encoding a 16-bit IP Address

As noted above, an IP address consists of four 8-bit numbers, which also represent the host and the network address. For the 16-bit IP address encoding, the first two 8-bit numbers, which represent the network address, were considered for one hot encoding, and the process was the same as those described above. Figure 3.6 explains what 16-bit IP addresses look like and figure 3.7 shows the process of encoding 16-bit IP addresses.

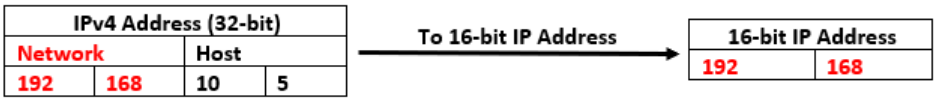

Figure 3.6. 16-bit IP Address

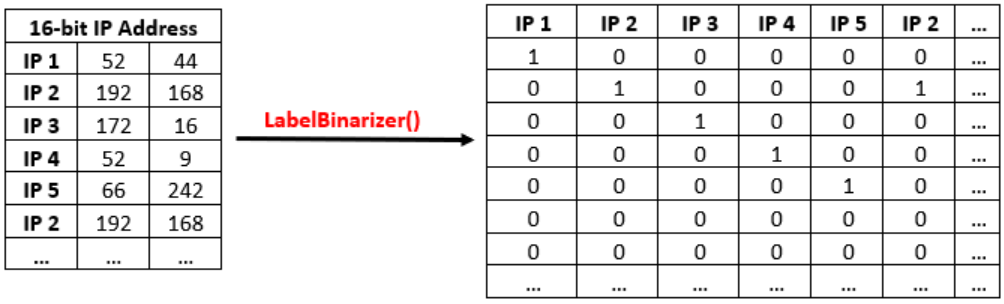

Figure 3.7. Encoding 16-bit IP address

3.5 Selecting Necessary Feature and Implementing Machine Learning Algorithms

The data set was made up of more than 2 million rows and 86 features, and not all of the features were necessary for predicting network attack traffic (Sharafaldin et al., 2018). Since the IP addresses had already been considered for encoding, source ports and destination ports associated with the source IPs and destination IPs were selected. Ports are communication endpoints of the identified protocol and address, commonly represented by port numbers. Port numbers can also identify network service or specific processes. In total, 21 features were chosen for predicting network attacks, including all features related to destination port, flow duration, flags, source port, destination IP, flow duration, flags, source IP, and packets. (Sharafaldin et al., 2018)

Before implementing the classification machine learning algorithm (SVM, decision tree, and random forest), the data set was split into training and testing data sets using stratified sampling. Specifically, 80 percent of the data set was selected as the training data set and 20 percent was selected as the testing data set. The Scikit-Learn library was utilized to implement the machine learning algorithms, and the package of the algorithm was called for predicting network attack traffic.

3.5.1 SVM

SVM is a classification algorithm, and it is mainly used for binary classification. The data is usually separated into two classes of data points and then considered geometrically to find the classifier. Figure 3.8 shows a simple example of finding the classification. The two classes of data were black and red points and the blue line was the classifier.

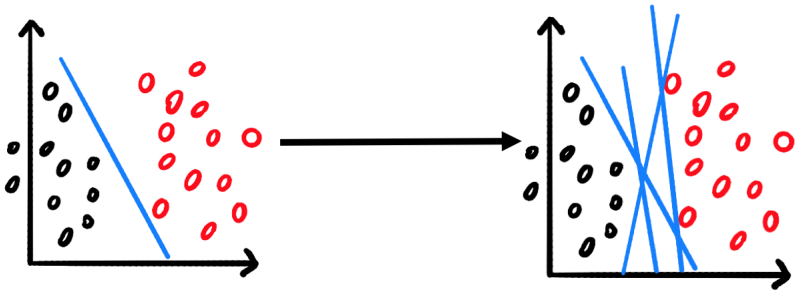

Figure 3.8. SVM Classifiers

After finding the classifiers, many hyperplanes could be chosen. The objective of SVM is to define a hyperplane that could uniquely classify the data. The hyperplane should have the maximum margin, which is the maximum distance between both classes of data points. In general, an optimal hyperplane that categorized new examples was found based on the training data. Figure 3.9 shows what the maximum margin hyperplane would look like after finding classifiers.

In the present study, the SVM model was called from the Scikit-Learn library for implementation. Since a classification task was performed, the support vector classifier (SVC) class was imported into the program. The fit method from the SVC class was called to train the data set, and the predict method was used to make predictions.

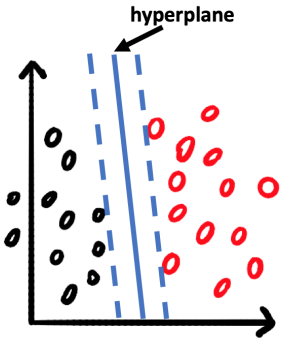

Figure 3.9. SVM Hyperplane

3.5.2 Decision Tree

A decision tree, also known as classification tree, is a tree-based algorithm, and it is considered one of the most popular supervised learning methods. The concept of the decision tree model is very simple, and it has three main components: the model of decisions, a tree-like graph, and possible consequences using if-then and yes-no logic. Figure 3.10 represents a decision tree model. The tree structure brakes the data set into different subsets. The root node is the first decision node and represents the entire sample. The decision node partitions the data, and it is split sub-node by sub-node. Eventually, predictions of the problem are given by leaf nodes, and the leaf nodes cannot be further split.

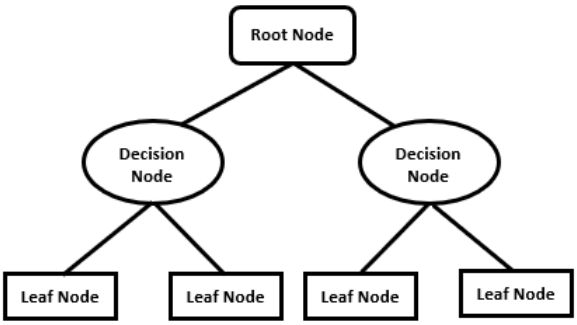

Figure 3.10. Decision Tree

The default classifier model of the decision tree was used in the present thesis. The tree package was imported from Scikit-Learn, the fit method from the DecisionTreeClassifier class was called to train the data, and the predict method was used to make predictions. There were many parameters in the function, some of which were set manually to make the process more efficient.

3.5.3 Random Forest

Random Forest is operated by building a large number of unique decision trees, and it is an ensemble learning method of decision tree algorithms. There are three main types of ensemble learning: bagging, boosting and stacking. The random forest training process normally applies bagging to the decision tree. Bagging is another name for bootstrap aggregating and is often used to reduce variances.

