MACHINE LEARNING:
APPLICATION TO THE SCADA SYSTEM

LEE SHENG KAI

UNIVERSITI TUNKU ABDUL RAHMAN

MACHINE LEARNING:
APPLICATION TO THE SCADA SYSTEM

LEE SHENG KAI

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electrical and Electronic Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2019

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : LEE SHENG KAI

ID No. : 1404202

Date : _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"MACHINE LEARNING: APPLICATION TO THE SCADA SYSTEM"** was prepared by **LEE SHENG KAI** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electrical and Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : TS. DR. YAP WUN SHE

Date : _____

ABSTRACT

An intrusion detection system is employed to protect supervisory control and data acquisition system from cyber-physical attacks. The effectiveness of the employed intrusion detection system relies on the accuracy in predicting different cyber-attacks. Different machine learning models had been proposed to increase the accuracy of an intrusion detection system in predicting different cyber-attacks. This is also known as multiclass classification problem. Most of the existing approaches remove features of different cyber-attacks and thus limit the number of predicted types of attacks which leads to a simpler multiclass classification problem. To make matters worse, inappropriate or artificial network data had been used to evaluate the accuracy of the proposed machine learning methods. The aforementioned concerns question the validity of existing machine learning models in predicting different cyber-attacks. In this project, wrapper-based feature selection technique with best-first search algorithm is used to consider all features of different cyber-attacks such that the trained machine learning classifier can be used to predict all types of cyber-attacks. In addition, ensemble method that combines two different machine learning models is performed to evaluate its effectiveness in predicting all different types of cyber-attacks. Experiments are conducted on three publicly recognised datasets, i.e., UNSW-NB15, ISCX 2012 and NSL-KDD. The results show that wrapper-based feature selection technique with best-first search algorithm is always effective to improve the accuracy of multiclass classification. On the other hand, ensemble learning is able to enhance the multiclass classification model only if the ensemble model is constructed with the correct combination of base learners or models. Thus, this final year project proposes to use feature extraction and ensemble learning on conventional machine learning algorithms improving the prediction performance. Conventional machine learning algorithms are the focus as these algorithms work well with the structured data provided in the aforementioned dataset. Lastly, as compared to the existing literature which mainly measures the accuracy of multiclass classification against six types of cyber-attacks, the multiclass classification model proposed in this project is able to predict up to ten different types of cyber-attacks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

FN	False Negative
FP	False Positive
IDS	Intrusion Detection System
J48	J48 Decision Tree
KNN	*K*-nearest Neighbour
NB	Naïve Bayes
RF	Random Forest without Tie-Breaking Capability
RF-BT	Random Forest with Tie-Breaking Capability
SCADA	Supervisory Control and Data Acquisition
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
WEKA	Waikato Environment for Knowledge Analysis Software Suite

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

In the mid-20th century, many manufacturing or industrial plants were greatly reliant on personnel to manually control and monitor different entities on-site. As the manufacturing processes are getting more complex and the industrial floors are getting bigger in physical size, supervisory control and data acquisition (SCADA) which contains software and hardware components was developed. SCADA allows industrial players to monitor and control different entities locally or at remote locations (Boyer, 2004). As SCADA is capable to control and monitor different entities connected with each other, SCADA tends to be the target of attackers. Shitharth and Winston (2015) listed vulnerabilities reported in various SCADA systems, including eavesdropping (Mo, Chabukswar and Sinopoli, 2014), SQL injection attack (Zhang, et al., 2016), denial-of-service attack (Barbosa, 2014), identity spoofing (Zhang, et al., 2016), man-in-the-middle attack (Maynard, McLaughlin and Haberler, 2014), related-key attack (Beaulieu, et al., 2017) and malware attack (Akhtar, Gupta and Yamaguchi, 2018). As SCADA systems are widely used in power plants and industry, measures are taken by governments and private companies to secure SCADA against attacks, in both cyber and physical environments. One of the measures taken is to monitor cyber-physical systems for malicious activity or policy violations. This system is coined as intrusion detection system (Rao and Nayak, 2014).

Generally, there exist two sources of audit data for intrusion detection system (IDS), namely network-based and host-based. Network-based IDS is implemented at several strategic points within the network to monitor the inflow and outflow of traffic (Barbosa, 2014) while host-based IDS is implemented on individual devices to analyse the system logs (Mitchell and Chen, 2014). Figure 1.1 illustrates the two locations where network-based IDS and host-based IDS collect the network data (Fogie and Peikari, 2002). Network-based IDS is the focus of this final year project since one does not need to consider the extension of network from time to time.

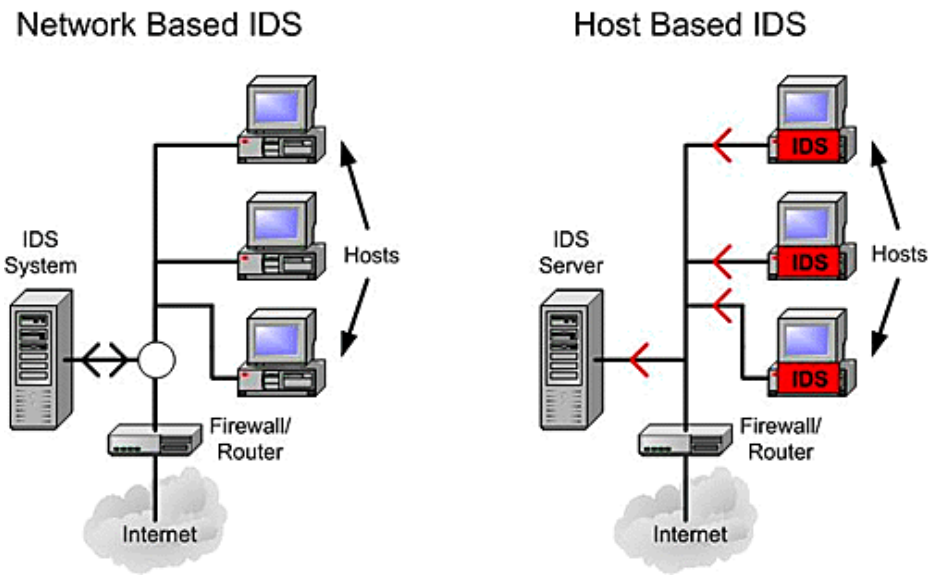


Figure 1.1: Network-Based IDS versus Host-Based IDS (Fogie and Peikari, 2002)

Network-based IDS analyses the network traffic and then subsequently maps the network traffic with the collection of identified attacks. Once abnormality is detected, the warning will be sent to the network administrator for further actions. There are generally two approaches of detection, namely misuse-based and anomaly-based detection approaches.

Misuse-based detection identifies the intrusion based on the pre-determined rules, also known as attack characteristics or signatures (Erez and Wool, 2015). Some examples of the signatures include the byte sequences in network traffic and identified malicious commands used by malware. Misuse-based intrusion detection although proved to be accurate, but it cannot detect the newer types of unseen attack. In contrast, an anomaly-based detection is dependent on the overall behaviour of the system where an attacker's behaviour is observably unusual as compared to that of a legitimate user by recognising a deviation from normal system behaviour (Erez and Wool, 2015). Nevertheless, anomaly-based detection system still suffers from high false positives issue, which treats a legitimate activity as a malicious attack (Barbosa, 2014). In addition, Erez and Wool (2015) claimed that anomaly-based detection is sensitive to noise.

There are copious of researches in both misuse and anomaly-based detection methods and it can be found that different approaches are proposed in the literature. Among the proposed approaches, machine learning approach shows a promising trend in tackling cybersecurity concerns, especially in the application of IDS (Ahmad,

Jian and Anwar, 2018). Specifically, supervised machine learning is well-adopted in building IDS models for the classification of normal and different attack instances.

1.2 Problem Statement

Although there are abundant researches in the academia, the industry has yet to employ a well-established machine learning based IDS model. The field of cybersecurity which utilises advanced information technology is still undergoing thorough experiments by thousands of researchers. However, machine learning based IDS is still in immature or experimentation stage.

Datasets play an important role to train the proposed machine learning based IDS such that the trained IDS can classify different types of attacks with higher accuracy. However in the literature, obsolete or questionable datasets are always selected as inputs to the proposed machine learning model. This leads to a lower success rate of intrusion detection because the attack patterns were outdated, so as the normal traffic patterns.

Besides, it is observed that some researchers performed biased evaluation. For instance, the researchers omitted the prediction of minor attack types by removing certain features of the datasets in the hope of producing machine learning model that can classify a smaller number of different attacks with higher accuracy. With the same motive, some researchers targeted to detect certain attack types instead of taking the weighted average of all normal and attack categories. As a summary, the biased dataset caused the machine learning model to learn towards major types of attacks only while ignoring the other classes which only constitute a small portion of the dataset.

Since all different types of attacks need to be predicted and classified (also known as multiclass classification), the accuracy of the proposed machine learning model needs to be further improved. To improve the multiclass classification accuracy of a machine learning model, feature selection (i.e., how to select good features) and ensemble learning (i.e., make use of hybrid machine learning models to exploit advantages of each individual machine learning model) can be the possible solutions. These two techniques are not commonly found in the literature of machine learning based IDS. Thus, the effectiveness of these two methods remains unknown.

1.3 Aim and Objectives

This paper aims to study the effectiveness of feature selection and ensemble learning in improving the multiclass classification accuracy based on three up-to-date and commonly-used datasets without performing biased evaluation. The specific objectives are listed as follows:

- To study the effectiveness of the proposed machine learning models in classifying different types of attacks based on publicly recognised datasets
- To improve the multiclass classification accuracy of machine learning models by using wrapper-based feature selection technique with best-first search algorithm
- To improve the multiclass classification accuracy of machine learning models by using both feature selection and ensemble learning methods

This final year project contributes to the literature in three ways. Firstly, this project thoroughly examines the performance of the IDS models developed using existing conventional machine learning algorithms. In addition, this project analyses the effectiveness of wrapper-based feature selection technique in building better IDS models. Furthermore, this project also investigates the effectiveness of ensemble learning in improving the performance of IDS models.

1.4 Scope and Limitation of the Study

This project focuses on the multiclass classification problem for machine learning based intrusion detection systems. The IDS models are able to discriminate attack instances from normal instances. In addition, the IDS models are able to classify different types of cyber-attacks.

This project is carried out in Waikato Environment for Knowledge Analysis (WEKA), an application suite specialised in data mining and machine learning (Witten, et al., 2016). The base models of the IDS in this project are limited to conventional machine learning models in Weka. Besides, this project is limited by hardware resources. The computer used in this project is running Ubuntu 14.04, equipped with Intel Core i9-7920X CPU at 2.9 GHz and 64 GB RAM.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction of Supervised Machine Learning

Machine learning is employed in building models for predictive data analytics (Kelleher, Namee and D'arcy, 2015). A typical predictive problem requires gaining insights from a huge amount of data or instances. The collection of huge data is commonly known as a dataset, say set X. An instance from a dataset contains several attributes (also called as features), where each attribute describes the characteristics of one instance in the dataset. Let the set X be defined as $X = [x_1, x_2, x_3, \ldots, x_{n-1}, x_n]$ where the feature set X' be defined as $X' = [x_1, x_2, x_3, \ldots, x_{n-1}]$ which consists of $n-1$ attributes and the class or label y be defined as $y = x_n$. Ideally, a distinctive feature set X' maps to a certain output y, which becomes the prediction of the model.

