Intrusion Detection Using the WEKA Machine Learning Tool

by

Sadiq Ali
B.E., Dawood University of Engineering and Technology, Pakistan, 2013

A Report Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

Intrusion Detection Using the WEKA Machine Learning Tool

by

Sadiq Ali
B.E., Dawood University of Engineering and Technology, Pakistan, 2013

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Mihai Sima, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

The internet has become essential for data sharing, so it must be secure. Data encryption and authentication are not sufficient for internet security, and firewalls are unable to detect fragmented malicious packets. Further, attackers are constantly modifying their techniques and tools to produce devastating consequences such as productivity loss, financial loss and brand damage. Thus, it has become crucial to implement an efficient intrusion detection system, which is a very difficult challenge. The CSE-CIC-IDS2018 dataset consists of 14 network attacks and benign traffic with 80 features. In this report, data for seven network attacks and benign data are used. Principal Components Analysis (PCA) is employed to extract the ten most relevant features. The Machine Learning (ML) algorithms studied are Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), *K*-Nearest Neighbor (KNN), Random Forest (RF) and Bayesian Network (BayesNet). The experiments are performed using the Waikato Environment for Knowledge Analysis (WEKA) tool with five-fold and ten-fold cross-validation. The performance measures employed are accuracy, precision, recall, F-measure, and execution time. The results obtained show that RF outperforms the other algorithms in accuracy, precision, recall, and F-measure.

Contents

List of Tables

List of Figures

Glossary

AI ..... Artificial Intelligence

API ..... Application Programming Interface

AIDS ..... Anomaly-based Intrusion Detection System

CIC ..... Canadian Institute for Cybersecurity

CPU ..... Central Processing Unit

CSE ..... Communications Security Establishment

DDoS ..... Distributed Denial-of-Service

DoS ..... Denial of Service

GNL ..... General Public License

GUI ..... Graphical User Interface

HTTP ..... Hypertext Transfer Protocol

HIDS ..... Host-based Intrusion Detection System

IDS ..... Intrusion Detection System

IP ..... Internet Protocol

IT ..... Information Technology

KNN ..... K-Nearest Neighbor

ML ..... Machine Learning

NIDS ..... Network Intrusion Detection System

RAM ..... Random Access Memory

SIDS ..... Signature-based Intrusion Detection System

SIEM ..... Security Information and Event Management

SMO ..... Sequential Minimal Optimization

SMOTE ..... Synthetic Minority Oversampling Technique

SVM ..... Support Vector Machine

UDP ..... User Datagram Protocol

WEKA ..... Waikato Environment for Knowledge Analysis

ACKNOWLEDGEMENTS

I would like to thank:

My Parents, for supporting me for my educational and professional career.

Dr. T. Aaron Gulliver, for mentoring, support, encouragement, and patience.

My Siblings, for their love and motivation.

Salahuddin Jokhio, Assad Raza, and Yasir Hussain, for their encouragement and support.

"Remember to celebrate milestones as you prepare for the road ahead.
Nelson Mandela

DEDICATION

I dedicate this work to my parents for their love, support, and sacrifices. Thank you for believing in me.

Chapter 1

Introduction

The information era has had a significant influence and considerable impact on most industries. As a result, the Information Technology (IT) market has developed into a vast multinational sector. The world now regards data as a more important resource than oil [1]. Analysis of data such as voting patterns and consumer purchases is now profitable. Thus, governments and private enterprises have made significant investments in IT infrastructure and facilities, raising concerns about protection and privacy. Moreover, the use of online data repositories for businesses has led to many cybercriminal activities. People involved in stealing confidential and personal information of a company or person are known as attackers or cybercriminals. Cyberattacks are malicious acts that differ in severity depending on the purpose and data involved.

Computer networks, facilities, and devices have faced an increasing number of threats in recent years. Thus, cybersecurity has become a major issue. Identifying malicious activities, especially those that have not yet occurred, is a critical problem that must be addressed urgently. One way to handle cyberattacks is to employ an Intrusion Detection System (IDS). These systems detect abnormal behavior or intrusions in a network through continuous monitoring. However, a conventional IDS cannot process large volumes of data and may not respond accurately to new threats. Moreover, traditional intrusion detection techniques that rely on identifying and matching port numbers and IP addresses are becoming less effective as new cyberattacks emerge and communication protocols expand. As a result, Machine Learning (ML) techniques for intrusion detection are being developed. ML is a branch of Artificial Intelligence (AI) that can recognize hidden trends or patterns in data and make accurate predictions.

In this project, Principal Component Analysis (PCA) is applied to the CSE-CIC-IDS2018 dataset to identify the ten most important features. To identify the most effective ML algorithms, seven classifiers are evaluated to predict seven attacks and legitimate (benign)

traffic.

1.1 Motivation

Cybercriminals have developed ways to identify system vulnerabilities and exploit network weaknesses [2]. According to the IDC Data Services for Hybrid Cloud Survey, almost 40% of IT security specialists, data management experts and business owners cite the increasing number of cyberattacks and complexity in supporting and managing security products as significant challenges [2]. In March 2020, an attacker hacked Sina Weibo (a Chinese social media platform), and obtained part of its database containing the personal information of 538 million users including email addresses, phone numbers, gender and other social media records. This kind of incident allows attackers to utilize the data for social engineering attacks. In another incident, a malicious third-party accessed account information including names, email addresses, encrypted passwords, and user data linked to Quora. The outcomes of these attacks can be financial loss, brand damage, productivity loss and strained relationships with the customers. This project aims to evaluate the performance and efficiency of ML algorithms in terms of accuracy, precision, recall, F-measure, and execution time to determine their suitability in IDSs.

1.2 Related Work

In this section, previous research related to IDSs is presented. In [3], the CICIDS2017 IDS dataset was introduced. This dataset consists of seven real-time attacks with 80 network traffic features. The Random Forest Regressor feature reduction algorithm was used to determine the best feature set for detecting each attack. Then, seven ML classifiers were employed to evaluate the performance with these features. The results obtained showed that Random Forest has the highest accuracy and shortest execution time. Moreover, a comparison was made between the CICIDS2017 dataset and 11 publicly available datasets based on criteria for an ideal dataset in [4]. It was concluded that none of the dataset exceeded the criteria except CICIDS2017.

In [5], the Random Tree, J48, Naïve Bayes, RF, Decision Table, Bayes Network, and MLP algorithms were evaluated using the KDD IDS dataset. This dataset consists of 19% benign instances, 79% DoS attacks and 2% other attacks. SQL server 2008 was used to extract 148753 instances to train the model and 60000 instances for testing. The results obtained show that the Random Forest classifier achieved the highest accuracy

and the smallest Root Mean Square Error (RMSE) and False Positive Rate (FPR). RMSE is a measure of the difference between the predicted and actual outcomes, while FPR is the ratio of benign instances incorrectly classified as attacks to the total number of benign instances.

In [6], fifteen ML classification algorithms, namely BayesNet, Naïve Bayes, Naïve Bayes Updateable, Naïve Bayes Multinominal Text, Logistic, Simple Logistic, Voted Perceptron, Decision Table, JRip, OneR, PART, ZeroR, J48, Random Tree, and Random Forest were implemented. The NSL-KDD dataset was considered using the data processing tool WEKA with 10-fold cross-validation. The results obtained show that Random Tree has the lowest execution time and the highest accuracy.

1.3 Report Organization

The structure of the report is as follows.

Chapter 1 introduced the problem and provided an overview of the project. The related work and motivation for this project were discussed and the structure of the report was given.

Chapter 2 provides details on the IDS types, the proposed framework, and the multi-label classification techniques used in this project.

Chapter 3 gives an overview of ML tasks, data splitting techniques, the WEKA tool, and the ML classifiers to predict benign and attack classes.

Chapter 4 provides information on the software, hardware and test environment configurations. The performance evaluation of the MLP, BayesNet, KNN, SVM, RF, SMO, and Simple Logistic algorithms is also given.

Chapter 5 gives the conclusions and directions for future work related to IDSs.

Chapter 2

Intrusion Detection Systems

An Intrusion Detection System (IDS) is software or a device that monitors network traffic to detect malicious activity. This activity is collected by a Security Information and Event Management (SIEM) system and reported to an administrator or security expert. This is a software system that provides real time analysis of security alerts generated by applications and network hardware such as firewalls, servers, switches, routers, and workstations. Network security specialists analyze these alerts. The primary purpose of an IDS is to detect malicious traffic in the early stages, which is impossible with a conventional firewall. Malicious software, known as malware, is evolving rapidly and poses significant challenges to IDS systems. Malware is software designed to exploit flaws in computer systems and networks. The most difficult challenge is the identification of unknown or obscured malware. Malware developers employ various tactics to conceal information and bypass detection by an IDS. Security threats like zero-day attacks have recently increased. As a result, computer security systems have become essential. A significant number of cybercriminals are driven to steal information, earn illicit profits, and find new targets. Therefore, a reliable IDS is required to detect sophisticated and unknown malware. Figure 2.1 presents the operation of an IDS system.

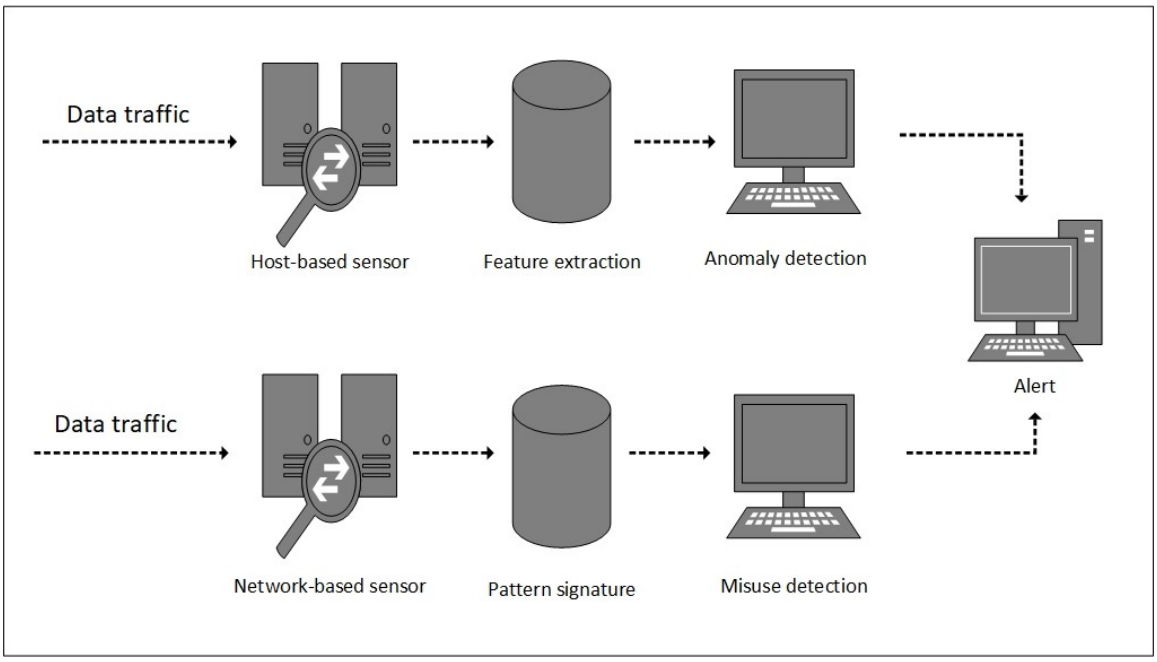

Figure 2.1: Signature and anomaly-based IDS systems.