Random forests mainly consist of four steps. First, the algorithms randomly select samples from a data set. Second, a decision tree is created for each sample, and a prediction is generated from each decision tree. After that, voting occurs for each prediction result. The prediction of the random forest is the class with most votes based on the class prediction, which is split from each decision tree created in the random forest. Figure 3.11 gives an example of how random forest is used for a prediction.

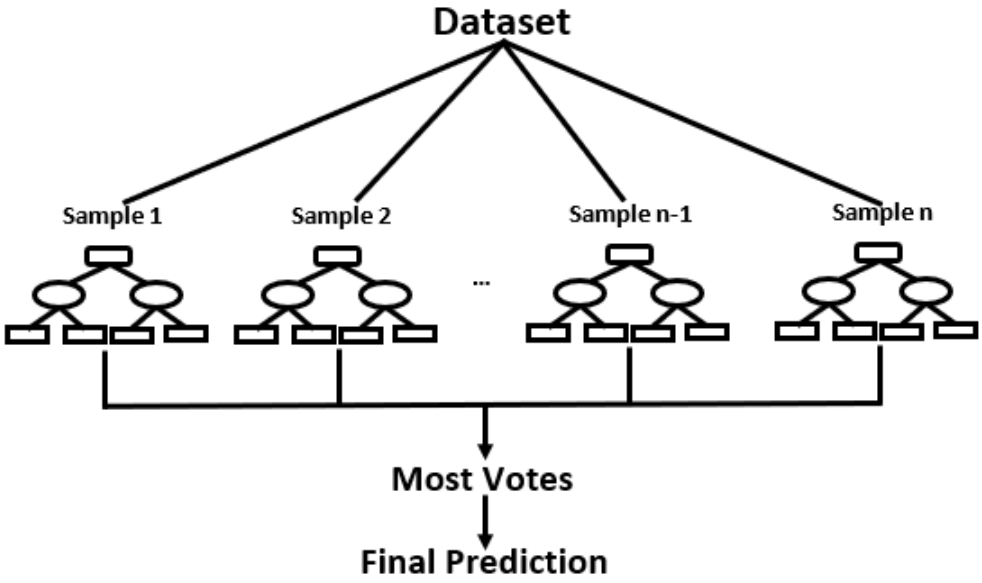

Figure 3.11. Random Forest

To implement classification random forest in the present study, the RandomForestClassifier function was imported from the Scikit-Learn library, and most parameters of random forest were the same as the decision tree. The fit method from RandomForestClassifier class was called to train the data, and the predict method was used to make predictions.

3.6 Summary

As described in this chapter, the present study integrated three machine learning algorithms and converted the IP addresses into binary numbers. In addition, the methods of encoding IP addresses were evaluated by the three machine learning algorithms. The results of the different methods will be discussed and compared in the next chapter.

CHAPTER 4. RESULTS

As described above, three different ways for converting IPv4 addresses and one hot encoding, converting to binary numbers, and splitting into four numbers were used in the present study. Moreover, the IP addresses should include both benign traffic's IP addresses and malicious IP addresses because some malicious IP addresses indicated benign traffic when malicious tasks were not assigned. This chapter describes the results of these conversion methods, as evaluated by SVM, the decision tree and the random forest. In addition, comparisons of the results were conducted to determine the optimal method for predicting network attack traffic.

It is important to note that, when one hot encoding was applied for both source IPs and destination IPs, a huge binary matrix was created. Due to RAM limitations, there would have been memory errors if the entire data set was trained with one hot encoded IPs. Therefore, to fit the system environment, part of the data set was randomly chosen.

4.1 One Hot Encoding vs. Binary IP vs. Split IP

In this section, the results for the binary, split, and one hot encoded IPs are compared, with the precision, recall, F-1 score, and accuracy scores shown in the accompanying tables. The accuracy scores were mainly evaluated through the three machine learning algorithms, and the goal was to determine which method of encoding IP addresses was the best for network intrusion detection.

4.1.1 32-bit IP Addresses

Both source and destination IPs were collected as 32-bit addresses for one hot encoding, but not all data was read by the machine learning algorithms. Due to the RAM limitations, only 10 percent of the data was used. To gather this 10 percent, the entire data set could not be randomly selected because the amount of data from each type of attack was not the same. In other words, the data set would have become imbalanced again if stratified sampling was used for the entire data set.

As a result, the 10 percent needed to be collected separately from each attack type. Therefore, 10 percent was randomly selected from the DDoS attack traffic data, as well as from the other attacks. After 10 percent of each attack data set had been collected, they were combined, and the new data set contained about 10 percent of the entire original data set. Then, the machine learning algorithms were applied to that data set.

All three methods of encoding IP addresses were applied, and the three machine learning algorithms were used to evaluate each method of encoding the IP addresses. The results are shown in the next nine tables.

Table 4.1. *Results of Decision Tree for One Hot Encoding (32-bit)*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	1.00	1.00
Brute Force	1.00	0.71	0.83
Web Attack	0.78	0.88	0.85
DoS	0.37	0.10	0.16
Infiltration	1.00	1.00	1.00
Botnet Attack	0.99	1.00	0.99
DDoS	0.46	0.46	0.46
PortScan	0.87	1.00	0.93
Unknown Traffic	0.42	1.00	0.59
Weighted Average	**0.88**	**0.88**	**0.87**
Accuracy	**0.8759**		

Table 4.2. *Results of Random Forest for One Hot Encoding (32-bit)*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.93	0.96
Brute Force	0.99	1.00	1.00
Web Attack	0.93	0.96	0.95
DoS	0.77	0.96	0.86
Infiltration	0.11	1.00	0.19
Botnet Attack	0.35	0.99	0.51
DDoS	0.97	0.86	0.91
PortScan	1.00	0.99	1.00
Unknown Traffic	0.26	1.00	0.42
Weighted Average	**0.97**	**0.95**	**0.95**
Accuracy	**0.9455**		

Table 4.3. *Results of SVM for One Hot Encoding (32-bit)*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.94	0.97
Brute Force	0.96	1.00	0.98
Web Attack	0.98	0.67	0.80
DoS	0.15	0.07	0.09
Infiltration	1.00	0.75	0.86
Botnet Attack	0.91	1.00	0.95
DDoS	0.71	1.00	0.83
PortScan	0.93	0.98	0.95
Unknown Traffic	1.00	1.00	1.00
Weighted Average	**0.92**	**0.90**	**0.90**
Accuracy	**0.9038**		

Table 4.4. *Results of Decision Tree for Binary IP (10 %)*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.99	0.84	0.91
Brute Force	0.97	1.00	0.98
Web Attack	0.85	0.76	0.80
DoS	0.22	0.99	0.36
Infiltration	0.03	0.40	0.05
Botnet Attack	0.96	1.00	0.98
DDoS	0.63	0.53	0.57
PortScan	0.82	0.98	0.90
Unknown Traffic	0.42	1.00	0.59
Weighted Average	**0.89**	**0.84**	**0.85**
Accuracy	**0.8363**		