The dataset will be split into two sets, namely training set and test set. The train-to-test ratio is usually 80:20 (Poria, et al., 2017) or 70:30 (Caruana, et al., 2015). If the proposed machine learning model is trained based on X' instead of X (which includes the label or outcome y), the model is considered as unsupervised machine learning; otherwise, the model is considered as supervised machine learning. Supervised machine learning is the focus of this final year project.

The objective of supervised machine learning is to develop a predictive function $f(X'_{train}) \rightarrow y_{train}$ from the training data. Then, the model will take in the feature set of test data X'_{test} and make prediction, or $y_{predict} = f(X'_{test})$. Finally, the predicted outcome will be compared to the actual outcome, or $y_{predict} \equiv y_{test}$ to measure the effectiveness of the underlying machine learning model. Notice that the test data will never be seen in the training process. In other words, the test set here is referred to the hold-out test set. The motive behind the hold-out process is to prevent peeking, which means the training process has already included the test data while developing the model. The general idea is that the predictive model needs to be measured on how well it can generalise beyond the training instances (Kelleher, Namee and D'arcy, 2015). Figure 2.1 shows the overall concept of developing and evaluating a machine learning model.

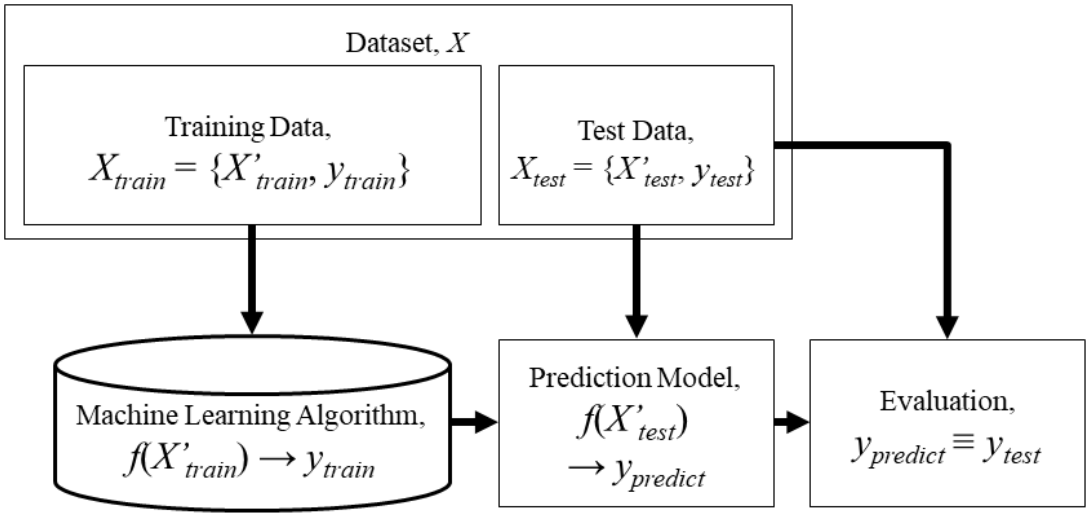


Figure 2.1: The Process of Developing and Evaluating a Machine Learning Model

In some cases, the dataset is not split into training or testing data directly. In *k*-fold cross validation, the dataset is divided into *k* equal segments. The process will train the model based on the *k*-1 parts, leaving one part to be the test data. Then, the training and testing processes repeat *k* times. The evaluation results will then be averaged by the number *k* (Kohavi, 1995). Figure 2.2 shows the train-test split of the dataset.

Figure 2.2: The Splitting of Dataset in *k*-fold Cross Validation. White Parts Indicate Training Data And Black Parts Indicate Testing Data (Kelleher, Namee and D'arcy, 2015)

The number $k = 10$ is commonly used (McLachlan, Do and Ambroise, 2005) but *k* can also be assigned to any positive integer greater than one. The *k*-fold cross validation technique allows validation for a small dataset (Mohammed, Khan and

Bashier, 2016). In addition, *k*-fold cross validation is useful for model selection, as it is capable to estimate the performance unbiasedly (Zhang and Yang, 2015).

2.2 Supervised Machine Learning Algorithms

There are various approaches to map the feature set to the expected outcome. For example, commonly used algorithms for supervised machine learning include Naïve Bayes (Rish, 2001), *k*-nearest neighbour (Liao and Vemuri, 2002), decision tree (Safavian and Landgrebe, 1991), random forest (Liaw and Wiener, 2001) and support vector machine (Sung and Mukkamala, 2003).

2.2.1 Naïve Bayes

Naïve Bayes is a simple probabilistic algorithm. It holds an assumption that the features are independent among each other. Grounded on Bayes' theorem, it calculates the probabilities of the events (i.e., classes in machine learning context) to happen. The class with the highest probability will be chosen as the decision. Formally, the rule of selecting the final outcome is called as maximum-a-posteriori rule. Recent studies show that Naïve Bayes classifier can be used in image recognition (Zhou, et al., 2015), anomaly detection (Swarnkar and Hubballi, 2016) and text categorisation (Tang, Kay and He, 2016). One advantage of using Naïve Bayes classifier is that it does not suffer from the curse of dimensionality (Kelleher, Namee and D'arcy, 2015). The curse of dimensionality refers to the phenomena when the unnecessary features cause the search space to increase dramatically, eventually, the generalisation will be slowed down and obstructed (Gheyas and Smith, 2010). Jadhav and Channe (2016) mentioned that Naive Bayes requires short training time as compared to other algorithms. It can also handle missing data due to its fast inference (Lowd and Domingos, 2005). Nevertheless, there are some disadvantages of using Naïve Bayes. For example, the naive assumption of the feature independence degrades the classifier performance (Rennie, et al., 2003). In this literature review, it is proved to have poor performance as compared to the other classification algorithm (Caruana and Niculescu-Mizil, 2006; Kim, Chung and Lee, 2017; Mocherla, Danehy and Impey, 2017).

2.2.2 *K*-Nearest Neighbour

K-nearest neighbour is one of the classification algorithms introduced in the early 1950s. It is now widely used in pattern recognition of the data. The instances in the *k*-nearest neighbour are represented spatially. If an instance consists of *n* features, then the data is represented as a point in *n*-dimensional spatial space (Syarif and Gata, 2017). The full training set spans the *n*-dimensional space with labelled class. Test data under prediction will also be described in the *n*-dimensional space. The prediction of the class is done by considering the proximate training data in the neighbourhood and calculating the Euclidean distances between the data points. The Euclidean distance in *n*-dimensional space between a training data $X = \{x_1, x_2, \ldots, x_n\}$ and a test data $Y = \{y_1, y_2, \ldots, y_n\}$ is given in Equation (2.1).

$$\text{dist}(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad (2.1)$$

The number *k* determines the number of closest training points to consider. If *k* is 1, then the test data is classified as the same class as the nearest training sample. Referring to the example in Figure 2.3, the dataset consists of two features ($n = 2$) and two classes. Red colour points indicate training instances for Class 1, whereas blue colour points indicate training instances for Class 2. Assuming the number $k = 4$, when a test data is introduced into the search space in $\mathbb{R}^2$, the algorithm finds the four shortest Euclidean distances (shown as arrows) from the test data to the training data. With reference to the labels of the four closest training records, the label of the test data can be deduced as Class 1.

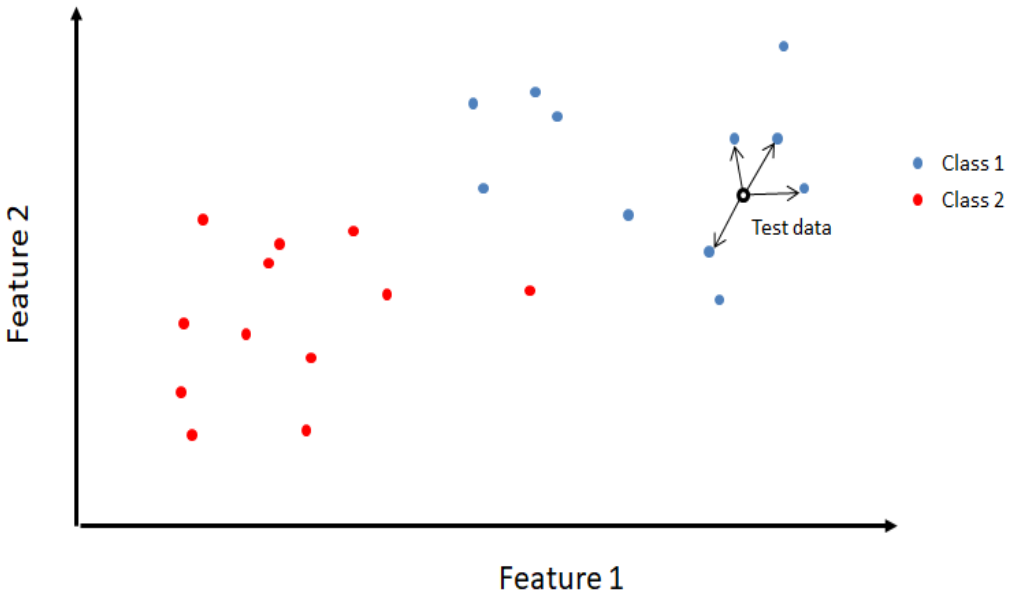


Figure 2.3: Illustration of *K*-Nearest Neighbour

One clear benefit of using *k*-nearest neighbour is the fast training time. It is also robust to noisy data (Bhatia, 2010). However, when the feature space is crowded or irrelevant, the prediction can be adversely affected (Syarif and Gata, 2017). It is also high in computational complexity, especially when the feature space is described in a very high dimension (Bhatia, 2010).

2.2.3 Decision Tree

A decision tree is a supervised machine learning technique and it is mainly used for classification task. Using divide and conquer rule, a decision tree consists of decision nodes and leaf nodes. The decision node defines a conditional test over an attribute, whereas the leaf node indicates the class (Ruggieri, 2002). Each path from the root node to the leaf node must follow a certain rule. Practically, a decision tree is generated according to the large training data, resulting in more branches and layers of the tree. For example, Figure 2.4 shows the generation of a decision tree. The outcome is to determine whether it is suitable to play outside by considering two features, i.e., weather outlook and humidity. These features contribute to the decision nodes and the final decisions are represented by the left nodes. According to the decision tree in Figure 2.4, one is allowed to play outside only if it is sunny outside and the humidity level is normal.

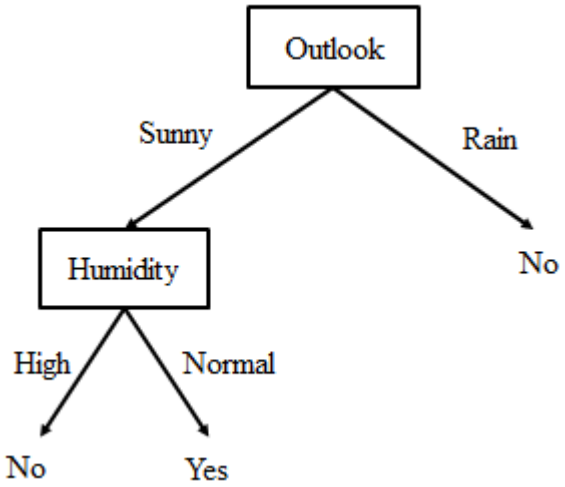


Figure 2.4: Illustration of Decision Tree

However, as the number of class categories increases, the classification accuracy decreases. This is known as overfitting (Ruggieri, 2002). Thus, pruning technique can be applied to improve the accuracy of the decision tree model (Patel and Upadhyay, 2012). Pruning reduces the size of the decision tree, prevents unnecessary branches and avoid overfitting. Nevertheless, the benefit of using a decision tree includes its inherent feature ranking technique in developing the tree model. It also has high interpretability to understand. There are different variants of decision tree generation, such as the ID3 (Hssina, et al., 2014), logistic model tree (Kabir and Zhang, 2016) and J48 (Aljawarneh, Yassein and Aljundi, 2017).