There are two main IDS types:

- Network IDS
- Host IDS

2.1 Network IDS

A Network Intrusion Detection System (NIDS) is typically placed or installed at strategic points in a network to protect traffic from attack. It compares the traffic observed with a database of known attacks. A NIDS is relatively simple to install and is difficult for intruders to detect. As a result, an attacker may be unaware that a NIDS is installed. A NIDS normally analyzes a huge amount of network traffic and may miss encrypted traffic because it is unable to scan the contents of the packets. Thus, constant monitoring by an administrator is needed.

2.2 Host IDS

A Host Intrusion Detection System (HIDS) is an IDS that is deployed on computers and network devices connected to the internet. An HIDS can analyze internal traffic more

critically than a NIDS and works as a second layer of defense against network attacks.

An HIDS examines system processes, memory regions and keeps track of attributes such as authorizations, modification dates, and hashed contents to recognize changes. It also scans the system registry, failed login attempts, and checks for the installation of malicious applications.

There are two approaches used to detect intrusions:

- Signature-based Intrusion Detection System (SIDS)
- Anomaly-based Intrusion Detection System (AIDS)

2.2.1 Signature-based IDS

A Signature-based Intrusion Detection System (SIDS) uses pattern-matching techniques to detect specific attacks and is often called misuse or knowledge-based detection. In these systems, an alarm is triggered when an intrusion matches a prior intrusion signature in the database [7].

For previously identified intrusions, SIDS typically provides excellent detection accuracy [8]. On the other hand, SIDS cannot identify zero-day attacks because no matching signature exists in the database before a new attack [11]. SIDS is used in various tools such as NetSTAT [9] and Snort [10].

SIDS is unable to detect fragmented attack packets. However, modern malware has become advanced so signature information from multiple packets may be required. Polymorphic malware (a type of malware that continuously changes its features to avoid detection), and the growing number of targeted attacks could further reduce the effectiveness of conventional SIDS.

2.2.2 Anomaly-based IDS

An Anomaly-based Intrusion Detection System (AIDS) can overcome the limitations of SIDS. A standard model of computer system behaviour is generated in an AIDS using machine learning, knowledge-based, or statistical-based approaches. An anomaly is defined as a substantial difference between model and observed behaviour which can be perceived as an intrusion. The assumption is that malicious behaviour varies from normal user behaviour. Training and testing are the two AIDS phases. During training, the model is trained on extracted network traffic data and later tested against unseen data to determine the accuracy. The key benefit of AIDS is detecting zero-day attacks because

it does not rely on signatures or patterns to detect threats [12]. When the observed behaviour varies from normal activity, AIDS triggers an alarm.

2.3 Proposed Framework

The proposed framework is shown in Figure 2.2 and outlines the steps used to train and test the ML model.

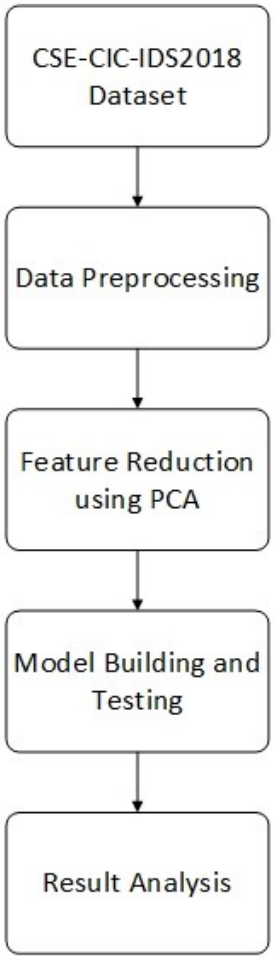

Figure 2.2: The proposed framework.

2.3.1 CSE-CIC-IDS2018 Dataset

The CSE-CIC-IDS2018 dataset was developed as a joint project between the Canadian Institute for Cybersecurity (CIC) and the Communications Security Establishment (CSE).

This dataset includes 14 different attacks, namely Bot, Infiltration, SSH-BruteForce, FTP-BruteForce, Brute Force-Web, Brute Force-XSS, SQL-Injection, DoS attacks-Hulk, DoS attacks-GoldenEye, DoS attacks-Slowloris, DoS attacks-SlowHTTPTest, DDoS attack-LOIC-UDP, DDoS attack-HOIC, DDoS attacks-LOIC-HTTP and Benign. The first seven attacks and benign are considered in this report. The details of these attacks are given below.

Brute Force Attack is a method of gaining unauthorized access to a system or login webpage by cracking a username and password combination. In some cases, attackers use scripts and applications as brute force tools. Attackers can obtain a list of commonly used or real credentials via a security breach or the dark web. Most brute force attacks are now performed by bots. The bots target websites using credentials (usernames and passwords), and notify hackers when they gain access. Burpsuite, Patator, THC Hydra, Medusa and Metasploit are tools available in Kali Linux which can be used to initiate a brute force attack against a system.

Botnet is a network of computers infected by malware which is linked to a malicious host computer known as a bot. This bot is controlled by an attacker who can send commands to the computer to carry out malicious activities. Zeus and Ares are two well-known bots which were used in the dataset creation. Zeus is Trojan horse malware that spreads through downloading a file from an infected drive or by phishing schemes whereas Ares is malware used to launch DDoS attacks.

Denial of Service (DoS) is a type of cyberattack in which an attacker floods a server with traffic, making resources unavailable to legitimate users [13]. Slowloris, GoldenEye, and Hulk are used to launch DoS attacks. Slowloris aims to bring down the server by sending partial Hypertext Transfer Protocol (HTTP) requests to initiate connections between the attacking machine and targeted system without affecting other ports and services. It tries to stay connected for as long as possible to overwhelm and slow the target system. Slowloris requires minimal bandwidth to launch a DoS attack. GoldenEye is a popular DoS attack tool. It uses KeepAlive (and Connection:keep-alive), paired with cache-control options to open many socket connections and stay connected until it consumes all available sockets on the server [14]. Hulk is a variant of GoldenEye used to bypass caching and hit the server resource pool with a high volume of unique and obscure traffic [15].

Distributed Denial of Service (DDoS) works on the same principle as a DoS attack but is launched from a network of connected devices to consume the resources of a target machine. Different tools are available for DDoS attacks such as High Orbit

Ion Cannon (HOIC) and Low Orbit Ion Cannon (LOIC). HOIC and LOIC are open-source flooding tools developed for network stress testing [16]. These tools are able to generate a substantial amount of network traffic in order to consume network resources. A DDoS attack can be initiated by flooding HTTP and User Datagram Protocol (UDP) packets against multiple URLs simultaneously.

Web Application Attack The Damn Vulnerable Web App (DVWA) is insecure PHP/MySQL free web application software. In the development of the dataset, DVWA was used as a victim exploited by an SQL injection attack. SQL injection is a web attack in which an attacker attempts to manipulate the data on the server using SQL commands and queries on a vulnerable website. A web application scanner is used to scan for vulnerabilities and launch an attack if any exist. The main objective of this attack is to steal and destroy sensitive information.

Infiltration occurs when a malicious program or a suspicious document is downloaded by a user. The Metasploit tool is used on the victim computer to create a backdoor. This allows remote access to the machine without permission or knowledge for surveillance, data theft, and server hijacking.

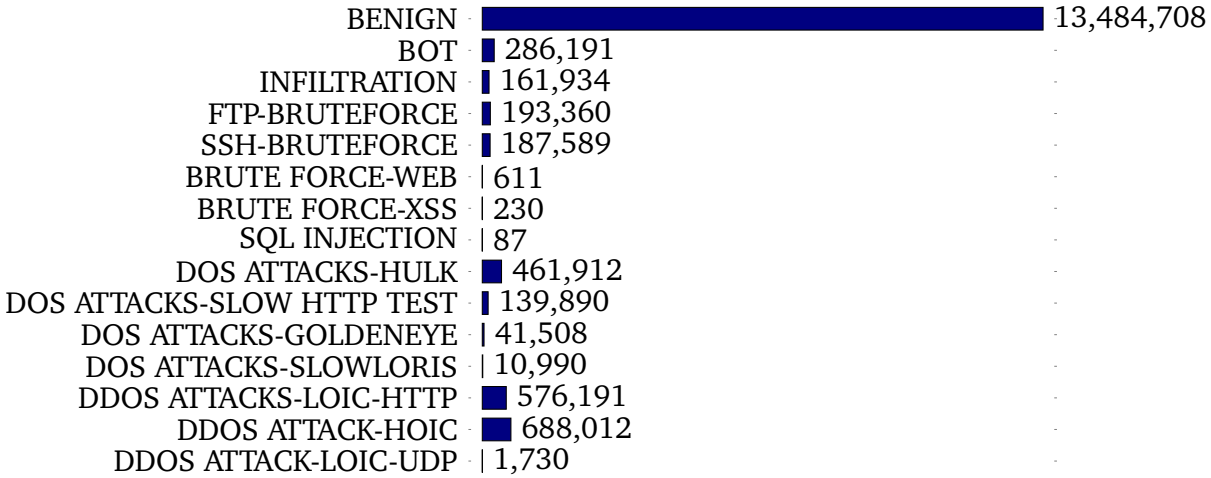

Table 2.1: The numbers of CSE-CIC-IDS2018 dataset instances [17].

2.3.2 Data Preprocessing

Data preparation is an essential step in building an ML model. This phase consists of data cleaning, standardization, and feature reduction.

Data Cleaning

Data cleaning involves identifying irrelevant, inaccurate, and incomplete data and refining, replacing or deleting it. This is achieved as follows.

1. Missing values are replaced with mean values.
2. The two columns Flow Bytes and Flow Pkts/s contain some infinity values and these are replaced with the mean values.
3. Remove the Time Stamp columns which are not useful features.
4. Remove the BWD URG, BWD PSH, Fwd Pkts/b Avg, Fwd Byts/b Avg, Fwd Blk Rate Avg, Bwd Pkts/b Avg, Bwd Byts/b Avg, and Bwd Blk Rate Avg columns which have zero values.