Table 4.5. *Results of Random Forest for Binary IP (10 %)*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.85	0.92
Brute Force	0.99	1.00	1.00
Web Attack	0.89	0.86	0.87
DoS	0.72	0.87	0.79
Infiltration	0.07	0.50	0.12
Botnet Attack	0.62	0.99	0.76
DDoS	0.69	1.00	0.82
PortScan	0.92	0.99	0.95
Unknown Traffic	0.83	1.00	0.91
Weighted Average	**0.93**	**0.90**	**0.91**
Accuracy	**0.9049**		

Table 4.6. *Results of SVM for Binary IP (10 %)*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.87	0.87	0.87
Brute Force	0.90	1.00	0.95
Web Attack	0.69	0.78	0.73
DoS	0.00	0.00	0.00
Infiltration	0.00	0.00	0.00
Botnet Attack	0.30	0.03	0.05
DDoS	0.57	0.44	0.50
PortScan	0.89	0.99	0.94
Unknown Traffic	0.00	0.00	0.00
Weighted Average	**0.79**	**0.81**	**0.80**
Accuracy	**0.8132**		

Table 4.7. *Results of Decision Tree for Split IP (10 %)*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.92	0.96
Brute Force	1.00	0.99	1.00
Web Attack	0.99	0.98	0.99
DoS	0.89	0.98	0.99
Infiltration	0.03	0.75	0.06
Botnet Attack	0.36	0.98	0.52
DDoS	0.99	0.99	0.99
PortScan	1.00	1.00	1.00
Unknown Traffic	0.00	1.00	0.00
Weighted Average	**0.99**	**0.96**	**0.97**
Accuracy	**0.9645**		

Table 4.8. *Results of Random Forest for Split IP (10 %)*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.98	0.99
Brute Force	0.99	1.00	0.99
Web Attack	0.83	0.95	0.88
DoS	0.63	0.90	0.74
Infiltration	0.37	0.88	0.52
Botnet Attack	0.72	0.98	0.83
DDoS	0.99	0.81	0.89
PortScan	1.00	1.00	1.00
Unknown Traffic	0.50	1.00	0.67
Weighted Average	**0.96**	**0.95**	**0.95**
Accuracy	**0.9501**		

Table 4.9. *Results of SVM for Split IP (10 %)*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.97	0.99	0.98
Brute Force	0.94	1.00	0.97
Web Attack	1.00	0.67	0.80
DoS	0.21	0.04	0.07
Infiltration	1.00	0.12	0.22
Botnet Attack	0.73	0.13	0.23
DDoS	0.73	1.00	0.85
PortScan	0.99	0.99	0.99
Unknown Traffic	0.05	0.50	0.08
Weighted Average	**0.92**	**0.91**	**0.90**
Accuracy	**0.9140**		

The accuracy scores of the three methods of encoding 32-bit IP addresses were compared, and Figure 4.1 shows the accuracy of each algorithm. Although the method of one hot encoding had better results than binary IP, the method of splitting an IP address into four numbers was even better. Therefore, splitting IP addresses into four numbers achieved the highest accuracy scores.

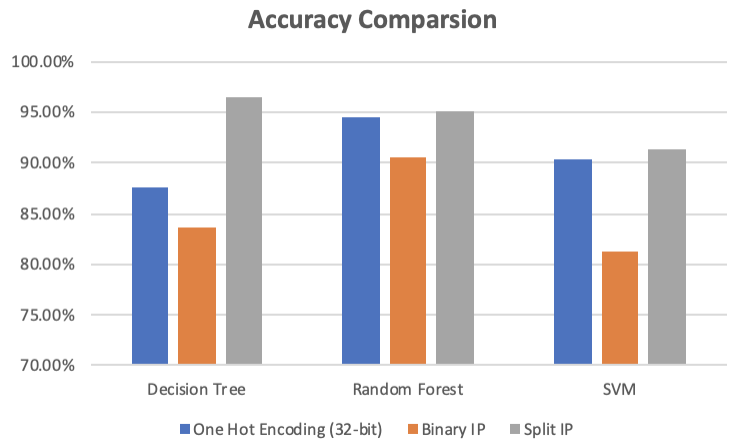

Figure 4.1. One Hot Encoding (32-bit) vs Binary IP vs Split IP

4.1.2 24-bit IP Addresses

This section covers the 24-bit IP address collected for one hot encoding. As described above, only the first three numbers of the IP address were used. As with the 32-bit IP addresses, due to the RAM limitations, not all of the data was used to avoid memory errors. In this case, 20 percent of the data was taken from each attack type. The reason why more data was taken was that the dimensions of one hot encoding for 24-bit IP addresses were lower than for 32-bit IP addresses. Therefore, to make the results more accurate, more data was collected.

Eventually, the machine learning algorithms were used to evaluate all three methods of encoding IP addresses. The following nine tables represent the results.

Table 4.10. *Results of Decision Tree for One Hot Encoding (24-bit)*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.91	0.95
Brute Force	1.00	0.99	1.00
Web Attack	1.00	0.99	1.00
DoS	0.99	0.90	0.95
Infiltration	0.61	0.99	0.75
Botnet Attack	0.00	0.57	0.00
DDoS	0.69	1.00	0.82
PortScan	0.87	0.98	0.92
Unknown Traffic	0.05	1.00	0.10
Weighted Average	**0.98**	**0.93**	**0.95**
Accuracy	**0.9317**		

Table 4.11. *Results of Random Forest for One Hot Encoding (24-bit)*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.92	0.96
Brute Force	1.00	1.00	1.00
Web Attack	1.00	0.98	0.99
DoS	0.71	0.99	0.83
Infiltration	0.01	0.71	0.03
Botnet Attack	0.25	0.98	0.40
DDoS	1.00	1.00	1.00
PortScan	1.00	1.00	1.00
Unknown Traffic	1.00	1.00	1.00
Weighted Average	**0.99**	**0.96**	**0.97**
Accuracy	**0.9594**		

Table 4.12. *Results of SVM for One Hot Encoding (24-bit)*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.99	0.99	0.99
Brute Force	0.90	0.99	0.95
Web Attack	0.78	0.96	0.86
DoS	0.50	0.04	0.07
Infiltration	1.00	0.14	0.25
Botnet Attack	1.00	0.40	0.57
DDoS	0.75	0.45	0.56
PortScan	0.98	0.98	0.98
Unknown Traffic	0.67	1.00	0.80
Weighted Average	**0.91**	**0.91**	**0.90**
Accuracy	**0.9113**		