2.2.4 Random Forest

Random forest is an ensemble model of multiple decision trees (Kelleher, Namee and D'arcy, 2015). Each tree in the random forest represents the single decision tree model developed from subspace sampling. The subspace can be in terms of feature space or instance space. The combination of the trees is also known as bootstrap aggregation or bagging. In classification tasks, the final decision is determined by the majority voting of the trees. For instance, Figure 2.5 shows the working algorithms of a random forest. Suppose the target is to determine whether it is suitable to play outside, and there are three features, namely weather outlook, humidity and wind condition. Instead of generating the full decision tree, the random forest builds three decision trees, where each tree randomly samples two features. Given that the weather is sunny, the humidity level is high and the wind is weak, the trees are each

responsible to provide respective outputs. After aggregating the outputs, the random forest finally predicts that it is not suitable to play outside.

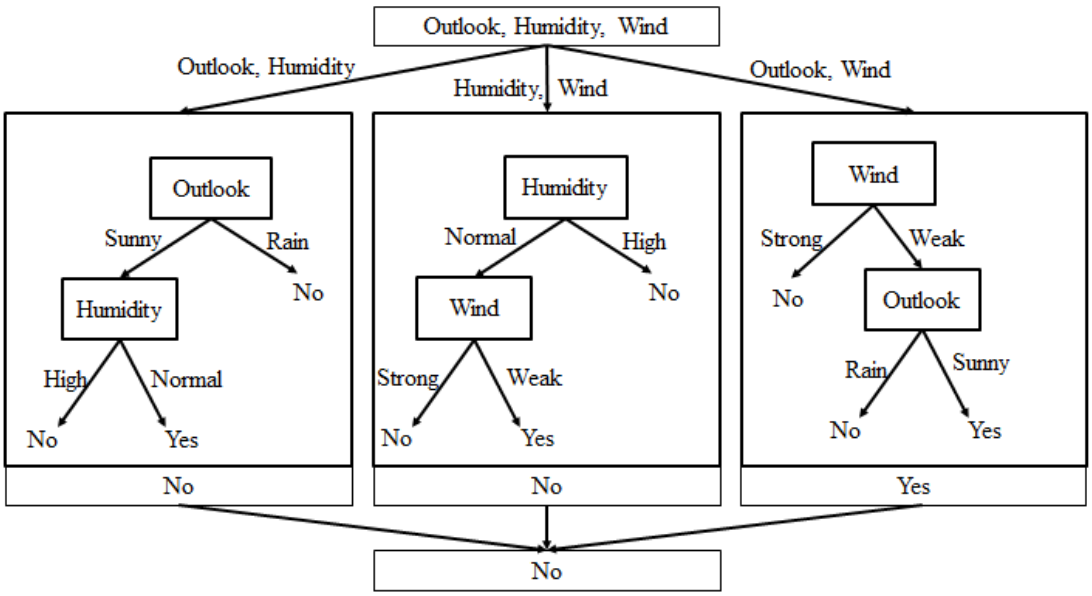


Figure 2.5: Illustration of Random Forest

The random forest has an option to choose random tie-breaking capability in feature space, which means that when two or more randomly selected features for a specific tree look equally important or 'tie', then the tree will select only one of the features. Otherwise, the tree will take all features in the sample into consideration. There are a few advantages of using a random forest (Ali, et al., 2012). Firstly, it can handle the issue of overfitting through bagging and hence getting better predictive accuracy. In addition, it does not require pruning since the overfitting issue is overcome. Besides, it is susceptible to outliers. Unfortunately, it is not easily interpretable to users (Strobl, et al., 2007). Other than that, a random forest can generate noisy trees, which result in wrong decision (Fawagreh, Gaber and Elyan, 2014).

2.2.5 Support Vector Machine

Support vector machine is a technique for regression and classification tasks. For classification, it performs binary classification in nature (Caruana, 2006). The idea is that given the training data, each with labelled class in *n*-dimensional feature space, the learning model tries to find a separating hyperplane in (*n*-1)-dimension with

maximum margin to differentiate two groups of labelled data into two zones. Then, the test data can be classified easily according to the separating hyperplane formulated. For example, consider Figure 2.6, the training data have two features (n = 2) and are classified to two labels, i.e. Class 1 (marked as blue points) and Class 2 (marked as red points). In linear classification, the training model finds the weight vector $\boldsymbol{w}$ and the bias term b as the inputs to the function of the linear hyperplane $f(x) \in \mathbb{R}^1$. Support vectors are the samples lies on the maximum margin at $f(x) = 1$ or $f(x) = -1$. As such $f(x)$ acts as a decision boundary where for the data lies above $f(x) > 0$, the data is classified as Class 1; while for the data lies below $f(x) < 0$, the data is then classified as Class 2. Thus, the test data can be classified easily according to the separating hyperplane created.

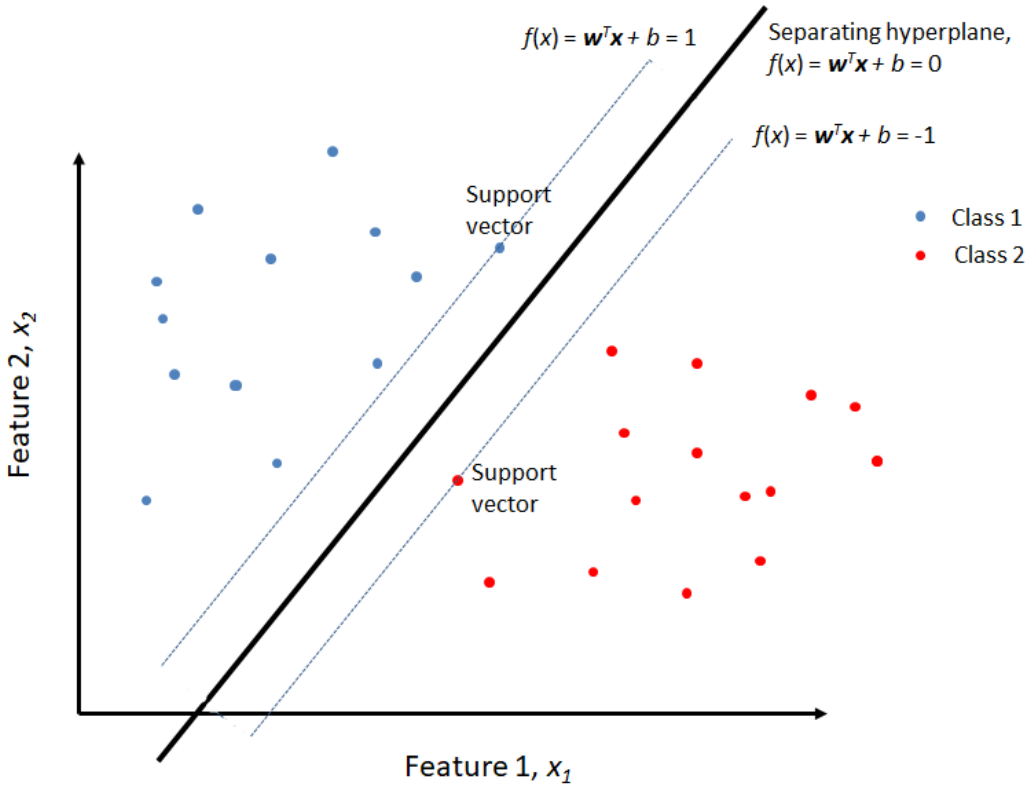


Figure 2.6: Illustration of Support Vector Machine

Support vector machine fundamentally performs binary classification. It is possible to have a nonlinear classifier using kernel tricks, which spans the feature space in a higher dimension (Caruana, 2006). The popular kernel tricks include polynomial kernel and Gaussian radial basis function kernel.

2.3 Ensemble Learning

An ensemble model is a prediction model which comprises of a set of multiple individual prediction models, also known as the base learner (Kelleher, Namee and D'arcy, 2015). Figure 2.7 shows the overall structure of an ensemble model.

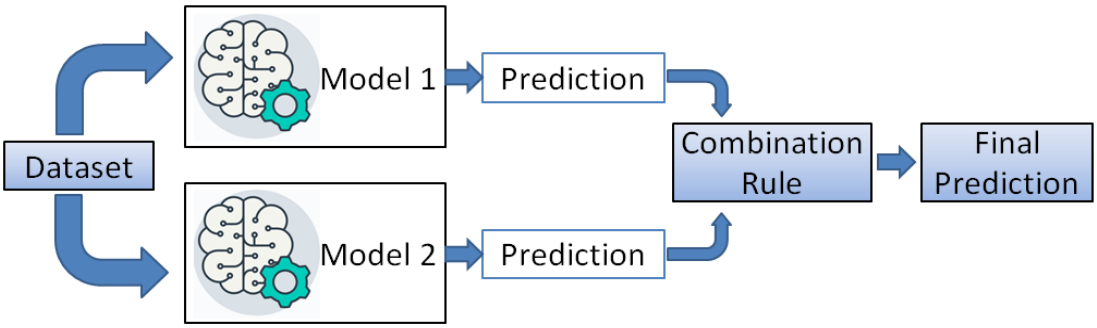


Figure 2.7: Illustration of Ensemble Modelling

According to Marsland (2011), the results generated by the ensemble model will be better than any one of the base learner, provided that the individual learning models are combined well. The theory behind ensemble learning is that individual learners see things differently than others. When it comes to decision, each learner will have individual decision based on its trained model. Then, the decisions from each learner will be combined. Eventually, the task will output a final decision based on certain combination mechanism. It takes a more holistic approach in decision making. Therefore, the generalisation ability of an ensemble model is much stronger than the base learner, resulting in better performance (Zhou, 2012).

There are a few options for the combination of base learners. For instance, majority voting is the most popular combination rule in classification (Zhou, 2012). An interesting fact to note is that in binary classification, the ensemble model can only be wrong if more than half of the base learners are wrong. The next option is by averaging the probabilities of each base learner (Zou, et al., 2015). The class with the highest average probability will be selected as the final decision. Besides, the class with maximum probability can be taken as the final prediction (Malli, Aygun and Ekenel, 2016). To illustrate, Table 2.1 shows the examples of different combination rules and the respective decisions. Assuming this ensemble model contains three base learners (namely, decision tree, Naïve Bayes and *k*-nearest neighbour) and the

task is to classify three outputs (namely, *A*, *B* and *C*). Table 2.1 shows the probabilities or confidence levels in predicting the correct class.

Table 2.1: Probabilities of Prediction for Different Classifiers (Example)

Prediction of Class	Decision Tree	Naïve-Bayes	K-nearest neighbour
A	0.6	0.1	0.7
B	0.3	0.1	0.1
C	0.1	0.8	0.2

When majority voting is chosen, decision tree will predict Class *A*; Naïve-Bayes will predict class *C*; and Neural Network will predict class *A*, based on the probability of each base learner. Hence, the final prediction is class *A*.