   After removing the unnecessary columns, the dataset is reduced to 70 features.

Data Standardization

Data standardization is used to ensure the feature value ranges are uniform by changing the mean to 0 and the standard deviation to 1. Thus, values greater than the mean become positive values, and the others become negative.

Dimensionality Reduction using Principal Component Analysis

Principal Component Analysis (PCA) is used to reduce the dimensionality of datasets by transforming a large set of correlated features into a smaller set of less correlated features called principal components which retain the majority of the information contained in the original dataset [18]. The accuracy of a classification model may decrease with a reduced number of features, so dimensionality reduction is a tradeoff between accuracy and simplicity because smaller datasets are easier to process. PCA has the following steps. The first is to normalize the values of the dataset features which is done using the default standardization in the WEKA platform. In the next step, the correlation matrix is calculated to identify the relationships between features. Then, the eigenvalues and eigenvectors are obtained using eigendecomposition on this matrix. Eigenvectors

are principal components, whereas the eigenvalues are the variances of the components. Then the eigenvalues are sorted in descending order, so the eigenvector with the highest eigenvalue is the first principal component of the dataset. This ranks the principal components. The components with small eigenvalues are discarded because they are less important. PCA is available in WEKA in the preprocess panel under the unsupervised folder. A rank filter is used with PCA which ranks the features from highest to lowest. The 10 highest ranked features extracted are shown in the output space in Figure 2.3.

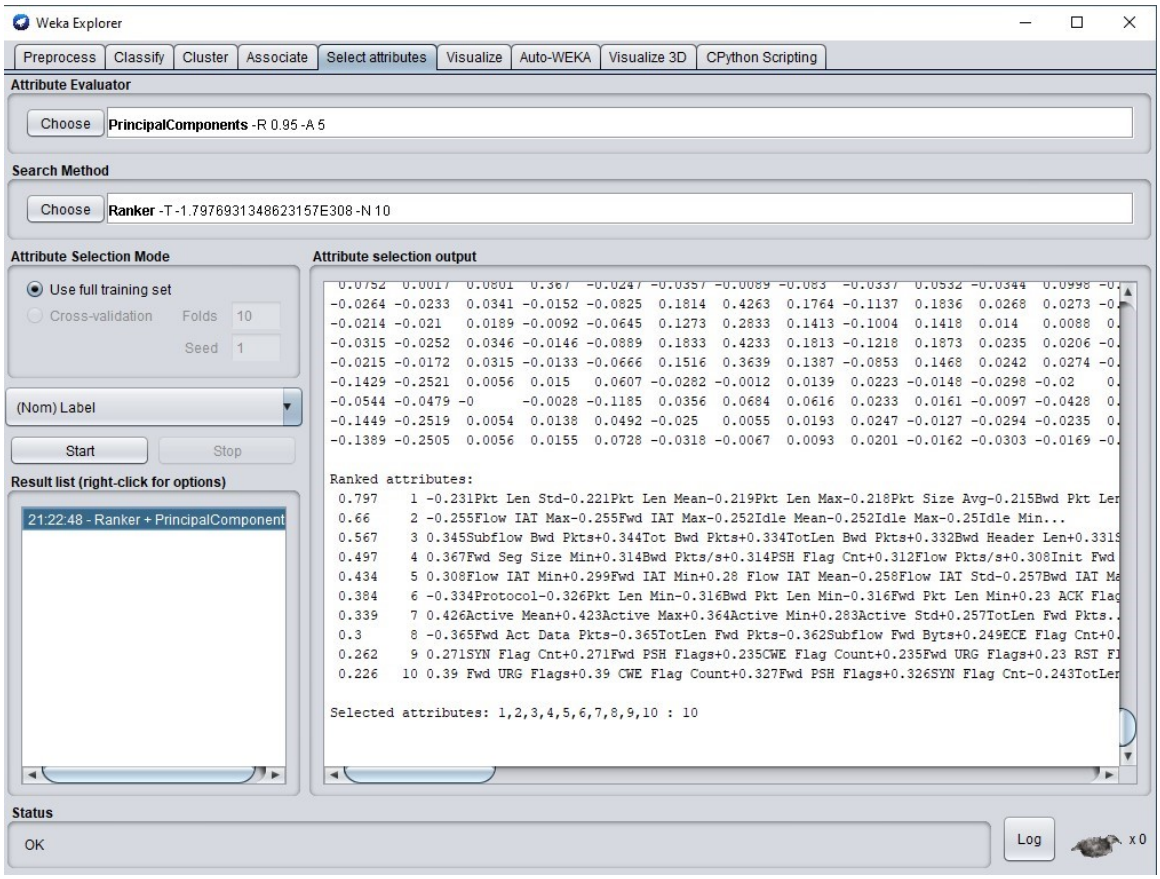

Figure 2.3: PCA feature reduction in WEKA.

2.3.3 Model Building and Testing

In this phase, the models are trained and tested using the CSE-CIC-IDS2018 dataset with the features selected in the dimensionality reduction stage. Five-fold and ten-fold cross-validation is used for training and testing to check the performance of the classifiers.

Chapter 3

Machine Learning

Machine learning (ML) is a branch of Artificial Intelligence (AI) that provides the ability to learn without being explicitly programmed. It has roots in statistics, data mining, and computational mathematics. ML algorithms are trained on a dataset to recognize patterns in the data to make future predictions. ML is widely used in IDSs. The types of ML algorithms are described below.

3.1 Types of Machine Learning Algorithms

ML algorithms can be divided into three categories.

- Supervised learning
- Unsupervised learning
- Reinforcement learning

3.1.1 Supervised Learning

The most widely used ML technique is supervised learning. It is the simplest method to employ. A supervised ML algorithm is trained on labelled data to yield the desired output [19] via feedback on whether the predicted output is right or wrong. When the algorithm is fully trained, it observes new instances and makes predictions of labels. Supervised learning is also known as task-oriented learning. This type of learning has many applications such as spam classification, face recognition, and advertisement popularity. Supervised learning can be divided into two categories.

- Regression: algorithms used to predict continuous values, such as salary, age, and price.
- Classification: algorithms used to predict discrete values such as spam or not spam, and true or false. Classification can be either binary or multi-class.

Some of the supervised learning algorithms are Naïve Bayes, Decision Trees, KNN, Neural Networks, SVM, and Linear Regression.

3.1.2 Unsupervised Learning

Unsupervised learning is a ML technique used to identify hidden patterns in a dataset containing data that has not been labelled [20]. Thus, it does not need labels to train the model, but the results are less accurate compared to supervised learning. Unsupervised learning is suitable when data is not available for desired outcomes, such as when a company wants to determine a target market for a newly launched product [20]. Unsupervised learning is used for clustering and dimensionality reduction problems. Some unsupervised learning algorithms are Hidden Markov model, K-means clustering and Gaussian mixture models.

3.1.3 Reinforcement Learning

Reinforcement learning is a ML technique in which a ML model learns from the feedback it receives from its actions and experiences. It is similar to how humans think and learn from their environment. Reinforcement learning is commonly used in video games, industrial simulation and resource management. Some of the popular reinforcement learning algorithms are Temporal difference, Q-learning and Deep adversarial networks.

3.2 Data Splitting

A important decision is how to use the available data. The most popular techniques for splitting the data for training and testing are k-fold cross-validation and percentage split. Percentage split is a fast and simple method in which the data is split into training and test sets. The most common split ratios used are 80:20 and 70:30. The training set is used to develop of model and the test set is used to evaluate the model. It is important that the test set not be used before testing stage as this can bias the results and affect model evaluation.

In this project, k-fold cross-validation is applied to the data with $k = 5$, where k is the number of random and equal size data partitions. It is a data resampling procedure which is widely used in ML. It produces less biased outputs as compared to percentage split [21]. The model is trained using $k-1$ partitions while the other is used as the test set [22] as shown in Figure 3.1. This process is repeated k times and the cross-validation accuracy is the mean of the k results

$$acc_{cv} = \sum_{i=1}^{k} \frac{acc_i}{k}$$

where acc_i is the accuracy for fold i.

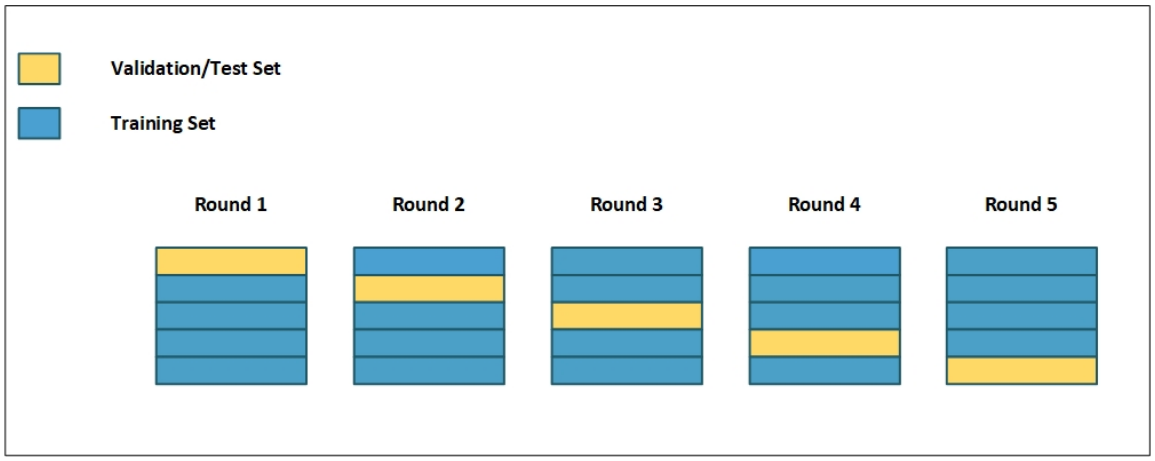

Figure 3.1: The five-fold cross-validation process.