Table 4.13. *Results of Decision Tree for Binary IP (20 %)*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.99	0.91	0.95
Brute Force	0.97	0.99	0.98
Web Attack	0.87	0.89	0.88
DoS	0.52	0.91	0.66
Infiltration	0.02	0.71	0.04
Botnet Attack	0.58	1.00	0.73
DDoS	0.81	0.94	0.87
PortScan	0.98	0.99	0.98
Unknown Traffic	0.20	1.00	0.33
Weighted Average	**0.94**	**0.92**	**0.93**
Accuracy	**0.9218**		

Table 4.14. *Results of Random Forest for Binary IP (20 %)*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.91	0.95
Brute Force	0.98	1.00	0.99
Web Attack	0.90	0.84	0.87
DoS	0.57	0.87	0.69
Infiltration	0.50	0.86	0.63
Botnet Attack	0.35	1.00	0.51
DDoS	0.70	0.92	0.79
PortScan	1.00	0.99	1.00
Unknown Traffic	0.50	0.91	0.67
Weighted Average	**0.94**	**0.91**	**0.92**
Accuracy	**0.9135**		

Table 4.15. *Results of SVM for Binary IP (20 %)*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.88	0.87	0.88
Brute Force	0.88	0.98	0.93
Web Attack	0.67	0.76	0.71
DoS	0.00	0.00	0.00
Infiltration	0.00	0.00	0.00
Botnet Attack	0.41	0.03	0.05
DDoS	0.67	0.46	0.54
PortScan	0.87	1.00	0.93
Unknown Traffic	0.00	0.00	0.00
Weighted Average	**0.80**	**0.81**	**0.80**
Accuracy	**0.8135**		

Table 4.16. *Results of Decision Tree for Split IP (20 %)*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.91	0.95
Brute Force	1.00	0.99	1.00
Web Attack	1.00	0.99	0.99
DoS	0.75	0.99	0.85
Infiltration	0.03	0.57	0.06
Botnet Attack	0.19	1.00	0.31
DDoS	1.00	1.00	1.00
PortScan	1.00	1.00	1.00
Unknown Traffic	0.15	1.00	0.27
Weighted Average	**0.99**	**0.95**	**0.97**
Accuracy	**0.9516**		

Table 4.17. *Results of Random Forest for Split IP (20 %)*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.99	0.99
Brute Force	1.00	1.00	1.00
Web Attack	1.00	0.86	0.92
DoS	0.66	0.95	0.78
Infiltration	0.67	0.86	0.75
Botnet Attack	0.61	1.00	0.76
DDoS	0.79	1.00	0.88
PortScan	1.00	0.99	1.00
Unknown Traffic	1.00	1.00	1.00
Weighted Average	**0.97**	**0.96**	**0.96**
Accuracy	**0.9614**		

Table 4.18. *Results of SVM for Split IP (20 %)*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.98	1.00	0.16
Brute Force	0.95	0.99	0.97
Web Attack	0.79	0.97	0.87
DoS	0.40	0.04	0.07
Infiltration	1.00	0.14	0.25
Botnet Attack	0.36	0.02	0.03
DDoS	0.79	0.46	0.58
PortScan	0.99	0.99	0.99
Unknown Traffic	0.09	1.00	0.16
Weighted Average	**0.96**	**0.95**	**0.95**
Accuracy	**0.9501**		

Figure 4.2 depicts a comparison chart of the accuracy of the three methods of encoding the 24-bit IP addresses. Splitting the IP address into four numbers achieved the highest accuracy scores, while binary IP resulted in the lowest accuracy. Although, the random forest accuracy scores for the one hot encoding and split IP methods were very close, there was still a slight improvement for the split IP method. As a result, splitting an IP address was again considered the best method for encoding IP addresses.

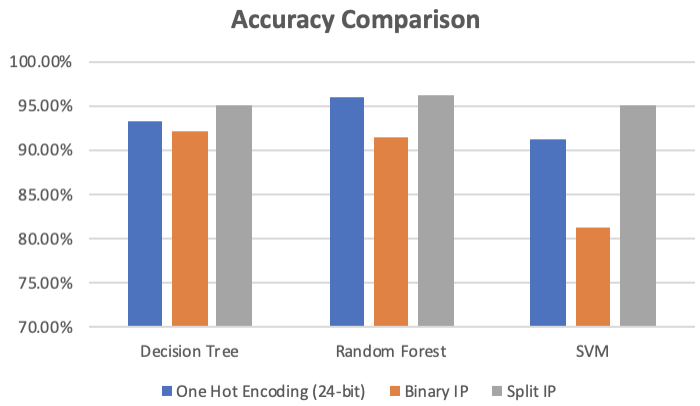

Figure 4.2. One Hot Encoding (24-bit) vs Binary IP vs Split IP

4.1.3 16-bit IP Address

As described above, an IPv4 address is a 32-bit number that uniquely identifies the network and the host. The first two 16-bit numbers, which represent the network portion, were encoded. Although the size of the IP address was smaller in this case, the entire data still could not fit the memory limitations; therefore 50 percent of the data was selected for a new database. The process of selecting the data by attack type was the same as the method described in the previous sections. After collecting data from each data type, they were combined. Then, the random forest, SVM, decision tree, evaluations were conducted. The evaluation results and comparisons are shown in below.

Table 4.19. *Results of Decision Tree for One Hot Encoding (16-bit)*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.88	0.94
Brute Force	1.00	1.00	1.00
Web Attack	1.00	0.91	0.95
DoS	0.19	1.00	0.32
Infiltration	0.01	0.67	0.01
Botnet Attack	0.07	1.00	0.13
DDoS	0.86	1.00	0.92
PortScan	1.00	0.90	0.94
Unknown Traffic	0.15	1.00	0.26
Weighted Average	**0.98**	**0.91**	**0.94**
Accuracy	**0.9081**		

Table 4.20. *Results of Random Forest for One Hot Encoding (16-bit)*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.93	0.96
Brute Force	1.00	1.00	1.00
Web Attack	0.97	1.00	0.99
DoS	0.99	0.90	0.94
Infiltration	0.34	0.96	0.50
Botnet Attack	0.01	1.00	0.01
DDoS	0.14	0.98	0.24
PortScan	0.98	0.98	0.92
Unknown Traffic	0.99	1.00	1.00
Weighted Average	**0.98**	**0.94**	**0.96**
Accuracy	**0.9393**		

Table 4.21. *Results of SVM for One Hot Encoding (16-bit)*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.99	0.99
Brute Force	1.00	1.00	1.00
Web Attack	0.89	0.99	0.94
DoS	0.79	0.97	0.87
Infiltration	0.89	0.03	0.07
Botnet Attack	0.00	0.00	0.00
DDoS	1.00	0.38	0.55
PortScan	0.87	0.45	0.60
Unknown Traffic	0.98	1.00	0.99
Weighted Average	**0.93**	**0.93**	**0.92**
Accuracy	**0.9255**		