When average of probability is chosen, class *A* will have averaged probability of $P(A) = (0.6 + 0.1 + 0.7)/3 = 0.47$; class *B* will have averaged probability of $P(B) = (0.3 + 0.1 + 0.1)/3 = 0.17$; and class *C* will have averaged probability of $P(C) = (0.1 + 0.8 + 0.2)/3 = 0.37$. By comparison, probability of predicting class *A* is the highest. Hence, the final prediction is *A*.

When the maximum probability is chosen, the highest probability found in Table 2.1 is $P(C) = 0.8$, where it is predicted from Naïve Bayes classifier. Hence, the final prediction is class *C*.

On the other hand, the results by Catal, et al. (2015) showed that the combination rule depends on the task itself. There is no straightforward hypothesis to claim that which combination rule is superior to the others. When the correct combination rule is selected, together with the suitable complementary base learners, the prediction accuracy will be improved.

2.4 Feature Selection

Another way to improve the prediction performance is by feature selection. A typical dataset may contain many features. However, not all features are equivalently significant or relevant. In addition, the high number of features may contribute to the curse of dimensionality. To get rid of the curse of dimensionality, feature selection can be used to select the subset of the relevant feature by eliminating redundant or

irrelevant features which proved to contain not much predictive information (Kaur, Sachdeva and Kumar, 2016). By removing noisy features, feature select is also capable of maximising the classification or predictive accuracy.

In general, there are two methods in feature selection, namely filter and wrapper methods (Kohavi and John, 1997). Filter method is irrespective of the machine learning model and fully dependent on the general properties of the training data (Yu and Liu, 2003). It does not need to undergo any learning algorithm. As such, it performs statistical tests on the training data and hence outputs the rank of features' scores. For instance, Wang, Khoshgoftaar and Gao (2010) listed a few techniques under the umbrella of filter methods such as information gain, gain ratio, chi-square test, and *Relief-F*. Nevertheless, recent studies found that the filter method fails to consider the dependencies between the features (Hira and Gillies, 2015). Zeng, et al. (2015) illustrated that a feature may be irrelevant to the class if it is presented individually. However, when it combines with other features, the combination may have a high correlation to the class.

In contrast, wrapper method is able to consider the relationship between the features. Wrapper method also depends on the learning algorithm, which is known as the induction algorithm in this context. Figure 2.8 shows the concept of using the wrapper method (Kohavi and John, 1997). It uses the performance of the induction algorithm to deduce the useful features in the training process (Yu and Liu, 2003). The idea is that it takes different subset of features, learns the induction algorithm, outputs the results and reiterates the process. The search continues until there is no improvement from the previous iterations. It requires more iterations to complete the feature selection process, hence it is claimed to be highly computationally expensive (Kaur, Sachdeva and Kumar, 2016). Despite the disadvantage, it generally performs better and more robust than the filter method, so it has higher accuracy in general (Hira and Gillies, 2015; Hu, et al., 2015; Zeng, et al., 2015).

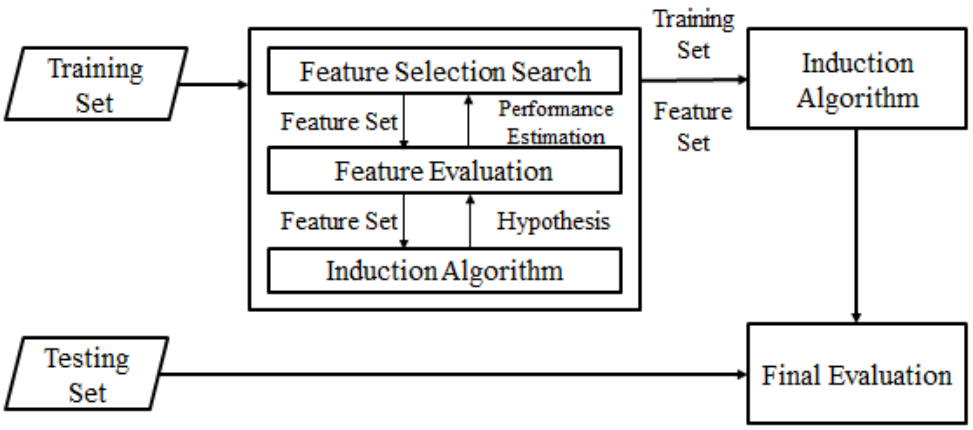


Figure 2.8: Wrapper Method in Feature Selection (Kohavi and John, 1997)

Apart from filter or wrapper-based feature selection, there are also different search algorithms: greedy search and best-first search. Greedy search, also known as hill-climbing algorithm, is the simplest search algorithm. The search starts at one certain node and evaluation will be carried out. Then, the child or possible path will be added to the node and the same evaluation will be performed. If the performance improves at the child node, the search will continue in the vicinity at the child node. The search will terminate when the surrounding child nodes are not improving (Kohavi and John, 1997).

Best-first search algorithm is more robust and complex. The algorithm of best-first search is presented in Figure 2.9 (Kohavi and John, 1997). In Figure 2.9, the number of non-improving iterations is expressed as k, the current working state is expressed as v, the child of v is expressed as w, and the required minimum improvement in accuracy during each iteration is expressed as ε. Best-first search finds the globally best solution in the search space, whereas greedy search can only find the local optima (Skiena, 1998).

Input: Initial state, k
Output: BEST state

OPEN = BEST = Initial state
CLOSED = {}
while BEST changed in the last k expansions **do**
    v = arg $\max_{w \in \text{OPEN}} f(w)$
    remove v from OPEN, add v to CLOSED
    if $f(v) - \varepsilon > f(\text{BEST})$, **then** BEST = v
    expand v, apply all operators to v, giving w (v's children)
    foreach w not in CLOSED or OPEN **do**
        evaluate w, add to OPEN
return BEST

Figure 2.9: Best-First Search Algorithm (Kohavi and John, 1997)

2.5 Datasets Review

This final year project examines three datasets. They are UNSW-NB15, ISCX-IDS2012 and NSL-KDD datasets.

2.5.1 UNSW-NB15 Dataset

Moustafa and Slay (2015) provided UNSW-NB15 dataset. They are publicly available online (refer Appendix A for direct download link). The UNSW-NB15 dataset is one of the newest datasets available for the study in cybersecurity system. The dataset was created using IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security.

The training data consists of 175 341 instances, whereas the test data consists of 82 332 instances. Both of them comprise of 45 attributes, including the two labelled classes, namely “attack_cat” and “label”. The full attributes are shown in Table 2.2.

Table 2.2: Attributes of the UNSW-NB15 Dataset

Number of attributes: 45 including 2 classes				
id	sttl	swin	response_body_len	ct_src_ltm
dur	dttl	stcpb	ct_srv_src	ct_srv_dst
proto	sload	dtcpb	ct_state_ttl	is_sm_ips_ports
service	dload	dwin	ct_dst_ltm	attack_cat
state	sloss	tcprtt	ct_src_dport_ltm	label
spkts	dloss	synack	ct_dst_sport_ltm	
dpkts	sinpkt	ackdat	ct_dst_src_ltm	
sbytes	dinpkt	smean	is_ftp_login	
dbytes	sjit	dmean	ct_ftp_cmd	
rate	djit	trans_depth	ct_flw_http_mthd	

The class named “attack_cat” is meant for multiclass classification. It enumerates the categories of the attack types. There are nine attack types: backdoor, analysis, fuzzer, shellcode, reconnaissance, exploits, denial-of-service, worms and generic. In total, there are ten categorical values for “attack_cat” including the normal category. The other class named “label” is meant for binary classification. It only distinguishes normal (0) and attack (1) instances. The focus of this final year project is to perform multiclass classification, therefore the class named “attack_cat” is treated as the final class. Figures 2.10 and 2.11 show the number of instances for each class in the training and test data respectively.

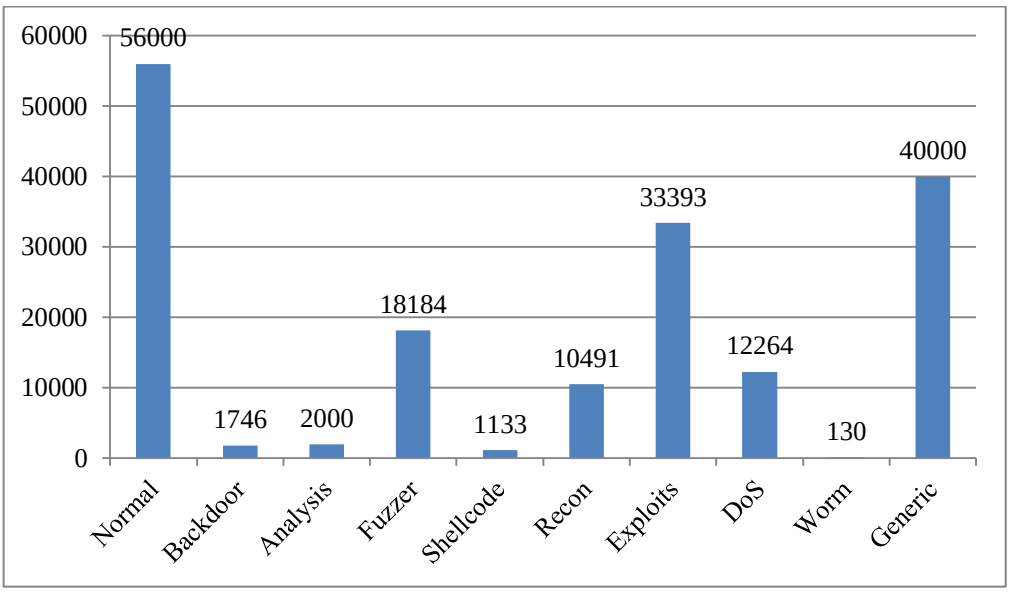


Figure 2.10: Distribution of the Class in UNSW-NB15 Training Data

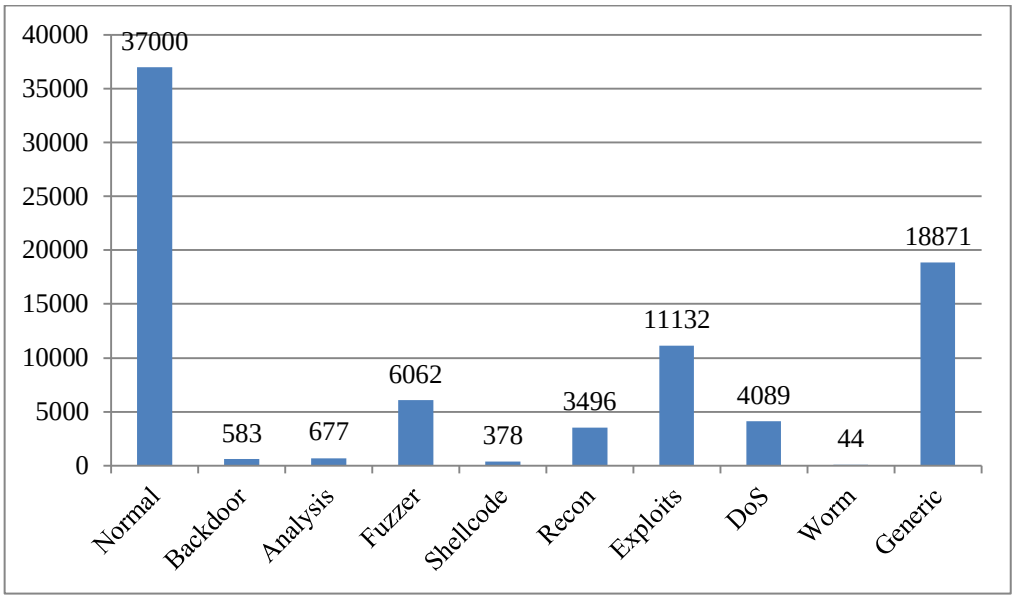


Figure 2.11: Distribution of the Class in UNSW-NB15 Test Data

2.5.2 ISCX-IDS2012 Dataset

Besides UNSW-NB15 dataset, many recent studies also use the ISCX-IDS 2012 dataset (Atli, et al., 2018; Folino, Pisani and Sabatino, 2016; Mirza and Cosan, 2018). This dataset is created by Shiravi, et al. (2012) at the Information Security Centre of Excellence in the University of New Brunswick. The network activity was simulated based on a few principles: realistic traffic and network, labelled dataset, total interaction capture, complete capture and various attack scenarios. The data was collected over seven days. It is different from the UNSW-NB15 dataset because it

contains full packet payloads. This dataset is also publicly available in both processed XML format and raw PCAP format (refer Appendix A for direct download link).