3.3 The WEKA Workbench for Machine Learning

WEKA is an open-source ML tool that allow users to perform data processing tasks. WEKA was developed at the University of Waikato, New Zealand, under the General Public License (GNL) [23]. It was written using the Java programming language and can be accessed through the Graphical User Interface (GUI) or Java Application Programming Interface (API). A variety of functions are provided for data preprocessing, visualization, classification, regression, clustering, feature selection, and reduction. After startup, the WEKA GUI chooser allows the user to choose from the following five types of applications as shown in Figure 3.2

- Explorer

- Experimenter
- KnowledgeFlow
- Workbench
- Simple CLI

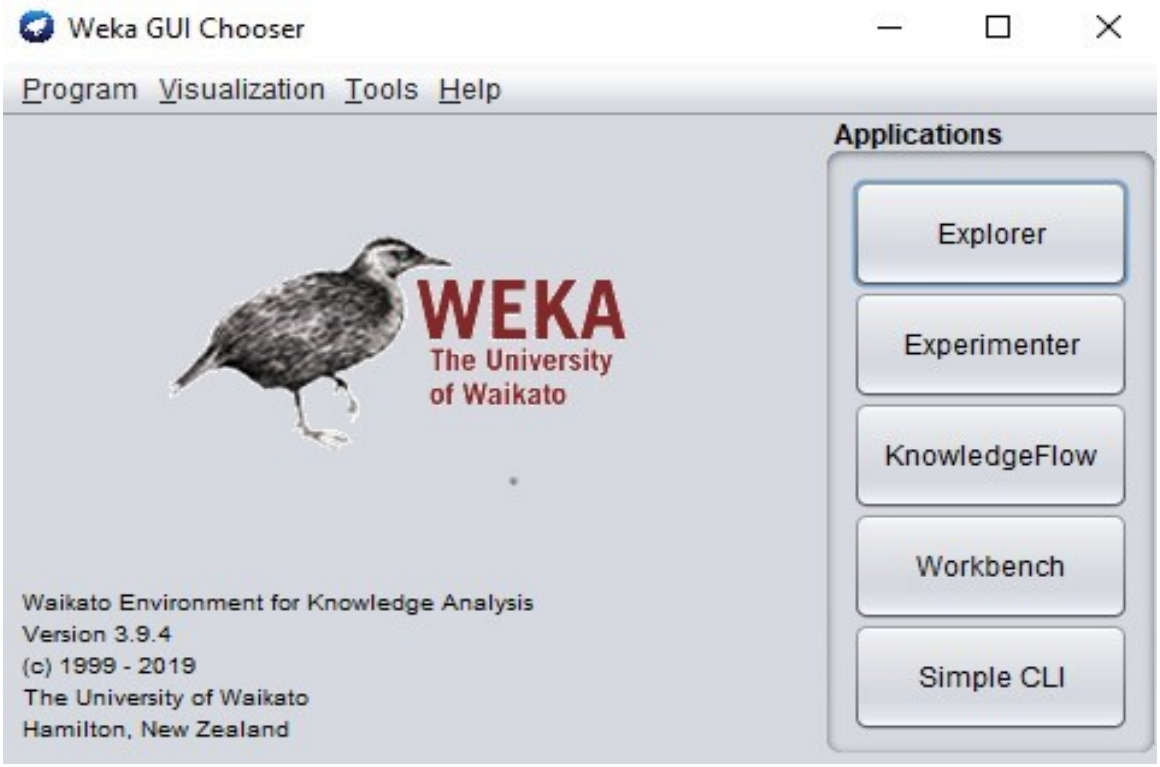

Figure 3.2: The WEKA GUI chooser.

The experiments were performed using the explorer application. Initially, only the preprocess tab is enabled which allows users to upload and preprocess the data. The data can be loaded from a local file directory, the web, or a database. The data is loaded from the Open file tab. WEKA supports a number of data file formats as shown in Figure 3.3.

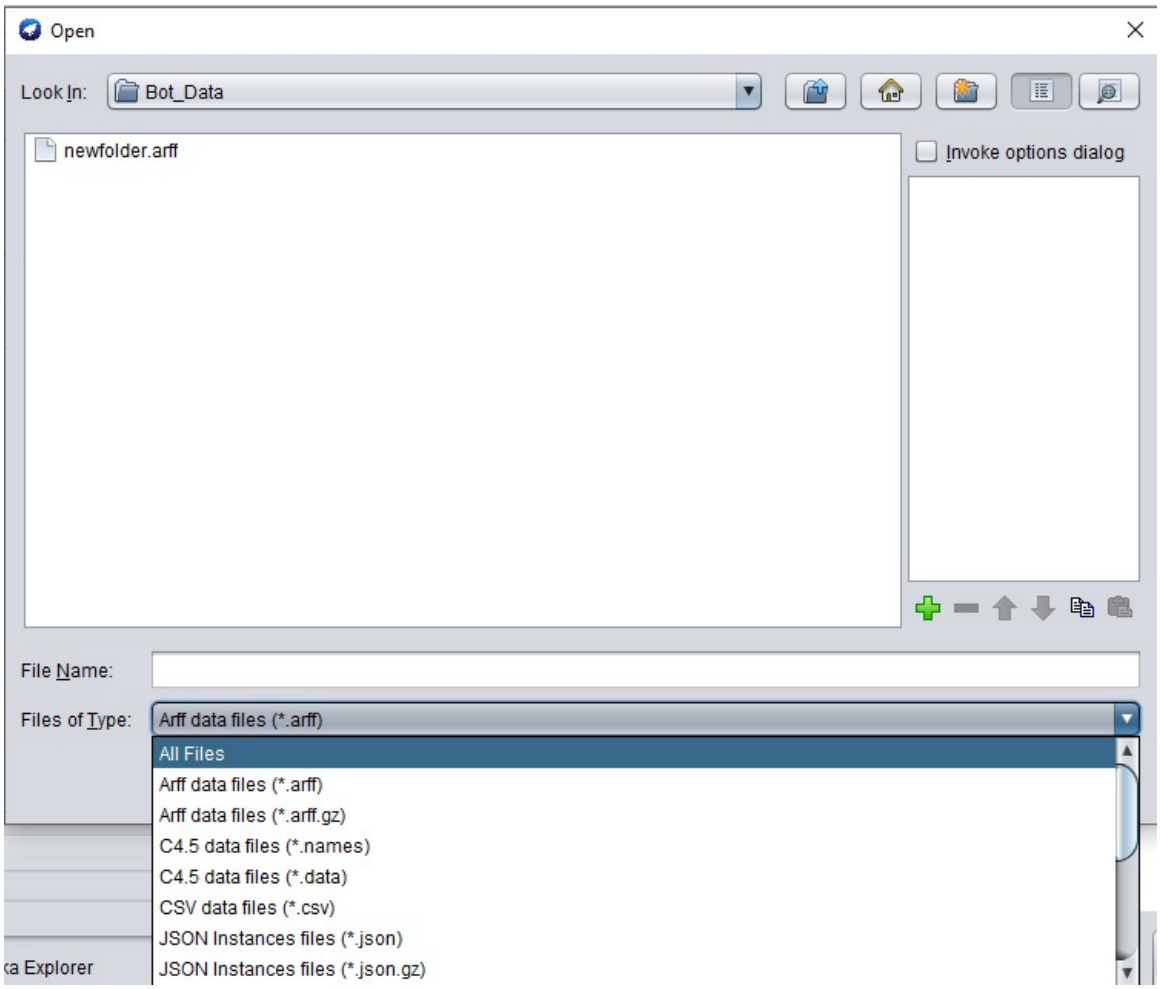

Figure 3.3: The file formats supported in WEKA.

The preprocess tab presents an overview of the dataset, such as the number of instances, attributes, missing values, and labels. A filter option supports ML tasks such as replacing missing values with mean values and several undersampling and oversampling techniques [25]. The details of the dataset in the WEKA preprocess panel are shown in Figure 3.4.

After data preprocessing, ML algorithms can be chosen from the classify panel. WEKA supports classifiers for categorical and numeric predictions such as KNN, C4.5 (decision trees), MLP, RF, SVM, and Simple Logistic.

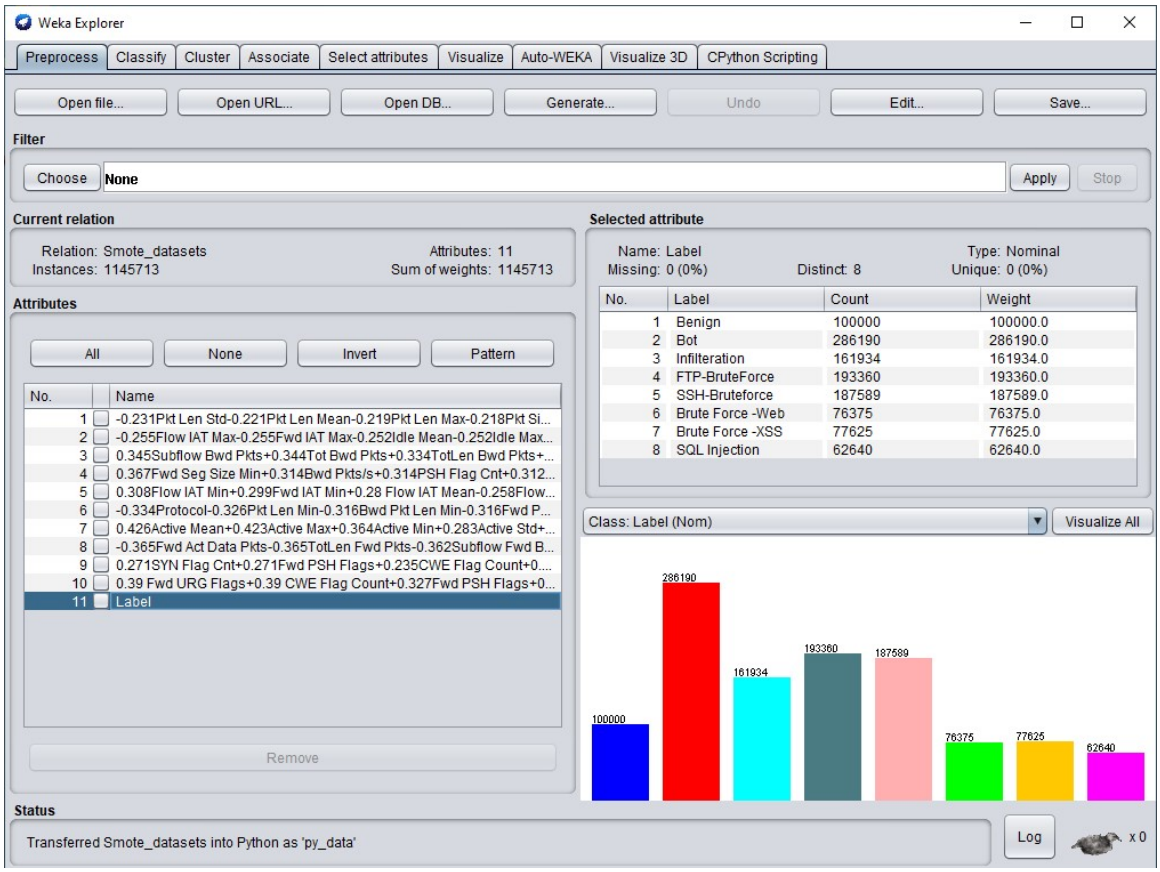

Figure 3.4: The dataset overview in WEKA.