Table 4.22. *Results of Decision Tree for Binary IP (50 %)*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.99	0.88	0.93
Brute Force	0.69	0.99	0.81
Web Attack	0.87	0.85	0.86
DoS	0.55	0.93	0.69
Infiltration	0.01	0.67	0.01
Botnet Attack	0.86	0.98	0.92
DDoS	0.84	0.97	0.90
PortScan	0.94	1.00	0.97
Unknown Traffic	0.00	1.00	0.00
Weighted Average	**0.93**	**0.90**	**0.91**
Accuracy	**0.9027**		

Table 4.23. *Results of Random Forest for Binary IP (50 %)*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.81	0.89
Brute Force	0.94	1.00	0.97
Web Attack	0.97	0.91	0.94
DoS	0.14	0.93	0.24
Infiltration	0.01	1.00	0.03
Botnet Attack	0.07	0.89	0.12
DDoS	0.77	1.00	0.87
PortScan	0.98	0.99	0.99
Unknown Traffic	0.43	1.00	0.60
Weighted Average	**0.96**	**0.88**	**0.91**
Accuracy	**0.8834**		

Table 4.24. *Results of SVM for Binary IP (50 %)*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.90	0.91	0.90
Brute Force	0.96	0.51	0.67
Web Attack	0.73	0.85	0.79
DoS	0.00	0.00	0.00
Infiltration	0.00	0.00	0.00
Botnet Attack	0.29	0.04	0.06
DDoS	0.70	0.45	0.55
PortScan	0.88	0.99	0.93
Unknown Traffic	0.40	0.67	0.50
Weighted Average	**0.83**	**0.84**	**0.83**
Accuracy	**0.8402**		

Table 4.25. *Results of Decision Tree for Split IP (50 %)*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.98	0.99
Brute Force	1.00	0.99	1.00
Web Attack	0.98	0.93	0.95
DoS	0.82	0.99	0.90
Infiltration	0.01	1.00	0.02
Botnet Attack	0.50	1.00	0.02
DDoS	0.87	0.96	0.91
PortScan	0.99	1.00	1.00
Unknown Traffic	0.05	1.00	1.00
Weighted Average	**0.98**	**0.97**	**0.97**
Accuracy	**0.9678**		

Table 4.26. *Results of Random Forest for Split IP (50 %)*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.98	0.99
Brute Force	0.90	1.00	0.95
Web Attack	1.00	0.95	0.78
DoS	0.42	0.92	0.58
Infiltration	0.28	1.00	0.44
Botnet Attack	0.30	1.00	0.46
DDoS	0.91	1.00	0.95
PortScan	1.00	0.99	1.00
Unknown Traffic	1.00	1.00	1.00
Weighted Average	**0.98**	**0.97**	**0.98**
Accuracy	**0.9719**		

Table 4.27. *Results of SVM for Split IP (50 %)*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.99	1.00	1.00
Brute Force	0.92	0.99	0.95
Web Attack	0.79	0.97	0.87
DoS	0.12	0.0.	0.05
Infiltration	0.00	0.00	0.00
Botnet Attack	0.21	0.01	0.02
DDoS	0.87	0.45	0.59
PortScan	0.99	1.00	0.99
Unknown Traffic	0.09	1.00	0.16
Weighted Average	**0.92**	**0.93**	**0.92**
Accuracy	**0.9251**		

Figure 4.3 shows the comparison of the accuracy scores among all three machine learning algorithms. Notwithstanding the fact that the accuracy score of the one hot encoding of SVM was a little bit higher than that of split IP, the difference was only 0.04 percent, and the method of splitting an IP address into four numbers had higher accuracy scores than the other two machine learning algorithms. As a result, the method of splitting an IP address into four numbers mostly received the highest accuracy scores again.

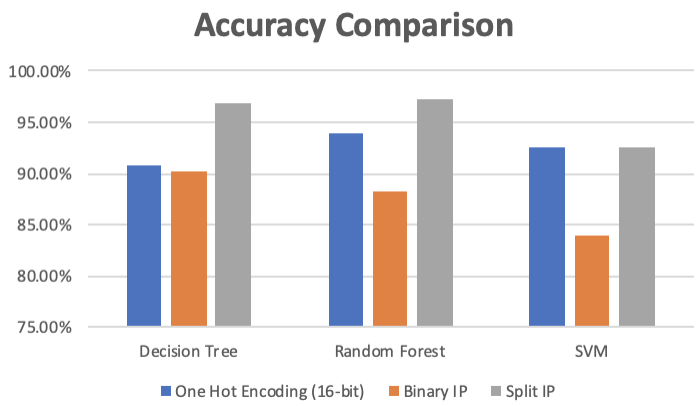

Figure 4.3. One Hot Encoding (16-bit) vs Binary IP vs Split

4.1.4 Summary

In this section, all three methods of encoding IP addresses were evaluated using decision tree, random forest, and SVM, and the split IP method mostly achieved the highest accuracy scores. Therefore, it can be determined that the best method of encoding an IP address is to split an IP address into four numbers.

4.2 Binary IP vs. Split IP

In the previous section, one hot encoded IPs were compared with the other two methods. Although the one hot encoded IPs had better results than those of the binary IPs, the split IPs had better accuracy scores than the one hot encoded IPs. To confirm the best method of encoding IP addresses among these methods, the accuracy of the binary IPs and split IPs were evaluated. Since the dimensions of the data set were not very high for these two methods, the entire data set was considered. The following six tables show the results of this analysis.

Table 4.28. *Results of Decision Tree for Binary IP*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.99	0.88	0.94
Brute Force	0.89	1.00	0.94
Web Attack	0.81	0.87	0.84
DoS	0.08	0.99	0.16
Infiltration	0.00	0.60	0.10
Botnet Attack	0.13	0.98	0.23
DDoS	0.65	0.72	0.28
PortScan	0.97	0.87	0.92
Unknown Traffic	0.20	1.00	0.33
Weighted Average	**0.91**	**0.86**	**0.88**
Accuracy	**0.8632**		

Table 4.29. *Results of Random Forest for Binary IP*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.90	0.95
Brute Force	0.58	1.00	0.74
Web Attack	0.87	0.89	0.88
DoS	0.19	0.87	0.31
Infiltration	0.02	0.80	0.05
Botnet Attack	0.64	0.98	0.77
DDoS	0.81	1.00	0.90
PortScan	1.00	0.99	1.00
Unknown Traffic	0.50	1.00	0.67
Weighted Average	**0.94**	**0.92**	**0.93**
Accuracy	**0.9247**		