In these seven days, a total of 2 450 324 network packets were collected. There are 20 attributes in this dataset. The full attributes are shown in Table 2.3.

Table 2.3: Attributes of the ISCX-IDS 2012 Dataset

Number of attributes: 20 including 1 class	
appName	sourceTCPFlagsDescription
totalSourceBytes	destinationTCPFlagsDescription
totalDestinationBytes	source
totalDestinationPackets	protocolName
totalSourcePackets	sourcePort
sourcePayloadAsBase64	destination
sourcePayloadAsUTF	destinationPort
destinationPayloadAsBase64	startDateTime
destinationPayloadAsUTF	stopDateTime
direction	Tag

In general, there are five categorical values for the “Tag” class, where one is of normal class and the remaining four classes are of four different attack types: distributed denial-of-service, brute force, infiltration and HTTP denial-of-service. However, from the total 2 450 324 network packets, 2 381 532 of them (97.2 %) are normal traffic. Aside from the highly biased nature, it is also found that there are duplicate network packets in this dataset. Besides, the training and test sets are not explicitly provided. Hence, pre-processing step is required for this dataset and it will be discussed in Chapter 3. After performing the pre-processing step, training set of 55 094 instances and test set of 23 613 instances are obtained. Figures 2.12 and 2.13 show the number of instances for each class in training and test data respectively.

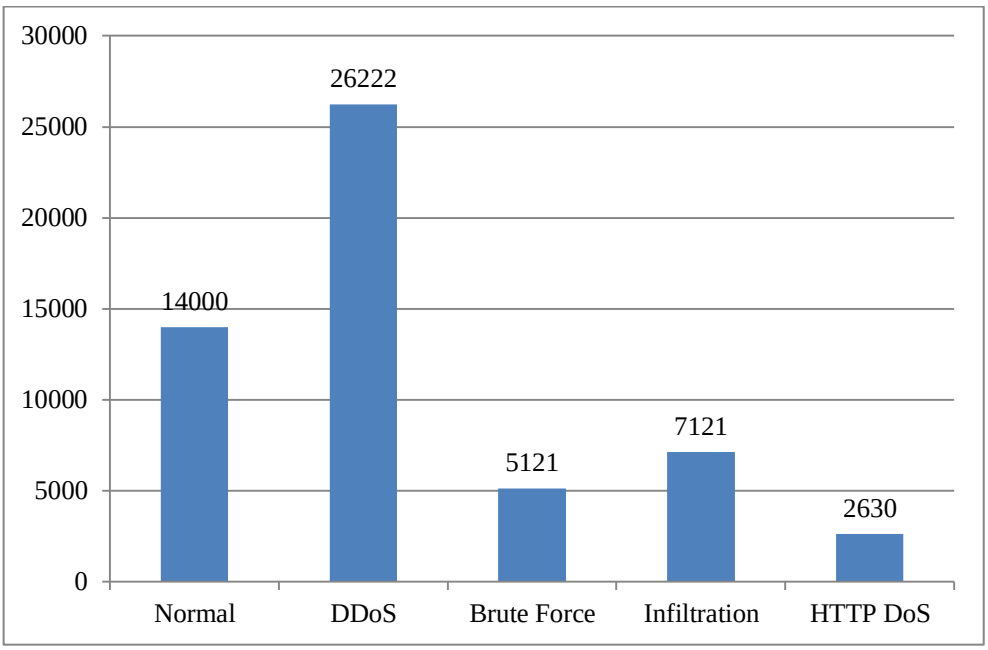


Figure 2.12: Distribution of the Class in ISCX-IDS 2012 Training Data

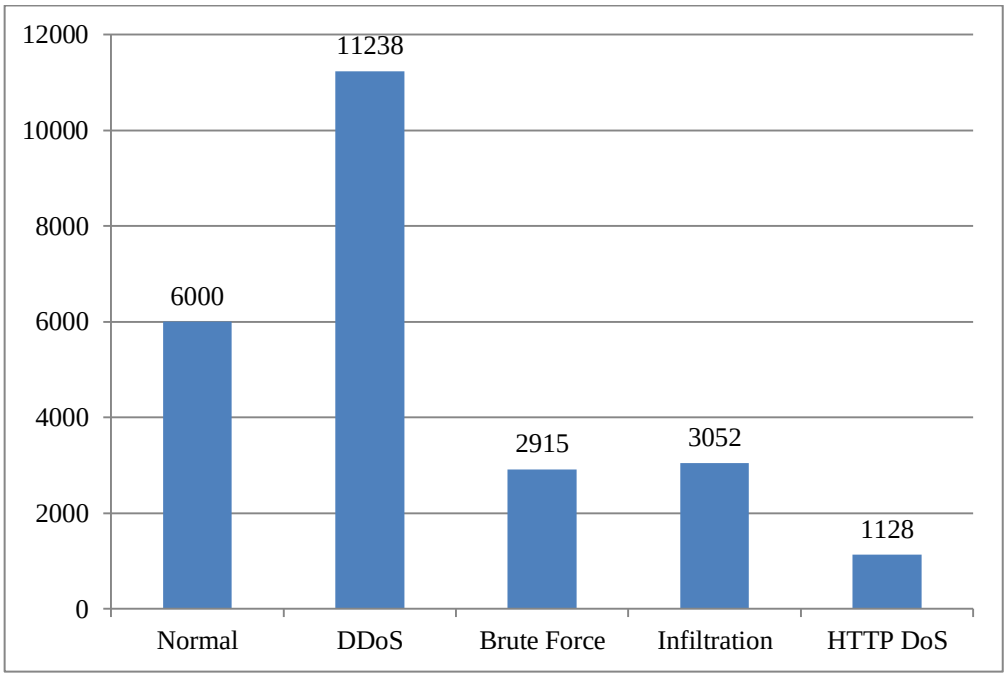


Figure 2.13: Distribution of the Class in ISCX-IDS 2012 Test Data

2.5.3 NSL-KDD Dataset

The NSL-KDD dataset is a newer derivative from the KDD-CUP'99 dataset. KDD-CUP'99 dataset is notorious for its highly duplicated records in both training and testing sets (Tavallaee, et al., 2009). As a solution, the NSL-KDD dataset has removed redundant records. Although it may not be a perfect representative of

existing modern network traffic, it can still be utilised as an effective benchmark dataset in the research of intrusion detection (Revathi and Malathi, 2013).

Tavallaee, et al. (2009) provided both the training and test data in ARFF, CSV and TXT formats (refer Appendix A for direct download link). Due to the WEKA platform, the dataset in ARFF format was downloaded for this final year project. The training set contains 125 973 instances while the test set contains 22 544 instances. This dataset consists of 42 attributes, as shown in Table 2.4.

Table 2.4: Attributes of the NSL-KDD Dataset

Number of attributes: 42 including 1 class		
duration	num_root	srv_diff_host_rate
protocol_type	num_file_creations	dst_host_count
service	num_shells	dst_host_srv_count
flag	num_access_files	dst_host_same_srv_rate
src_bytes	num_outbound_cmds	dst_host_diff_srv_rate
dst_bytes	is_host_login	dst_host_same_src_port_rate
land	is_guest_login	dst_host_srv_diff_host_rate
wrong_fragment	count	dst_host_serror_rate
urgent	srv_count	dst_host_srv_serror_rate
hot	serror_rate	dst_host_rerror_rate
num_failed_logins	srv_serror_rate	dst_host_srv_rerror_rate
logged_in	rerror_rate	class
num_compromised	srv_rerror_rate	
root_shell	same_srv_rate	
su_attempted	diff_srv_rate	

The dataset in ARFF format is merely meant for binary classification. The class contains “normal” and “anomaly” categories. The distributions of classes in training and test set are shown in Figure 2.14 and Figure 2.15 respectively.

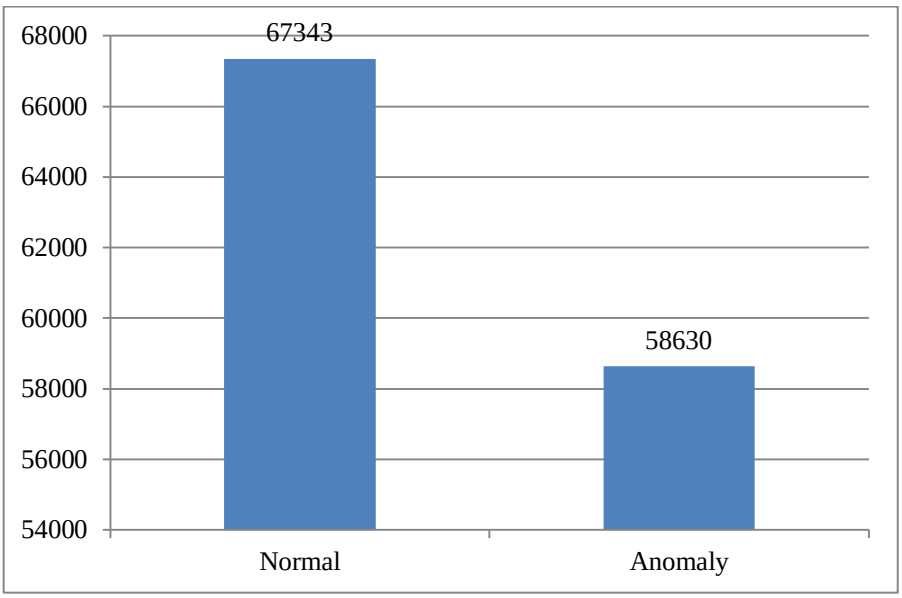


Figure 2.14: Distribution of the Class in NSL-KDD Training Data

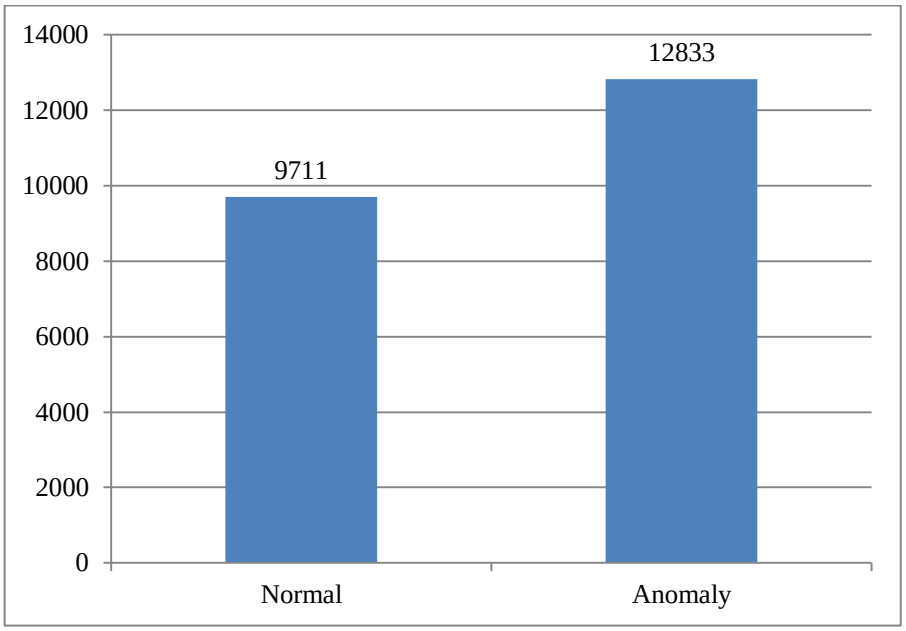


Figure 2.15: Distribution of the Class in NSL-KDD Test Data

2.6 Evaluation Metrics

To quantify the performance measures of different techniques, a suitable and universal evaluation metric should be objectively applied to all related research in the intrusion detection system. All the outputs of the evaluation can be classified into four categories, as shown in Figure 2.16. In Figure 2.16, Class A is assumed to be the targeted class.