WEKA offers two options to train and test a model, cross-validation and percentage split. Cross-validation divides the dataset using a number of folds whereas percentage split divides the dataset into explicit training and testing sets. The class labels must be specified prior to building and evaluating the model. In this project, eight labels are used to identify malicious and benign traffic. The model results are given in the classifier output space as shown in Figure 3.5. The output results show the performance and confusion matrix of each class. The output can be saved in .txt, .csv or .xml file formats.

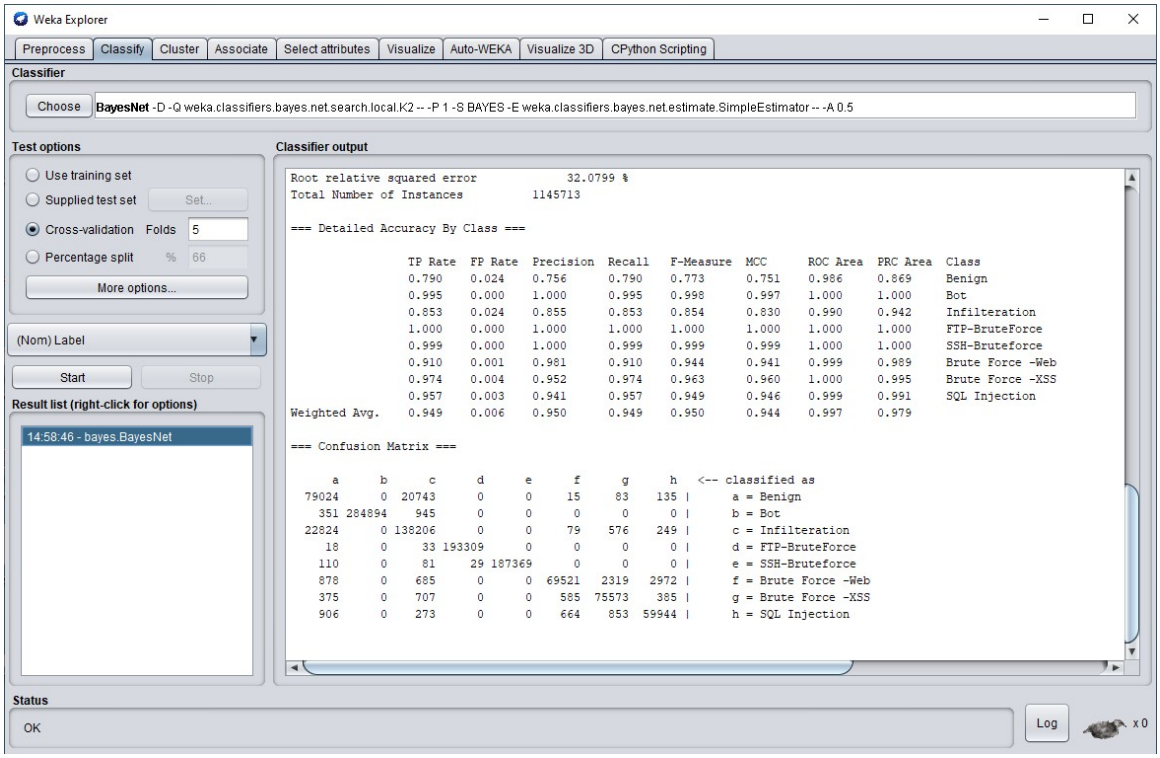

Figure 3.5: Classify tab with confusion matrix and classifier output.

3.4 Machine Learning Classifiers

The ML classifiers used in this project are described below.

3.4.1 Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is a form of Artificial Neural Networks (ANN) that consists of three types of layers, an input layer that receives input data, an output layer to make predictions, and one or more hidden layers in between input and output layers [26] as shown in Figure 3.6. The hidden layers are the computational engine. MLP is a well-known algorithm in which data flows in a forward direction similar to feedforward neural networks. MLP neurons are trained with a backpropagation algorithm. Pattern classification, recognition, prediction, and approximation are four MLP applications.

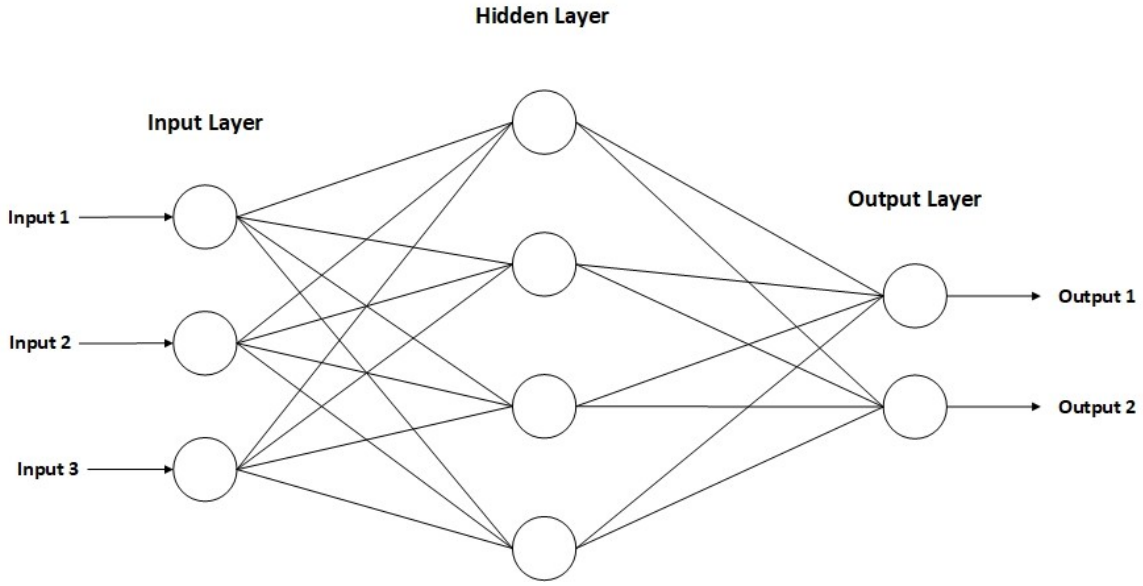

Figure 3.6: The MLP architecture.

3.4.2 Bayesian Network (BayesNet)

A Bayesian Network (BayesNet) is a probabilistic model that employs a graphical structure to solve complex problems. It provides knowledge about a domain in which each node represents a set of random variables, and each edge represents the statistical relationship between these variables [27]. In addition, each node has a conditional probability distribution associated with the corresponding random variables. The random variable can be conditions or states. BayesNet is used to represent probabilistic causal relationships [28].

3.4.3 K-Nearest Neighbor (KNN)

K-Nearest Neighbor (KNN) is a simple ML algorithm that considers the nearest neighbors. KNN can be applied on classification and regression problems. It is known as lazy learning or instance-based learning because the whole dataset is used in the testing phase. This makes testing slow and costly. KNN uses feature similarity and majority votes from the neighbors. The class with the highest number of votes is prediction result [29]. KNN has disadvantages such as computational complexity, an inability to deal missing values, and substantial storage requirements [30].

3.4.4 Support Vector Machine (SVM)

The Support Vector Machine (SVM) algorithm constructs a hyperplane in multi-dimensional space to maximize the margin between support vectors to improve classification accuracy. Support vectors are data points that lies on either side of the hyperplane. The number of dimensions is the number of features in the dataset. Hyperplanes are the decision boundaries that separate data points associated with different classes [31]. If the margin between a hyperplane to closest data points is large than the probability of misclassification of unseen data is low. SVM is unsuitable for large datasets and the training time can be long.

3.4.5 Random Forest (RF)

Random Forest (RF) is an ML classifier that employs multiple decision trees on different subsets of a dataset and averages the results to improve the prediction accuracy. At each tree node, a condition is compared with one or more features of the input data. Instead of depending on a single decision tree, RF collects predictions from multiple trees to obtain the results [32]. Each tree votes for a particular class and the class with the maximum number of votes is the predicted class. RF performs very well on imbalanced datasets and the number of classification errors is low.

3.4.6 Sequential Minimal Optimization (SMO)

The LibSVM library is used to implement the Sequential Minimal Optimization (SMO) algorithm. SMO overcomes the Quadratic Programming (QP) problem that arises during training of SVM [33]. SMO divides a large QP problem into smaller QP sub-problems and solves them analytically. As a consequence, SMO has a lower execution time than SVM. It is a robust classification algorithm but is costly to train.

3.4.7 Simple Logistic

Linear logistic regression is the foundation of the Simple Logistic ML algorithm. It is commonly used for classification tasks due to its simple structure [34]. Simple Logistic performs very well when the data is linear, but performance can be poor with non-linear or complex data. It cannot handle missing data in a dataset, i.e. feature values not provided in the dataset.

Chapter 4

Performance Evaluation

In this chapter, experimental results for seven supervised ML algorithms, namely Multilayer Perceptron (MLP), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Random Forest (RF), Bayesian Network (BayesNet), Sequential Minimal Optimization (SMO) and Simple Logistic are given. The experiments were conducted on a personal computer with the hardware and software parameters in Table 4.1.

Item	**Specification**
Manufacturer	Asus Tek Computer Inc.
Model	Asus Prime Z390-A
Operating System	Windows 10 Professional
System Type	64-bit Operating System, x64-based processor
Processor Type	Intel(R) Core (TM) i7-9700K CPU
Installed Memory (RAM)	32 GB @ 3200 MHz
Processor Speed	3.60 GHz
Number of Cores	8
Number of Threads	8
Machine Learning Tool	WEKA version 3.9.4

Table 4.1: The hardware and software parameters.

4.1 Evaluation Metrics

ML models work on the principle of constructive feedback. Models are built, evaluated based on metrics, and then modified to improve the performance. Several metrics are employed for regression, classification, clustering, and ranking. The performance met-

rics used in this project to evaluate the ML classifiers for intrusion detection are described below.

Precision is the ratio of true positive to the sum of false positive and true positive

$$\frac{tp}{tp + fp}$$

where true positive (tp) is the number of attacks classified correctly and false positive (fp) is the number of incorrect classification of benign as an attack.

Recall is the ratio of true positive to the sum of false negative and true positive

$$\frac{tp}{tp + fn}$$

where false negative (fn) is the number of incorrect classification of an attack as benign.

Accuracy is the number of correct classifications either as an attack or benign out of all instances in a dataset

$$\frac{(tn + tp)}{(tn + tp + fn + fp)}$$

where true negative (tn) is the number of correct classifications of benign as benign.

F-Measure is the weighted harmonic mean of the recall and precision

$$\frac{2tp}{2tp + fp + fn}$$

Execution Time is the time taken to train and test the classification model.

In this chapter, the accuracy, precision, recall, and F-measure are expressed as percentages.