Table 4.30. *Results of SVM for Binary IP*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	0.91	0.90	0.90
Brute Force	1.00	0.29	0.45
Web Attack	0.72	0.82	0.77
DoS	0.00	0.00	0.00
Infiltration	0.00	0.00	0.00
Botnet Attack	0.00	0.00	0.00
DDoS	0.69	0.45	0.55
PortScan	0.88	1.00	0.93
Unknown Traffic	0.00	0.00	0.00
Weighted Average	**0.83**	**0.84**	**0.83**
Accuracy	**0.8377**		

Table 4.31. *Results of Decision Tree for Split IP*

Decision Tree			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.91	0.96
Brute Force	0.99	1.00	0.99
Web Attack	1.00	1.00	1.00
DoS	0.83	0.92	0.87
Infiltration	0.00	0.70	0.00
Botnet Attack	0.07	0.99	0.13
DDoS	1.00	1.00	1.00
PortScan	1.00	1.00	1.00
Unknown Traffic	0.43	1.00	1.00
Weighted Average	**1.00**	**0.96**	**0.97**
Accuracy	**0.9562**		

Table 4.32. *Results of Random Forest for Split IP*

Random Forest			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	0.97	0.99
Brute Force	0.90	1.00	0.95
Web Attack	1.00	0.91	0.95
DoS	0.25	0.87	0.39
Infiltration	0.04	0.70	0.07
Botnet Attack	0.14	1.00	0.24
DDoS	0.87	1.00	0.93
PortScan	1.00	0.99	0.99
Unknown Traffic	0.75	1.00	0.86
Weighted Average	**0.98**	**0.96**	**0.97**
Accuracy	**0.9631**		

Table 4.33. *Results of SVM for Split IP*

SVM			
	Precision	**Recall**	**F-1 Score**
Benign Network	1.00	1.00	1.00
Brute Force	0.89	1.00	0.94
Web Attack	0.79	0.97	0.87
DoS	0.28	0.03	0.06
Infiltration	0.00	0.00	0.00
Botnet Attack	0.25	0.00	0.01
DDoS	0.93	0.45	0.61
PortScan	0.98	1.00	0.99
Unknown Traffic	0.14	1.00	0.24
Weighted Average	**0.93**	**0.92**	**0.92**
Accuracy	**0.9296**		

Figures 4.4 indicates the results of the binary and split IPs. The accuracy scores of the split IPs among the three machine learning algorithms were higher than those of the binary IPs.

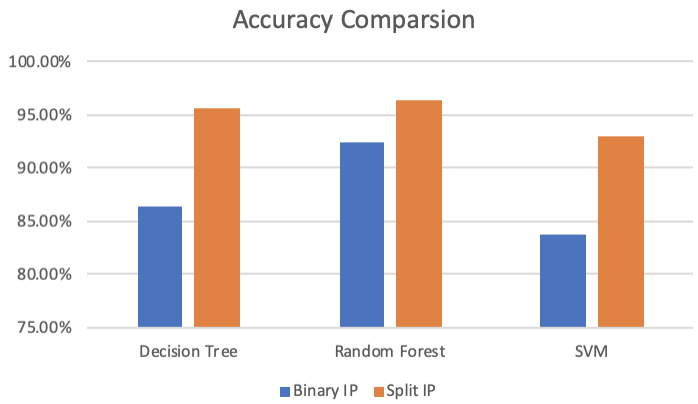

Figure 4.4. Binary IP vs Split IP

4.3 Attack Column

In this section, the F-1 scores of each attack by applying decision tree, random forest and SVM were evaluated based on the tables in section 4.2. The goal was to determine how each machine learning algorithm was performed for each kind of attack.

The chart for F-1 scores by each attack type was displayed in the following and each bar of the graph represented the F-1 scores of each method of encoding IP addresses by applying the machine learning algorithm.

Figure 4.5 showed the F-1 scores for detecting the brute force attack. It was determined that most F-1 scores were above 80 percent. Moreover, the F-1 scores of random forest were all above 90 percent and random forest might be a great machine learning algorithm for detecting the brute force attack.

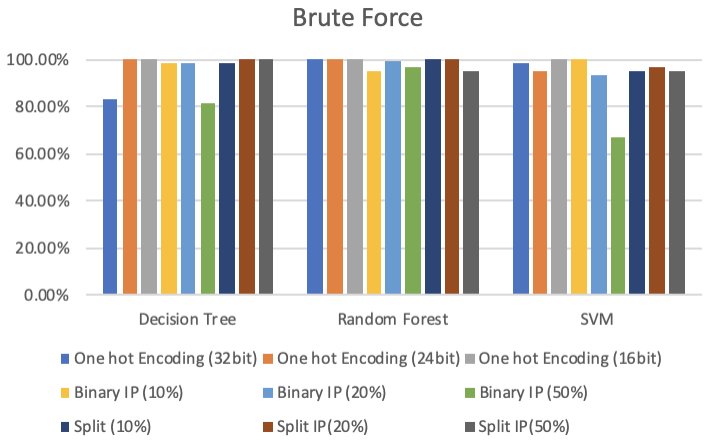

Figure 4.5. F-1 Scores for Brute Force

Figure 4.6 showed the F-1 scores for detecting the web attack. The F-1 scores of decision tree and random forest relatively higher than SVM. Some F-1 scores of SVM were even lower than 80 percent. Therefore, both decision tree and random forest could probably be good machine learning algorithms for detecting the web attack.

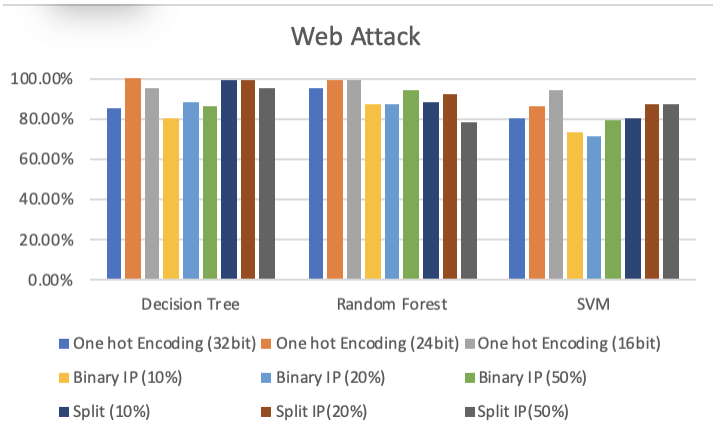

Figure 4.6. F-1 Scores for Web Attack

Figure 4.7 represented the F-1 scores for detecting the DoS attack. The F-1 scores of SVM were mostly very low. Based on the chart, the F-1 scores of random forest would relatively be higher than the other two. Therefore, random forest would possibly a good machine learning algorithm for detecting the DoS attack.