		Predicted outcome	
		Positive (Class A)	**Negative (Not Class A)**
Actual outcome	**Positive (Class A)**	**True positive (TP):** Correctly classified instances of Class A	**False negative (FN):** Misclassified instances of Class A as other classes
	Negative (not Class A)	**False positive (FP):** Misclassified instances not of Class A as Class A	**True negative (TN):** Correctly rejected instances not of Class A

Figure 2.16: Confusion Matrix of Classification

The simplest and the most intuitive performance measures of the intrusion detection system is the accuracy (Park, Song and Cheong, 2018). It is defined as the ratio of the total number of correct predictions to the total number of available data. The accuracy is defined in Equation (2.2).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

However, the evaluation using accuracy is biased when the data is imbalanced. Park, Song and Cheong (2018) gave an example as follows. A dataset contains 1000 samples where 990 samples are positive and 10 samples are negative. One can effortlessly predict all samples as positive and ignore all the negative samples, but the accuracy remains exceptional and up to 99 %. So, another evaluation metrics called the precision and recall are suggested. They are defined mathematically in the Equations (2.3) and (2.4).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.4}$$

A high accuracy does not necessarily mean high precision and recall. The precision and recall are interrelated. Park, Song and Cheong (2018) claimed that IDS

evaluation by using only either one of them is not sufficient. They can be combined as a new parameter of evaluation metric, the F_β –score, given by Equation (2.5).

$$F_\beta\text{-score} = \frac{(1+\beta^2)(\text{precision} \times \text{recall})}{\beta^2 \times \text{precision} + \text{recall}} \tag{2.5}$$

The β value represents how significant is the recall as compared to the precision. For example, if $\beta = 1$, then the evaluation considers precision and recall as equally important; if $\beta = 2$, then recall is twice as important as the precision. Park, Song and Cheong (2018) mentioned that the common approach to relating recall and precision is the F_1-score, which represents the harmonic mean between the two, defined mathematically in Equation (2.6).

$$F_1\text{-score} = \frac{2(\text{precision} \times \text{recall})}{\text{precision} + \text{recall}} \tag{2.6}$$

The higher F_1-score indicates that the performance of the IDS model is better. F_1-score takes into account the ability to predict minor classes. In this project, F_1-score is also known as F-measure.

2.7 Related Works

Suleiman and Issac (2018) evaluated the performances of their six machine learning based IDS models in WEKA platform against NSL-KDD, UNSW-NB15 and Phishing datasets. These six IDS models are random forest, J48 decision tree, *k*-nearest neighbour, artificial neural network, support vector machine and Naïve Bayes. They claimed that *k*-nearest neighbour and random forest are the best performing algorithms across the three datasets. For example, random forest and *k*-nearest neighbour classifiers have exceptionally high accuracies of 99.76 % and 99.44 % respective for NSL-KDD dataset. However, when the experiment was reproduced using UNSW-NB15 dataset, it is found that they recorded the results for “Generic” class only. In fact, UNSW-NB15 contains ten classes, including the normal category. A fair result should be captured by taking the weighted average of the classification for all ten classes. It remains unclear whether the results using other datasets were also biasedly justified. Secondly, UNSW-NB15 dataset provides two types of classes,

i.e., “label” and “attack_cat” classes for binary and multiclass classifications respectively (Moustafa and Slay, 2015). While performing multiclass classification, Suleiman and Isaac (2018) failed to remove the “label” class, which is meant for binary classification scenario. The model would consider “label” class as a feature when training the model. In fact, the real-time network would not reveal whether the incoming packets are of attack or normal categories in advance, therefore the “label” is not a feature for the model to learn from and thus it should be removed from the feature set.

A recent study by Al-kasassbeh, et al. (2018) used KDD-CUP’99 dataset to test the IDS models. The IDS models were also developed and tested in WEKA environment. Six algorithms were employed as the IDS classifiers, including J48 decision tree, random forest, random tree, multilayer perceptron, Naïve Bayes and Bayesian network. They aimed to classify the four attack types (probing, denial-of-service, user-to-root and remote-to-user). They found that random forest has the best accuracy of up to 93.775 %. Unfortunately, McHugh (2000) pointed out that the KDD-CUP’99 dataset was merely generated from the simulation of military networking in the old days, which cannot represent the modern low-footprint attack. It is no longer fit for the development of modern IDS. Besides, Tavallaee, et al. (2009) stated that KDD-CUP’99 dataset contains a huge portion of duplicated records, where 78 % in training data and 75 % in test data were redundant. The duplicate records would result in biased training and testing towards the majority instances and ignoring the rest. Hence, a newer and balanced dataset should be used for the development and the evaluation of modern IDS model.

Furthermore, Haider, et al. (2017) claimed that there is a lack of suitable dataset in the research of IDS. It is because the datasets containing network packets have some degree of confidentiality, so the data is not available publicly due to privacy issues. Besides, the available datasets they had examined, including DARPA and KDD-CUP’99 were claimed to be outdated and unrealistic for modern network traffic. Therefore, they generated a new dataset called the next-generation IDS (NGIDS) dataset. However, in the study by Haider, Hu and Moustafa (2017), the performance of the classifiers using NGIDS dataset was rather poor. For example, using support vector machine classifier, the true positive rate for NGIDS dataset was only 5 %, as compared to 70% and 40 % for ADFA and KDD datasets respectively.

Despite the low performance, using self-generated dataset would raise the question of whether the dataset is biased for use. It cannot be guaranteed that the self-generated dataset is free from artificiality. Consequently, the validity of the experiment is doubtful when the self-generated dataset was used. Conversely, the literature shows that there are newer available public datasets such as UNSW-NB15 and ISCX-IDS2012, which contain modern traffic and diverse attacks.

Yassin, et al. (2013) presented a combined machine learning technique from *k*-means clustering and Naïve Bayes for anomaly detection in IDS. *K*-means clustering is used to cluster the attack traffic and the normal traffic, and then the Naïve Bayes classifier further verifies the clustered data and classifies them into normal or attack categories. This is a binary classification problem. They used ISCX-IDS2012 dataset for the evaluation of their proposed technique. Since the ISCX-IDS2012 dataset is large, they decided to select only the incoming packets at one particular host. There are 77 526 training data, where 75 372 (97.2 %) of them are normal and 2154 (2.8 %) of them belong to attack class. Using their *k*-means clustering plus Naïve Bayes model, they obtained an accuracy of 99 % and a true positive rate of 98.8 %. Despite the exceptional result, the training data is highly imbalanced, which means the IDS model learned towards biased normal class. Longadge and Dongre (2013) said that a biased model towards the major class will have a poor detection rate on the minor class. Also, the classifier may ignore the minor class and assumes everything as the major class, yet it can produce good accuracy. It is important to balance the training data through data pre-processing, so that a wide range of attack categories including the minor categories can be recognised.

Jabbar and Aluvalu (2017) proposed an ensemble classifier of IDS. The base learners of the IDS are the random forest algorithm and the average one dependency estimator. Random forest builds multiple decision trees through randomly selected bootstrap samples. Average one dependency estimator is used to generalise the dependency among the features. They used Kyoto dataset with 24 features for training and testing the IDS model. The result shows that the ensemble classifier outperforms the individual random forest and average one dependency estimator models. More specifically, the accuracy obtained using the ensemble IDS model was 90.51 %, as compared to 89.34 % for random forest and 89.68 % for average one

dependency estimator. However, in the pre-processing step, they only included 15 features in model training and excluded features related to security analysis. They did not provide further justification on why those features regarding to security analysis were negligible. The features in security analysis could be useful in prediction. In fact, when deciding the features prior to model training, feature selection technique can be applied. It systematically reduces the insignificant features through either filter or wrapper methods.

Gharaee and Hosseinvand (2016) developed an IDS model using support vector machine with genetic algorithm. Inspired by the evolutionary concept of natural genetics, the genetic algorithm is a type of wrapper-based feature selection technique. The genetic algorithm comprises of mutation and crossover operations. These operations are applied to the feature set. The genetic algorithm will then generate a better feature subset as the 'offspring'. The generated feature subset will then be evaluated using fitness function. Eventually, the latest generated feature subset will be chosen for the model training using support vector machine. They conducted the experiment on UNSW-NB15 and KDD-CUP'99 datasets. The performance of this IDS model is excellent, with an accuracy of up to 99.45 % for "Shellcode" category in UNSW-NB15 dataset. Unfortunately, they excluded the minor classes of the UNSW-NB15 dataset in the prediction, namely "Analysis", "Backdoor" and "Worms" categories. The proposed IDS model was only meant to classify six categories. In addition, after evaluating the classification performance on the attack-to-attack basis, they did not calculate the weighted average of the overall performance. This raises the argument of whether the IDS model is capable to recognise minor classes. In fact, Gharaee and Hosseinvand (2016) should unbiasedly evaluate the IDS ability to predict minor classes. They should also calculate the overall performance of the IDS model with respect to all attack categories so that their IDS model can be compared fairly with other proposed models. Furthermore, they are not supposed to use KDD-CUP'99 in their study due to outdated network data and a large portion of duplicated records (McHugh, 2000; Tavallaee, et al., 2009).

The review of other related works highlights some concerns. First, the dataset used in evaluating the performance of the IDS model should be pre-processed for the purpose of balancing. Besides, an obsolete dataset of network information should be

avoided for modern IDS development. The self-generated dataset should not be due to lacking of common benchmarking. In addition, a highly biased dataset should also be avoided. The evaluation of the IDS model should be carried out fairly, which examines the major and minor classes equally. Lastly, any insignificant features can be removed using feature selection technique. It is known that in the machine learning context, a few techniques such as feature selection and ensemble learning can be used to improve the prediction accuracy. However, the effectiveness of these techniques remains unclear in the application of IDS.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Overview of Project Work Plan

Figure 3.1 shows the overview of the project work plan. Several machine learning algorithms in WEKA tool are utilised such as Naïve Bayes (NB), *k*-nearest neighbour (KNN), J48 decision tree (J48), random forest without tie-breaking capability (RF), random forest with tie-breaking capability (RF-BT) and support vector machine (SVM). The details of the work plan are discussed in subsequent sub-chapters.