4.2 Performance of the Classifiers

Data from the CSE-CIC-IDS2018 dataset for seven attacks and benign traffic are used to evaluate the ML models. The data distribution plays a vital role in model training. A balanced dataset can produce better results in classification problems as an imbalanced dataset may result in biased predictions. Thus, it is important to balance the dataset before model training. Undersampling and oversampling and their combination are techniques used to obtain a balanced dataset and these are considered here.

4.2.1 Performance without Oversampling or Undersampling using 5-fold Cross-validation

In this section, the dataset without oversampling or undersampling is considered. The number of instances in the benign class is 13484708, and a subset of 100,000 is considered. The numbers of dataset instances are presented in Table 4.2. Table 4.3 shows the performance evaluation results of the ML classifiers. The accuracy, precision, recall and F-measure for the KNN classifier are the highest at 97.8, 97.8, 97.8, 97.8, followed by RF at 97.7, 97.7, 97.7, 97.7, SVM at 95.3, 95.3, 95.3, 95.2, BayesNet at 94.8, 95.2, 94.8, 95.0, Simple Logistics at 88.5, 87.5, 88.5, 87.6, MLP at 86.4, 89.3, 86.5, 86.7, SMO at 84.5, 82.9, 84.5, 82.6. In terms of execution time, BayesNet had the lowest at 61 s followed by SMO at 267 s, RF at 381 s and SVM had the longest at 11280 s. Since there is a minimal performance difference between KNN and RF, the latter provides the best tradeoff between performance and execution time.

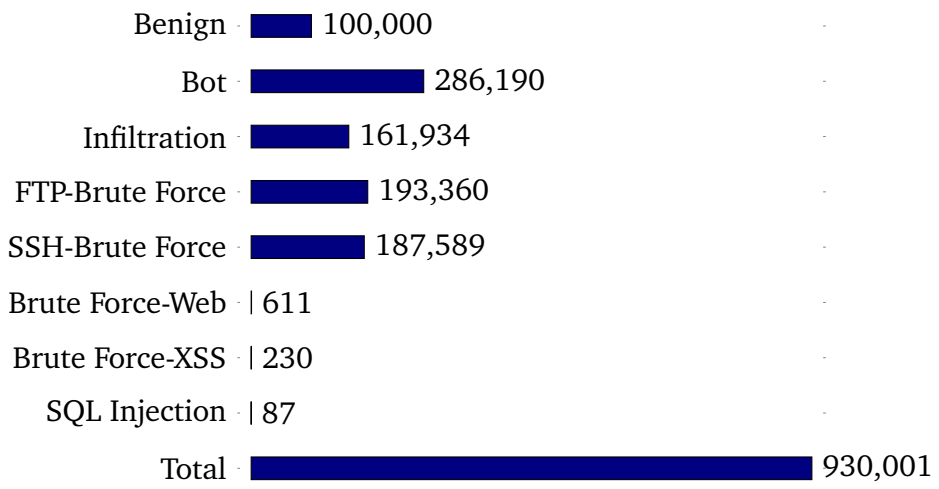

Table 4.2: The numbers of dataset instances without oversampling or undersampling.

Classifier	Accuracy	Precision	Recall	F-Measure	Execution Time (s)
MLP	86.4	89.3	86.5	86.7	853
SVM	95.3	95.3	95.3	95.2	11280
KNN	97.8	97.8	97.8	97.8	8962
Random Forest	97.7	97.7	97.7	97.7	381
BayesNet	94.8	95.2	94.8	95.0	61
SMO	84.5	82.9	84.5	82.6	267
Simple Logistic	88.5	87.5	88.5	87.6	1310

Table 4.3: Performance results without oversampling or undersampling with 5-fold cross-validation.

4.2.2 Performance with Oversampling using 5-fold Cross-validation

Oversampling is used to balance the dataset by replicating instances of the minority classes. Random oversampling is a common approach which duplicates instances but this can result in overfitting for some classification models. In [35], the Synthetic Minority Oversampling TEchnique (SMOTE) was proposed to mitigate this overfitting problem. This approach has been shown to be effective in a wide range of applications. It generates new synthetic instances using the feature similarity spaces between specific instances [36]. To make a synthetic instance, the K-nearest neighbours from minority data points are chosen and linear interpolation is used to create a new minority instance between

these data points. In this project, SMOTE is used for oversampling. It was employed independently to Brute Force-Web, Brute Force-XSS, and SQL Injection classes. The new synthetic instances were generated using SMOTE iteratively for the minority classes with a percentage parameter between 100 and 200. The numbers of dataset instances after oversampling are presented in Table 4.4, and the performance of the classifiers is given in Table 4.5. The accuracy, precision, recall and F-measure for RF are 98.5, 98.5, 98.5, 98.5, followed with KNN at 98.1, 98.1, 98.1, 98.1, BayesNet with 94.9, 95.0, 94.9, 95.0, SMO with 75.1, 77.0, 75.1, 72.4, Simple Logistic with 73.7, 75.5, 73.8, 72.5, and MLP with 53.9, 72.8, 53.9, 55.3 respectively. Further, BayesNet had the lowest execution time at 68 s followed by RF at 692 s, MLP at 1007 s and SMO at 1733 s whereas SVM had the longest execution time at 22294 s followed by KNN at 21606 s and Simple Logistic at 3701 s. There is a minimal performance difference between KNN and RF, so the latter provides the best tradeoff between performance and execution time.

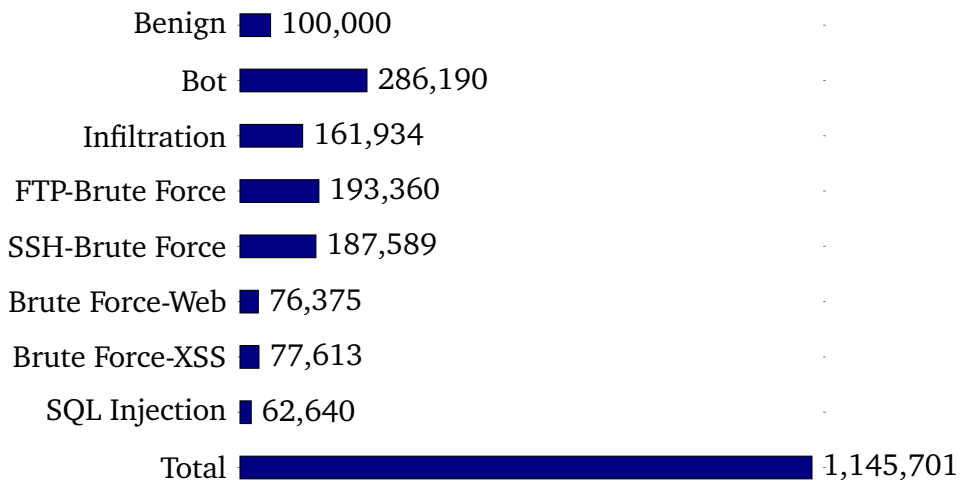

Table 4.4: The numbers of dataset instances after oversampling.

Classifier	Accuracy	Precision	Recall	F-Measure	Execution Time (s)
MLP	53.9	72.8	53.9	55.3	1007
SVM	92.3	93.1	92.4	92.2	22294
KNN	98.1	98.1	98.1	98.1	21606
Random Forest	98.5	98.5	98.5	98.5	692
BayesNet	94.9	95.0	94.9	95.0	68
SMO	75.1	77.0	75.1	72.4	1733
Simple Logistic	73.7	75.5	73.8	72.5	3701

Table 4.5: Performance results with oversampling and 5-fold cross-validation.

4.2.3 Performance with Undersampling using 5-fold Cross-validation

Undersampling removes instances of the majority classes to balance the data. Resample, an undersampling tool in WEKA, is used which randomly removes instances from the majority classes. The disadvantage of undersampling is that it can remove valuable information. Resample is employed independently to Benign, Bot, Infiltration, FTP-Brute Force, and SSH-Brute Force. Instances from the majority classes were removed using resample iteratively with a percentage parameter between 50 and 90. Table 4.6 shows the numbers of dataset instances and Table 4.7 gives the performance results for the ML classifiers using undersampling. The accuracy, precision, recall and F-measure for RF are at 98.7, 98.7, 98.7, 98.7, KNN are 97.1, 97.2, 97.2, 97.2, BayesNet are 92.8, 93.2, 92.9, 93.0, SVM are 92.3, 92.3, 92.3, 92.3, MLP are 89.5, 89.3, 89.5, 89.4, and Simple Logistic are 84.5, 84.0, 84.6, 84.1, and SMO is 79.7, 78.3, 79.7, 77.8. In terms of execution time BayesNet is the fastest at 68 s followed by RF at 692 s and MLP at 1007 s, whereas SVM had the longest execution time at 22294 s followed by KNN at 21606 s, Simple Logistic at 3701 s, and SMO at 1733 s. There is minimal performance difference between RF and KNN so RF provides the best tradeoff between performance and execution time.

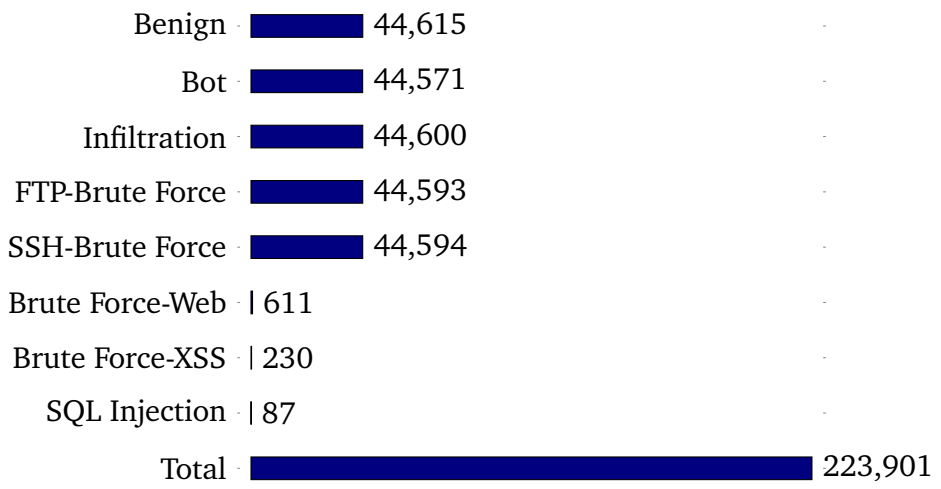

Table 4.6: The numbers of dataset instances after undersampling.

Classifier	Accuracy	Precision	Recall	F-Measure	Execution Time (s)
MLP	89.5	89.3	89.5	89.4	192
SVM	92.3	92.3	92.3	92.3	1319
KNN	97.1	97.2	97.2	97.2	493
Random Forest	98.7	98.7	98.7	98.7	79
BayesNet	92.8	93.2	92.9	93.0	21
SMO	79.7	78.3	79.7	77.8	70
Simple Logistic	84.5	84.0	84.6	84.1	621

Table 4.7: Performance results with undersampling and 5-fold cross-validation.