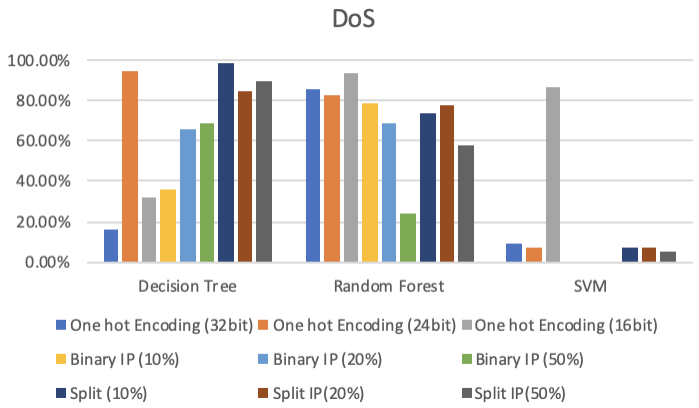

Figure 4.7. F-1 Scores for DoS Attack

Figure 4.8 represented the F-1 scores for detecting the infiltration attack. It was difficult to decide which algorithm was better for detecting infiltration attack because most of the F-1 scores were very low.

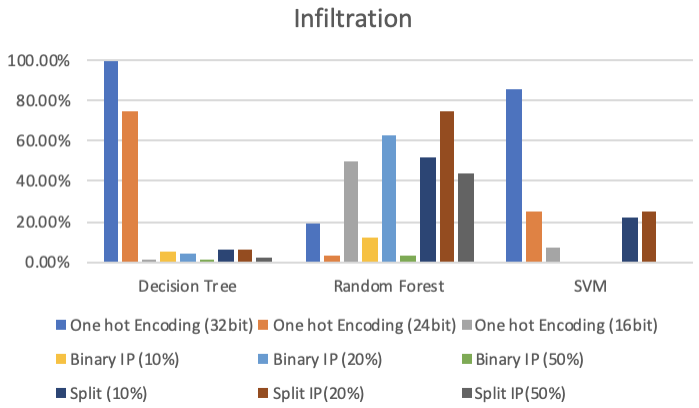

Figure 4.8. F-1 Scores for Infiltration Attack

Figure 4.9 represented the F-1 scores for detecting the botnet attack. It was also difficult to determine a better machine learning algorithm for detecting the botnet attack. The discrete of all F-1 scores for each machine learning was high.

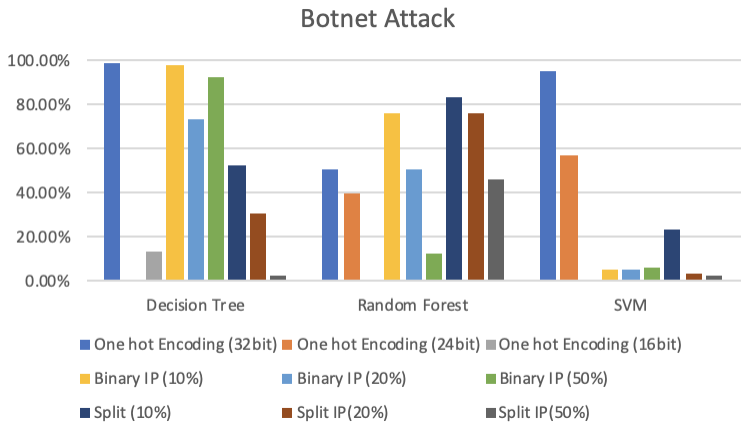

Figure 4.9. F-1 Scores for Botnet Attack

Figure 4.10 displayed the F-1 scores for detecting the DDoS attack. The F-1 scores of random forest and decision tree were mostly higher than 80 percent, and most F-1 scores of SVM were lower than 60 percent. As a result, decision tree and random forest would be better for detecting the DDoS attack.

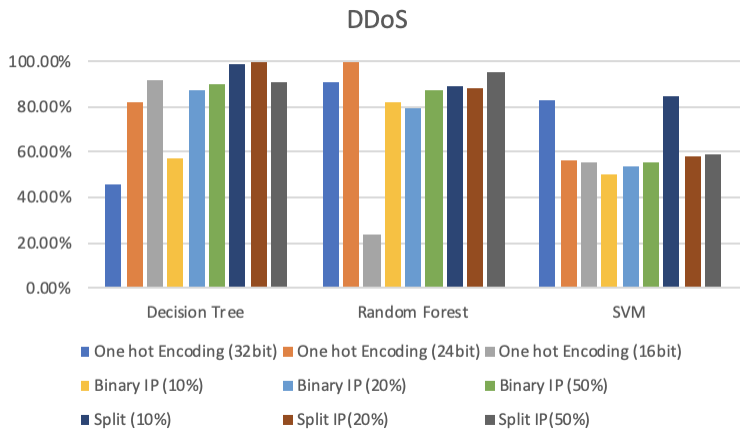

Figure 4.10. F-1 Scores for DDoS Attack

Figure 4.11 displayed the F-1 scores for detecting the PortScan attack. Most F-1 scores were higher than 90 percent and the lower F-1 score still existed for SVM. Thus, both decision tree and random forest could possibly be fit for detecting the PortScan attack.

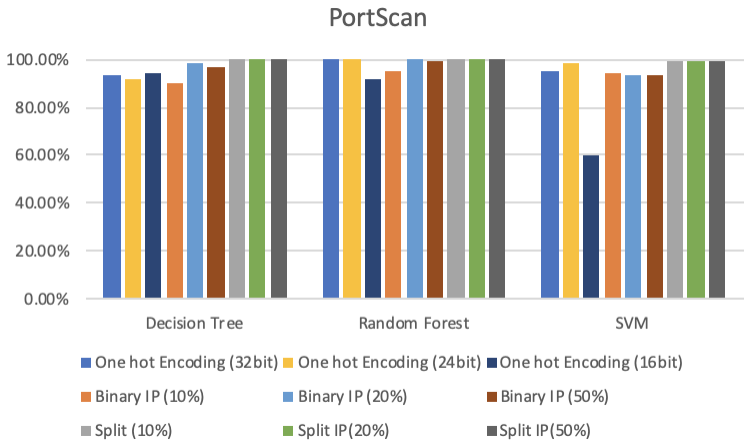

Figure 4.11. F-1 Scores for PortScan Attack

4.4 Summary

According to charts and tables from section 4.1 to 4.2, the method of split IPs gave the best results among decision tree, random forest, and SVM. As a result, it was determined that the method of splitting IP addresses to 4 numbers was the best way to encode IP addresses. In section 4.3, F-1 scores of each kind of attack by applying the machine learning algorithms were compared and the goal was to determine the relatively better machine learning algorithms for each kind of attack.