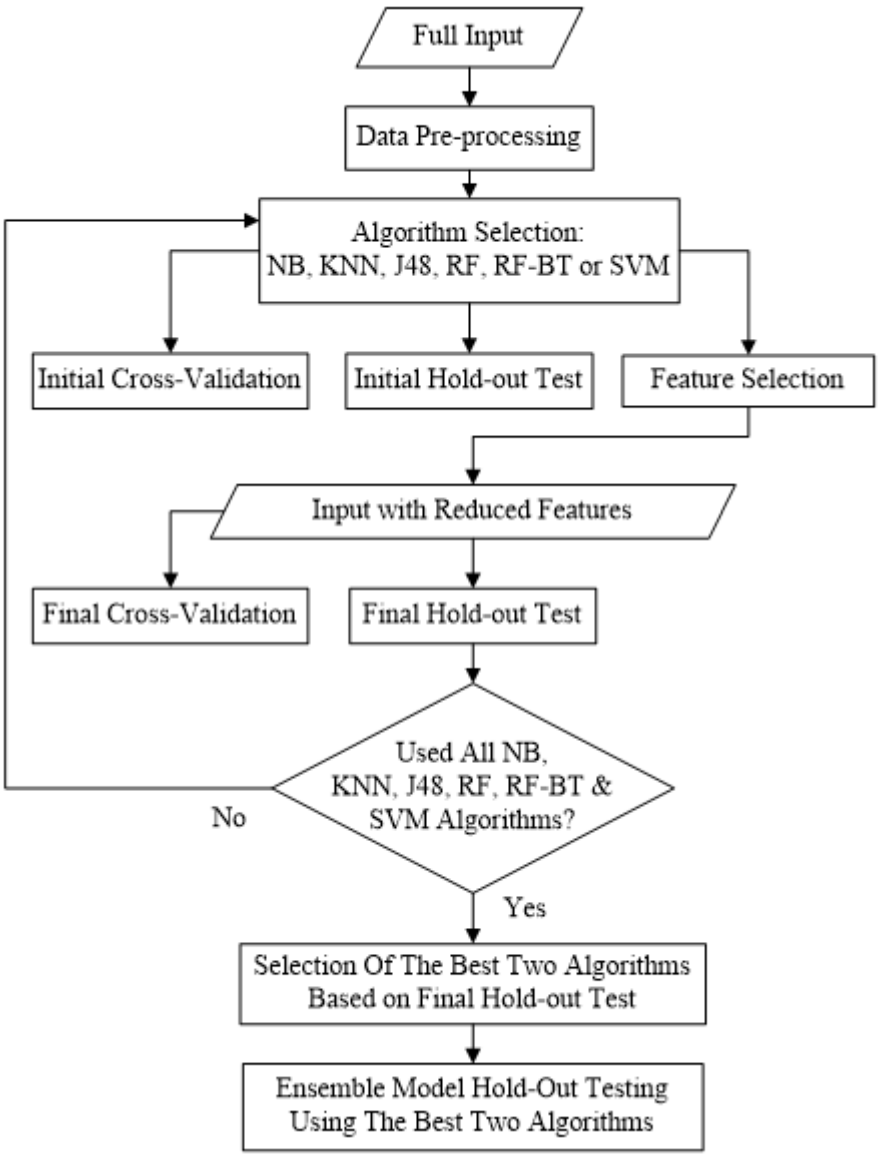


Figure 3.1: Project Overview and Proposed Architecture

3.2 Dataset Preparation

Three different datasets are used, namely UNSW-NB15, ISCX-IDS2012 and NSL-KDD datasets. As a matter of fact, sufficient data pre-processing steps are required to produce a fair and logical result. Specifically, UNSW-NB15 and ISCX-IDS2012 datasets require data pre-processing. Meanwhile, the downloaded NSL-KDD dataset has been adequately processed and thus does not require further data pre-processing.

3.2.1 UNSW-NB15 Pre-Processing

As discussed in sub-chapter 2.5.1, there are two types of classes in UNSW-NB15 dataset. The "label" class is meant for binary classification whereas the "attack_cat" class is meant for multiclass classification. The focus of this project is to perform multiclass classification. Hence, "label" class is removed, leaving "attack_cat" as the final class for prediction. Otherwise, the machine learning model might consider "label" as one of the features and give a biased result.

Furthermore, there is a feature named "id", which only serves as an index for each instance in the training and test sets. It does not provide practical information in predicting the malicious attack, thus it should be removed. Moreover, if "id" remained in the dataset, then the learning process would be biased. For example, if the "id" ranging from 1 to 20 000 are all normal network packets, then the model would conclude that the indices within this range are all normal during model testing, which is essentially incorrect. Therefore, the feature "id" is also removed.

Although it can be seen in sub-chapter 2.5.1 that the dataset is imbalance across different classes, further data pre-processing is not carried out to balance the number of instances for each class. It is because the difference in quantity between the major and minor classes is too large. For example, in the training set, 56 000 instances are normal and only 130 instances belong to "worm" attack. If the data is insisted to be balanced by removing most of the normal instances, then the training data might end up to be very small, which only consists of about 1000 instances. A small dataset is not sufficient for comprehensive generalisation. In addition, overfitting and bias are more likely to occur in a small dataset, which contribute to even poorer prediction. Considering the trade-off, balancing is not performed for UNSW-NB15 dataset. Eventually, the training and test sets are converted from CSV to ARFF format to suit the WEKA tool.

3.2.2 ISCX-IDS2012 Pre-Processing

ISCX-IDS2012 dataset requires more pre-processing steps. The dataset is made up of 12 files in XML format. The 12 files were converted to CSV format using a python script (refer Appendix B). After combining all 12 files into a single CSV file, it is found that there were duplicate instances. Therefore, the duplicate records are eliminated using another python script (refer Appendix C).

After the removal of duplicates, the dataset consisted of 2 071 657 unique instances, where 2 002 747 (96.7 %) of them were normal, 3776 (0.18 %) were HTTP flooding, 37 460 (1.81 %) were DDoS attacks, 7316 (0.35 %) were brute force attacks and 20 358 (0.98 %) were infiltration. As mentioned, balancing is performed on the data so that the trained model would not be biased towards the majority normal class. In specific, the undersampling technique is applied to proportionate the classes. From 2 002 747 normal instances, 20 000 of them were randomly selected using python script (refer Appendix C).

In addition, since the training and testing sets were not explicitly provided, random train-test split with the ratio of 70:30 is applied using python script (refer Appendix C). Eventually, training set of 55 094 instances and test set of 23 613 instances are obtained and saved in ARFF format.

3.3 Algorithm and Settings

After data pre-processing stage, the next step is to test the effectiveness of different conventional machine learning classifiers. The paths of the machine learning models in WEKA are shown in Table 3.1.

Table 3.1: Machine Learning Classifiers in WEKA and Respective Paths

Classifiers	Paths
Naïve Bayes	weka.classifiers.bayes.NaiveBayes
K-nearest neighbour	weka.classifiers.lazy.IBk
J48 decision tree	weka.classifiers.trees.J48
Random forest	weka.classifiers.trees.RandomForest
Support vector machine	weka.classifiers.functions.SMO

Each algorithm has dedicated parameter settings, in which modifying those settings would produce different results. In order to study the optimisation effects of feature selection and ensemble learning, but no other factors, the parameter settings are fixed throughout the project. Figures 3.2 to 3.7 show the experimental settings for the six machine learning algorithms in WEKA.