4.2.4 Performance with Oversampling and Undersampling using 5-fold Cross-validation

In this section, both oversampling and undersampling are employed to evaluate the ML algorithms. SMOTE was employed independently to the Brute Force-Web, Brute Force-XSS, and SQL Injection classes whereas resample was employed independently to Benign, Bot, Infiltration, FTP-Brute Force, and SSH-Brute Force. New synthetic instances were generated using SMOTE iteratively for the minority classes with a percentage parameter between 100 and 200 while instances from the majority classes were removed using resample iteratively with a percentage parameter between 50 and 90. The numbers of instances of each class are shown in Table 4.8, and the performance of the ML classifiers

is given in Table 4.9. The accuracy, precision, recall and F-measure of RF are 98.7, 98.8, 98.8, 98.8, followed by KNN with 97.7, 97.7, 97.7, 97.7, BayesNet with 93.5, 93.6, 93.5, 93.5, SVM 90.5, 91.0, 90.5, 90.4, MLP with 81.7, 84.0, 81.8, 81.4, Simple Logistic with 73.7, 77.4, 73.8, 73.7, and SMO with 71.1, 74.7, 71.1, 70.7. BayesNet had the lowest execution time at 29 s while SVM had the highest at 3321 s. However, there is minimal performance difference between RF and KNN so RF provides the best tradeoff between performance and execution time.

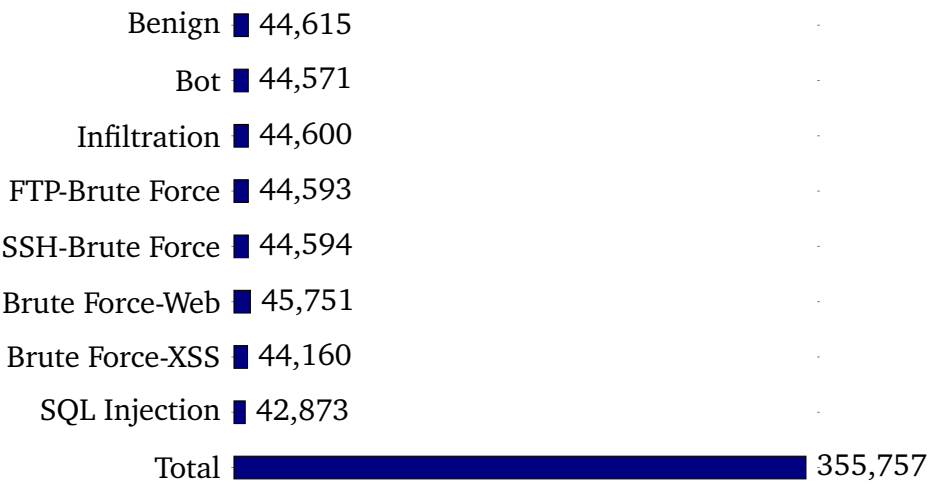

Table 4.8: The numbers of dataset instances after oversampling and undersampling.

Classifier	Accuracy	Precision	Recall	F-Measure	Execution Time (s)
MLP	81.7	84.0	81.8	81.4	307
SVM	90.5	91.0	90.5	90.4	3321
KNN	97.7	97.7	97.7	97.7	1612
Random Forest	98.7	98.8	98.8	98.8	109
BayesNet	93.5	93.6	93.5	93.5	29
SMO	71.1	74.7	71.1	70.7	446
Simple Logistic	73.7	77.4	73.8	73.7	973

Table 4.9: Performance results with oversampling and undersampling and 5-fold cross-validation.

4.2.5 Performance with Oversampling and Undersampling using 5-fold Cross-validation with Reduced Number of Instances

A smaller dataset with oversampling and undersampling is considered with 5-fold cross-validation. SMOTE was employed independently to the Brute Force-Web, Brute Force-XSS, and SQL Injection classes whereas resample was employed independently to Benign, Bot, Infiltration, FTP-Brute Force, and SSH-Brute Force. New synthetic instances were generated using SMOTE iteratively for the minority classes with a percentage parameter between 100 and 200 while instances from the majority classes were removed using resample iteratively with a percentage between 50 and 90. The numbers of dataset instances is shown in Table 4.10. and the performance results is shown in Table 4.11. The accuracy, precision, recall and F-measure for RF are 98.3, 98.3, 98.3, 98.3, KNN are 96.6, 96.6, 96.6, 96.6, BayesNet are 92.6, 92.8, 92.6, 92.7, SVM are 86.8, 88.2, 86.8, 86.8, MLP are 83.3, 83.0, 83.3, 83.0, Simple Logistic are 72.9, 76.1, 72.9, 72.7, and SMO are 70.4, 75.4, 70.4, 69.9. The execution time is reduced as the number of instances reduced in dataset. BayesNet had the lowest execution time at 5 s while Simple Logistic had the highest at 145 s. Again, RF provides the best tradeoff between performance and execution time.

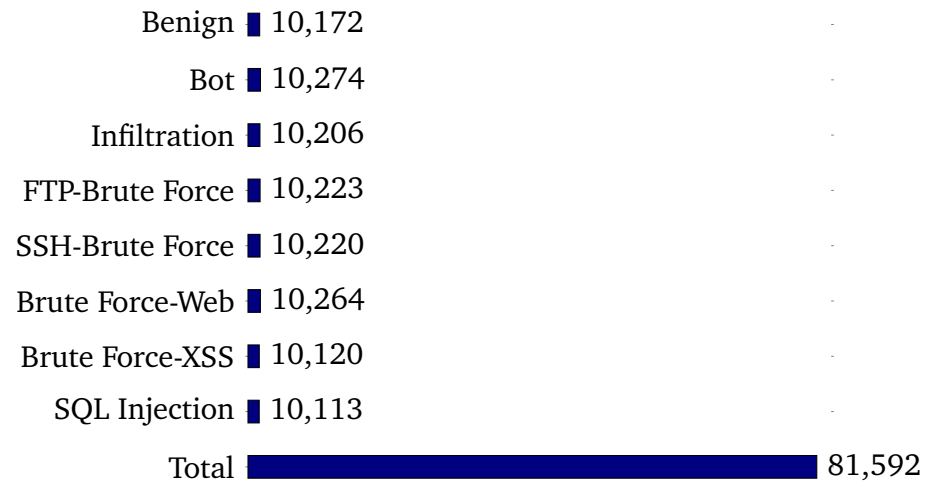

Table 4.10: The numbers of dataset instances after oversampling and undersampling.

Classifier	Accuracy	Precision	Recall	F-Measure	Execution Time (s)
MLP	83.3	83.0	83.3	83.0	41
SVM	86.8	88.2	86.8	86.8	104
KNN	96.6	96.6	96.6	96.6	36
Random Forest	98.3	98.3	98.3	98.3	19
BayesNet	92.6	92.8	92.6	92.7	05
SMO	70.4	75.4	70.4	69.9	41
Simple Logistic	72.9	76.1	72.9	72.7	145

Table 4.11: Performance results with oversampling and undersampling and 5-fold cross-validation.

4.2.6 Performance without Oversampling or Undersampling using 10-fold Cross-validation

The dataset without undersamping and oversampling is considered with 10-fold cross-validation. The numbers of dataset instances considered are shown in Table 4.2 and the performance results are given in Table 4.10. The accuracy, precision, recall and F-measure for RF are 98.8, 98.8, 98.8, 98.8, KNN are 97.7, 97.7, 97.7, 97.7, BayesNet are 94.8, 95.2, 94.8, 95.0, SVM are 95.3, 95.4, 95.3, 95.2, MLP are 81.4, 85.1, 81.4, 82.3, Simple Logistic are 88.5, 87.5, 88.5, 87.6, and SMO are 84.3, 82.6, 84.3, 82.4. BayesNet had the lowest execution time at 85 s while SVM had the highest at 24611 s. Thus, RF provides best tradeoff between performance and execution time.

Classifier	Accuracy	Precision	Recall	F-Measure	Execution Time (s)
MLP	81.4	85.1	81.4	82.3	1494
SVM	95.3	95.4	95.3	95.2	24611
KNN	97.8	97.8	97.8	97.8	12451
Random Forest	98.3	98.3	98.3	98.3	756
BayesNet	94.8	95.2	94.8	95.0	85
SMO	84.3	82.6	84.3	82.4	476
Simple Logistic	88.5	87.5	88.5	87.6	1981

Table 4.12: Performance results without oversampling or undersampling with 10-fold cross-validation.

4.2.7 Performance with Undersampling and Oversampling using 10-fold Cross-validation with Reduced Number of Instances

A smaller dataset with oversampling and undersampling is now considered with 10-fold cross-validation. The numbers of dataset instances considered are shown in Table 4.10 and the performance results are shown in Table 4.13. The accuracy, precision, recall and F-measure for RF are 98.5, 98.5, 98.5, 98.5, KNN are 96.8, 96.8, 96.8, 96.8, BayesNet are 92.7, 92.9, 92.7, 93.8, SVM are 87.0, 88.2, 87.0, 87.0, MLP are 82.9, 83.3, 82.9, 82.5, Simple Logistic are 73.3, 76.6, 73.3, 73.2, and SMO are 70.4, 75.2, 70.4, 69.9. The execution time is reduced as the number of instances reduced in dataset. BayesNet had the lowest execution time at 7 s while Simple Logistic had the highest at 411 s. Again, RF provides the best tradeoff between performance and execution time.

Classifier	Accuracy	Precision	Recall	F-Measure	Execution Time (s)
MLP	82.9	83.3	82.9	82.5	90
SVM	87.0	88.2	87.0	87.0	135
KNN	96.8	96.8	96.8	96.8	37
Random Forest	98.5	98.5	98.5	98.5	40
BayesNet	92.7	92.9	92.7	93.8	07
SMO	70.4	75.2	70.4	69.9	59
Simple Logistic	73.3	76.6	73.3	73.2	411

Table 4.13: Performance results with oversampling and undersampling and 10-fold cross-validation.

4.3 Discussion

KNN and RF performed better than the other classifiers in terms of accuracy, precision, recall, and F-measure without oversampling or undersampling using 5-fold cross-validation. However, in terms of execution time, BayesNet was the fastest at 61 s whereas SVM was slowest at 11280 s. The accuracy of SVM is 3% better than that of SVM with oversampling (Table 4.5) and with undersampling (Table 4.7), and 4.8% better than with both oversampling and undersampling (Table 4.9).