CHAPTER 5. SUMMARY, CONCLUSION, AND FUTURE WORK

5.1 Summary and conclusion

The goal of this thesis was to find the best method of encoding IP addresses for predicting network intrusion detection. Three methods of encoding IP addresses were applied to predict network attack traffic: converting IP addresses to binary numbers, splitting IP addresses into four numbers and one hot encoding. Moreover, different IP address sizes were considered and one hot encoding was applied to each size individually. All of the methods of encoding IP addresses were evaluated using random forest, decision tree, and SVM. The recall, precision, F-1, and accuracy scores were provided in the previous chapter, and the accuracy scores were compared.

In terms of processing speed, decision tree was the quickest and the duration was about five minutes at most. It took a little bit more time for random forest because multiple trees were created. The total time for processing by random forest was 20 minutes at most. SVM was the slowest, especially for one hot encoding and it took almost 48 hours to process the data.

As shown in chapter 4, one hot encoding had better results than binary IPs; however, the method of one hot encoding was not excellent. The results of split IP addresses were better than those of one hot encoding, and the accuracy scores of split IPs were mostly higher than those of one hot encoding. One hot encoding was supposed to be better for pre-processing data for machine learning; one possible reason why one hot encoding did not give the best result was the high dimensions of the data set. Since the amount of data was in the millions, the dimensions of the data set were extremely high after encoding and this affected the results of the predictions. According to the results of the present study, the split IP method was the best way to encode the IP addresses. To further prove this, the binary IP and split IP methods were compared in section 4.3, and the results show that splitting IP into four numbers was still the best method for encoding IP addresses for network intrusion detection.

5.2 Future Work

Due to the limitations of the system environment and existing problems in the methods, the approaches could be improved using the following recommendations:

- Deep learning could be used for predicting network intrusion detection.
- All data could be considered for one hot encoding if there was sufficient RAM.

REFERENCES

Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A., Kossaifi, J., ... Varoquaux, G. (2014). Machine learning for neuroimaging with scikit-learn. *Frontiers in Neuroinformatics*, *8*. doi: 10.3389/fninf.2014.00014

Assefi, M., Behravesh, E., Liu, G., & Tafti, A. P. (2017, Dec). Big data machine learning using apache spark mllib. In *2017 ieee international conference on big data (big data)* (p. 3492-3498). doi: 10.1109/BigData.2017.8258338

Belayneh, A. M., Adamowski, J., Khalil, B., & Quilty, J. (2015). Coupling machine learning methods with wavelet transforms and the bootstrap and boosting ensemble approaches for drought prediction. *Atmospheric Research*, *172–173*, 37–47. doi: 10.1016/j.atmosres.2015.12.017

Deng, S., Luo, J., Liu, Y., Wang, X., & Yang, J. (2014). Ensemble learning model for p2p traffic identification. In *2014 11th international conference on fuzzy systems and knowledge discovery (fskd)* (pp. 436–440). IEEE.

Dietterich, T. (2000). Ensemble methods in machine learning. *Multiple Classifier Systems*, *1857*, 1–15.

Garca, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers Security*, *45*, 100123. doi: 10.1016/j.cose.2014.05.011

Lei, S., Xinming, M., Lei, X., & Xiaohong, H. (2010). Financial data mining based on support vector machines and ensemble learning. *2010 International Conference on Intelligent Computation Technology and Automation*. doi: 10.1109/icicta.2010.787

Li, K., & Hao, L. (2009). Naive bayes ensemble learning based on oracle selection. *2009 Chinese Control and Decision Conference*. doi: 10.1109/ccdc.2009.5194867

Liu, N., & Wang, H. (2010, Aug). Ensemble based extreme learning machine. *IEEE Signal Processing Letters*, *17*(8), 754-757. doi: 10.1109/LSP.2010.2053356

Livadas, C., Walsh, R., Lapsley, D., & Strayer, W. T. (2008). Usilng machine learning technliques to identify botnet traffic. In *Proceedings. 2006 31st ieee conference on local computer networks(lcn)* (p. 967-974). Retrieved from `doi.ieeecomputersociety.org/10.1109/LCN.2006.322210` doi: 10.1109/LCN.2006.322210

Miller, S., & Busby-Earle, C. (2016). The role of machine learning in botnet detection. In *2016 11th international conference for internet technology and secured transactions (icitst)* (pp. 359–364). Infonomics Society.

Mitchell, H. B., & Schaefer, P. A. (2001). A soft k-nearest neighbor voting scheme. *International Journal of Intelligent Systems*, *16*(4), 459-468. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/int.1018 doi: 10.1002/int.1018

Newman, C. B. D., & Merz, C. (1998). *UCI repository of machine learning databases.* Retrieved from http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html

Oyama, S., & Ishida, T. (2003). Applying association rules to information navigation. *Systems and Computers in Japan*, *34*(4), 12-20. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/scj.10120 doi: 10.1002/scj.10120

Ross, M., Graves, C. A., Campbell, J. W., & Kim, J. H. (2013). Using support vector machines to classify student attentiveness for the development of personalized learning systems. In *2013 12th international conference on machine learning and applications* (Vol. 1, p. 325-328). doi: 10.1109/ICMLA.2013.66

Ryu, S., & Yang, B. (2018). A comparative study of machine learning algorithms and their ensembles for botnet detection. *Journal of Computer and Communications*, *06*(05), 119129. doi: 10.4236/jcc.2018.65010

Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th international conference on information systems security and privacy - volume 1: Icissp,* (p. 108-116). SciTePress. doi: 10.5220/0006639801080116

Stevanovic, M., & Pedersen, J. M. (2014). An efficient flow-based botnet detection using supervised machine learning. In *2014 international conference on computing, networking and communications (icnc)* (p. 797-801). doi: 10.1109/ICCNC.2014.6785439

Tan, Y., & Zhang, G.-J. (2005, Aug). The application of machine learning algorithm in underwriting process. In *2005 international conference on machine learning and cybernetics* (Vol. 6, p. 3523-3527). doi: 10.1109/ICMLC.2005.1527552

Ting, K. M., & Zheng, Z. (2003). A study of adaboost with naive bayesian classifiers: Weakness and improvement. *Computational Intelligence*, *19*(2), 186–200.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Sigir.*

Zhan, Y., Chen, H., Zhang, S.-F., & Zheng, M. (2009). Chinese text categorization study based on feature weight learning. In *2009 international conference on machine learning and cybernetics* (Vol. 3, p. 1723-1726). doi: 10.1109/ICMLC.2009.5212257

Zhang, P., Su, Y., & Wang, C. (2007, Aug). Statistical machine learning used in integrated anti-spam system. In *2007 international conference on machine learning and cybernetics* (Vol. 7, p. 4055-4058). doi: 10.1109/ICMLC.2007.4370855