With oversampling, RF had the highest accuracy at 98.5% whereas in terms of execution time, BayesNet was the fastest at 68 s. SVM had similar results in terms of accuracy at 92.3% with oversampling (Table 4.5) and with undersampling (Table 4.7) but the execution time was longer at 22294 s with oversampling as the dataset size is larger. The worst performance in terms of accuracy, precision, recall, and F-measure was obtained with the MLP classifier. The execution time of the ML classifiers with oversampling increased by 97% for SVM and 141% for KNN versus without oversampling and undersampling.

With undersampling, again RF had the highest accuracy at 98.7% and the execution time was lower than previously at 79 s due to the smaller data size. BayesNet execution time was 21 s which was fastest among the classifiers and SVM was slowest at 1319 s. MLP was 3.1% better in terms of accuracy than MLP without oversampling or undersampling (Table 4.3), and 35.6% with oversampling (Table 4.5). Undersampling the majority class instances reduced the dataset size which reduced the execution time of the ML classifiers by 88% for SVM and 94% for KNN than without oversampling or undersampling (Table 4.3).

With both oversampling and undersampling, again RF had the highest accuracy at 98.8%. However, in terms of execution time BayesNet was fastest at 29 s and SVM was the slowest at 3321 s. A balanced dataset was obtained with both sampling methods but the accuracy of MLP, SMO, and Simple Logistic classifiers decreased by 7.8%, 8.6%, and 10.8% respectively compared to with undersampling (Table 4.7).

Without oversampling or undersampling using 5-fold and 10-fold cross-validation, the performance of the classifiers is similar except for a 5% performance degradation with the MLP classifier and the execution time was longer with 10-fold cross-validation (Table 4.12). Similar performance results were obtained with oversampling and undersampling using 5-fold and 10-fold cross-validation except for the execution time. Overall, the execution time was approximate linear with the dataset size. SMO performed worst with oversampling and undersampling using 10-fold cross-validation. SVM had the highest execution time in all cases followed by KNN. The most robust model in all cases was RF followed by KNN, BayesNet and SVM (except execution time), while the least robust were MLP, Simple Logistic, and SMO.

Chapter 5

Conclusion and Future Work

In this project, Machine Learning (ML) algorithms were used to detect seven network attacks, namely Infiltration, FTP-BruteForce, SSH-BruteForce, Brute Force-Web, Brute Force-XSS, SQL Injection, Bot, and Benign class. The CSE-CIC-IDS2018 dataset from the Canadian Institute for Cybersecurity (CIC) and the Communications Security Establishment (CSE) was employed. Oversampling using SMOTE, and undersampling using Resample were used to balance the dataset. Seven supervised ML classifiers, namely MLP, BayesNet, KNN, SVM, RF, SMO and Simple Logistic, were employed to classify the attacks and benign class. Five-fold and ten-fold cross-validation was used to train and test the classifiers. The performance metrics accuracy, precision, recall and F-measure were used for evaluation. The results obtained show that RF had the best performance and an average execution time. SVM had the highest execution time. BayesNet had the lowest execution time and the performance results was slightly lower than with RF and KNN. Overall, it is concluded that RF provides the best tradeoff between execution time and performance.

5.1 Future Work

For future work, other classifiers can be considered. Further, unsupervised ML algorithms can be employed with the CSE-CIC-IDS2018 dataset. For data splitting, techniques other than k-fold cross-validation such as percentage split can be employed. WEKA supports supervised and unsupervised classifiers written in Python (ScikitLearnClassifier), so it can be used to implement other algorithms and the results compared to those in this report.

Bibliography

[1] *The world's most valuable resource is no longer oil, but data.* The Economist, 2017/05/06.

`https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data/`

[2] Phil Goodwin, Andrew Smith, and Robert Westervelt. *Addressing cyber protection and data protection holistically.* 2019/06.

`https://dl.acronis.com/u/rc/WP_IDC_Acronis_Cyber_Protection_EN-US_190626.pdf/`

[3] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. *Toward generating a new intrusion detection dataset and intrusion traffic characterization.* International Conference on Information Science and Security. vol. 1, pp. 108-116, Funchal, Madeira, Portugal, Jan. 2018.

[4] Amirhossein Gharib, Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. *An evaluation framework for intrusion detection dataset*. IEEE International Conference on Information Science and Security. pp. 1-6, Pattaya, Thailand, Dec. 2016.

[5] Almseidin Mohammad, Maen Alzubi, Szilveszter Kovacs, and Mouhammad Alkasassbeh. *Evaluation of machine learning algorithms for intrusion detection system.* IEEE International Symposium on Intelligent Systems and Informatics. pp. 277-282, Subotica, Serbia, Sep. 2017.

[6] Prachi Chaudhary. *Usage of machine learning for intrusion detection in a network.* International Journal of Computer Networks and Applications. vol. 3, no. 6, pp. 139-147, Nov.-Dec. 2016.

[7] Ansam Khraisat, Iqbal Gondal, and Peter Vamplew. *An anomaly intrusion detection system using C5 decision tree classifier.* Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer-Verlag, Lecture Notes in Computer Science, vol. 11154, pp. 149-155, Melbourne, Australia, Jun. 2018.

[8] Christian Kreibich and Jon Crowcroft. *Creating intrusion detection signatures using honeypots.* ACM SIGCOMM Computer Communication Review. vol. 34, pp. 51-56, Jan. 2004.

[9] Giovanni Vigna and Richard A. Kemmerer. *NetSTAT: A network-based intrusion detection system.* Journal of Computer Security. vol. 7, pp. 37-71, Jan. 1999.

[10] Martin Roesch. *Snort: Lightweight intrusion detection for networks.* Systems Administration Conference vol. 99, no. 1, pp. 229-238, Seattle, Washington, USA, Dec. 1999.

[11] *Symantec, Internet Security Threat Report.* vol. 22. Apr. 2017.

`https://docs.broadcom.com/doc/istr-22-2017-en/`

[12] Ammar Alazab, Michael Hobbs, Jemal Abawajy, Moutaz Alazab. *Using feature selection for intrusion detection system.* IEEE International Symposium on Communications and Information Technologies. pp. 296-301, Gold Coast, QLD, Australia, Oct. 2012.

[13] *FORTINET. DoS vs DDoS.*

`https://www.fortinet.com/resources/cyberglossary/dos-vs-ddos/`

[14] Alma D. Lopez, Asha P. Mohan, and Sukumaran Nair. *Network traffic behavioral analytics for detection of DDoS attacks. SMU Data Science Review.* vol. 2, no. 1, 2019. https://scholar.smu.edu/datasciencereview/vol2/iss1/14.

[15] *REDLEGG. REDLEGG BLOG.*

`https://www.redlegg.com/blog/3-tools-to-test-denial-of-service-vulnerability/`

[16] *HOIC (High Orbit Ion Cannon).*

`https://www.radware.com/security/ddos-knowledge-center/ddospedia/hoic-high-orbit-ion-cannon/`

[17] *A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018).*

`http://www.unb.ca/cic/datasets/ids-2018.html/`

[18] Ayon Dey, Jyoti Singh, and Neeta Singh. *Analysis of supervised machine learning algorithms for heart disease prediction with reduced number of attributes using principal component analysis.* International Journal of Computer Applications. vol. 140, no. 2, pp. 27-31, Apr. 2016.

[19] *Supervised Learning By: IBM Cloud Education.* Aug. 2020.

`https://www.ibm.com/cloud/learn/supervised-learning/`

[20] *DataRobot.Unsupervised machine learning.*

https://www.datarobot.com/wiki/unsupervised-machine-learning/

[21] Prashant Gupta. *Cross-validation in machine learning.* Jun. 2017.

https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f/

[22] Sourav Sen Gupta. *Decision trees. Towards data science.* 2017.

https://towardsdatascience.com/decision-trees-in-machinelearning/

[23] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, Chennai, India, Oct. 2016.

[24] Zanariah Zainudin, Siti Mariyam Shamsuddin, and Shafaatunnur Hasan. *Deep learning for image processing in WEKA environment.* International Journal of Advances in Soft Computing and its Applications. vol. 11, no. 1, pp. 1-21, 2019.6.

[25] Remco R. Bouckaert, Eibe Frank, Mark Hall, Richard Kirkby, Peter Reutemann, Alex Seewald, and David Scuse. *WEKA manual version 3-8-1.* University of Waikato, Hamilton, New Zealand, Dec. 2016.

[26] Fariba Haddadi, Sara Khanchi, Mehran Shetabi, and Vali Derhami. *Intrusion detection and attack classification using feed-forward neural network.* IEEE International Conference on Computer and Network Technology. pp. 262-266, Bangkok, Thailand, Apr. 2010

[27] Sukhan Lee, Shunichi Shimoji. *BAYESNET: Bayesian classification network based on biased random competition using Gaussian kernels.* IEEE International Conference on Neural Networks. pp. 1354-1359, San Francisco, CA, USA, Mar. 1993.

[28] Eugene Charniak. *Bayesian networks without tears. AI Magazine.* vol. 12, no. 4, pp. 50-50, Dec. 1991.

[29] Sawsan Kanj, Fahed Abdallah, Thierry Denoeux, and Kifah Tout. *Editing training data for multi-label classification with the k-nearest neighbor rule. Pattern Analysis and Applications.* vol. 19, no. 1, pp. 145-161, Feb. 2016.

[30] Zhongheng Zhang. *Introduction to machine learning: K-nearest neighbors.* Annals of Translational Medicine. vol. 4, no. 11, pp. 218-218, Jun. 2016.

[31] Wun-Hwa Chen, Sheng-Hsun Hsu, and Hwang-Pin Shen. *Application of SVM and ANN for intrusion detection.* Computers and Operations Research. vol. 32, no. 10, pp. 2617-2634, Oct. 2005.

[32] Mouhammd Alkasassbeh, Ghazi Al-Naymat, Ahmad Hassanat, and Mohammad Almseidin. *Detecting distributed denial of service attacks using data mining techniques.* International Journal of Advanced Computer Science and Applications. vol. 7, no. 1, pp. 436-445, Jan. 2016.

[33] John C. Platt. *Fast training of SVMs using sequential minimal optimization Advances in kernel methods-support vector learning.* MIT Press, Cambridge, pp. 185-208. Jan. 1998.

[34] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. *A comparison of machine learning techniques for phishing detection.* Proceedings of the Anti-phishing Working Groups Annual eCrime Researchers Summit. pp. 60-69, Pittsburgh, PA, Oct. 2007.

[35] Miriam Seoane Santos, Jastin Pompeu Soares, Pedro Henrigues Abreu, Helder Araujo, and Joao Santos. *Cross-validation for imbalanced datasets: avoiding overoptimistic and overfitting approaches.* IEEE Computational Intelligence Magazine. vol. 13, no. 4, pp. 59-76, Oct. 2018.

[36] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. *SMOTE: Synthetic minority over-sampling technique.* Journal of Artificial Intelligence Research. vol. 16, no. 1, pp. 321-357, Jan. 2002.