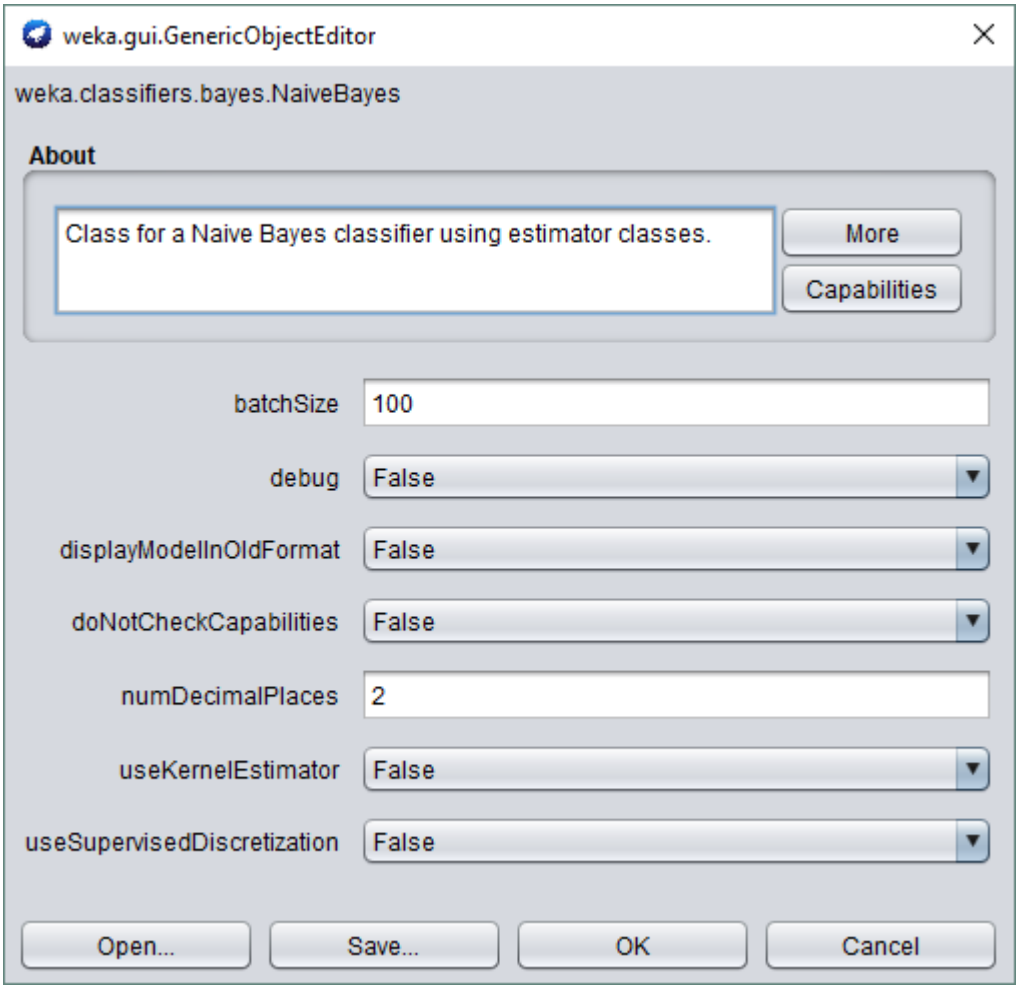


Figure 3.2: Parameter Settings for Naïve Bayes Classifier

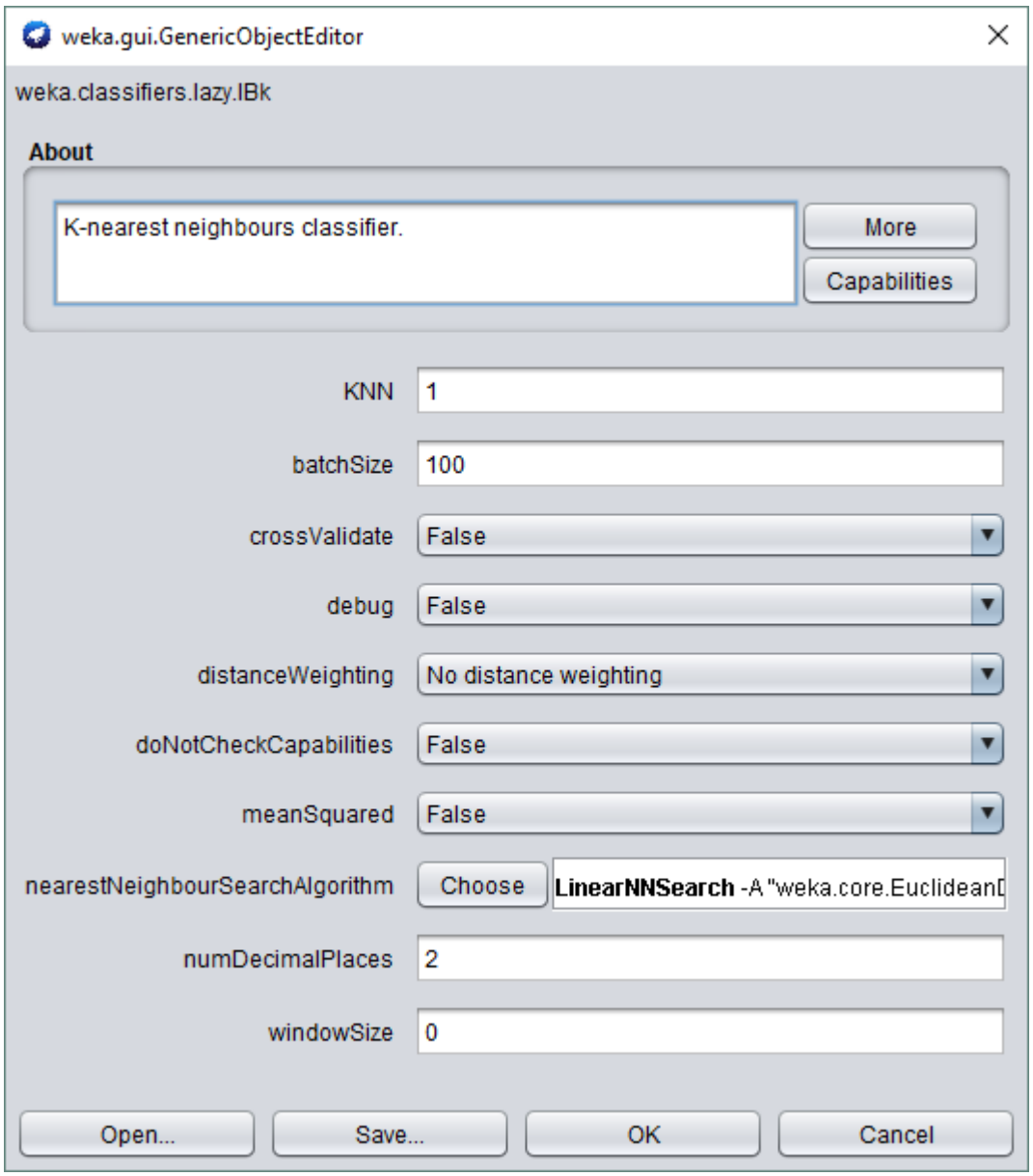


Figure 3.3: Parameter Settings for *K*-Nearest Neighbour Classifier

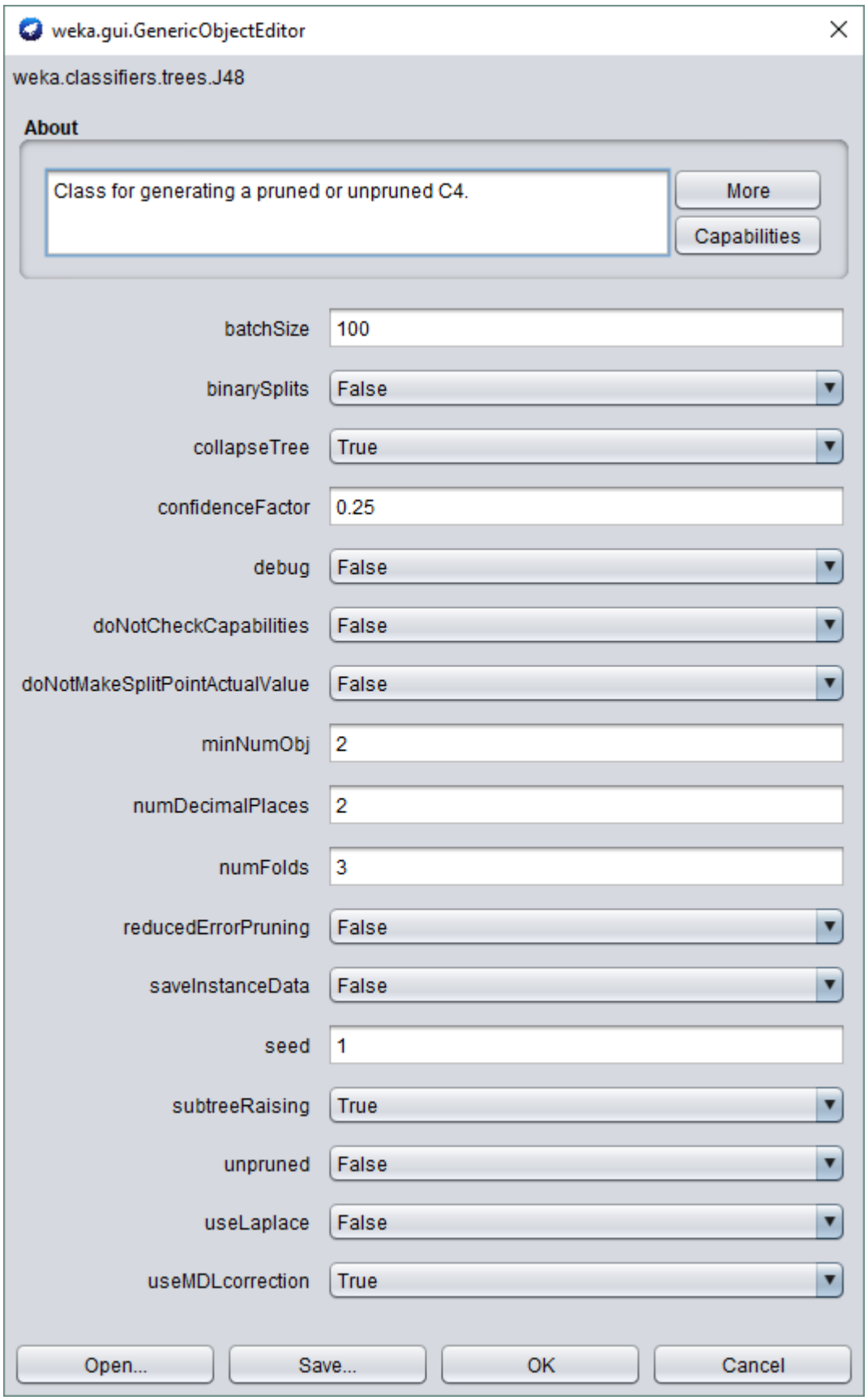


Figure 3.4: Parameter Settings for J48 Decision Tree Classifier

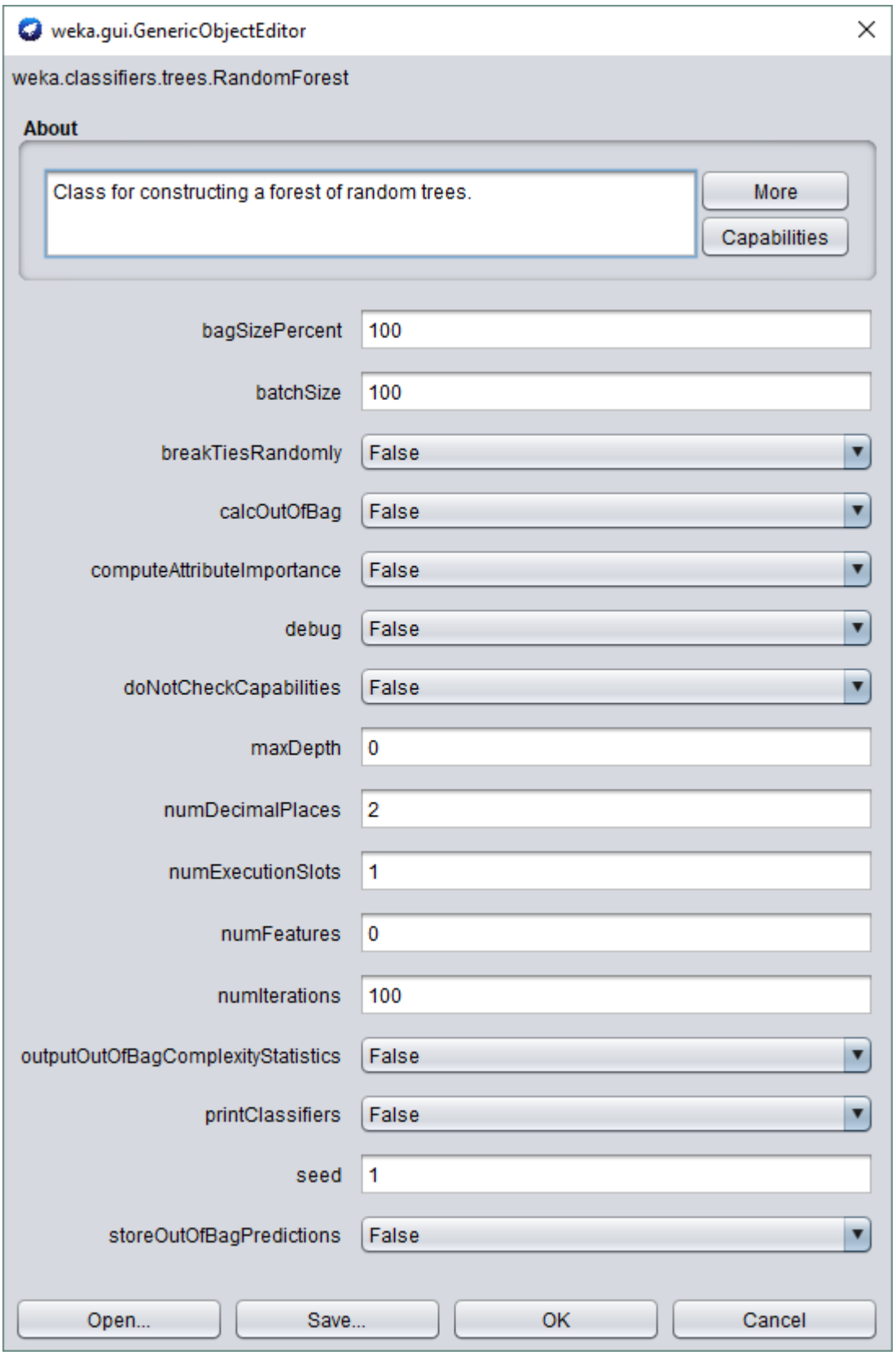


Figure 3.5: Parameter Settings for Random Forest Classifier without Tie-Breaking Capability

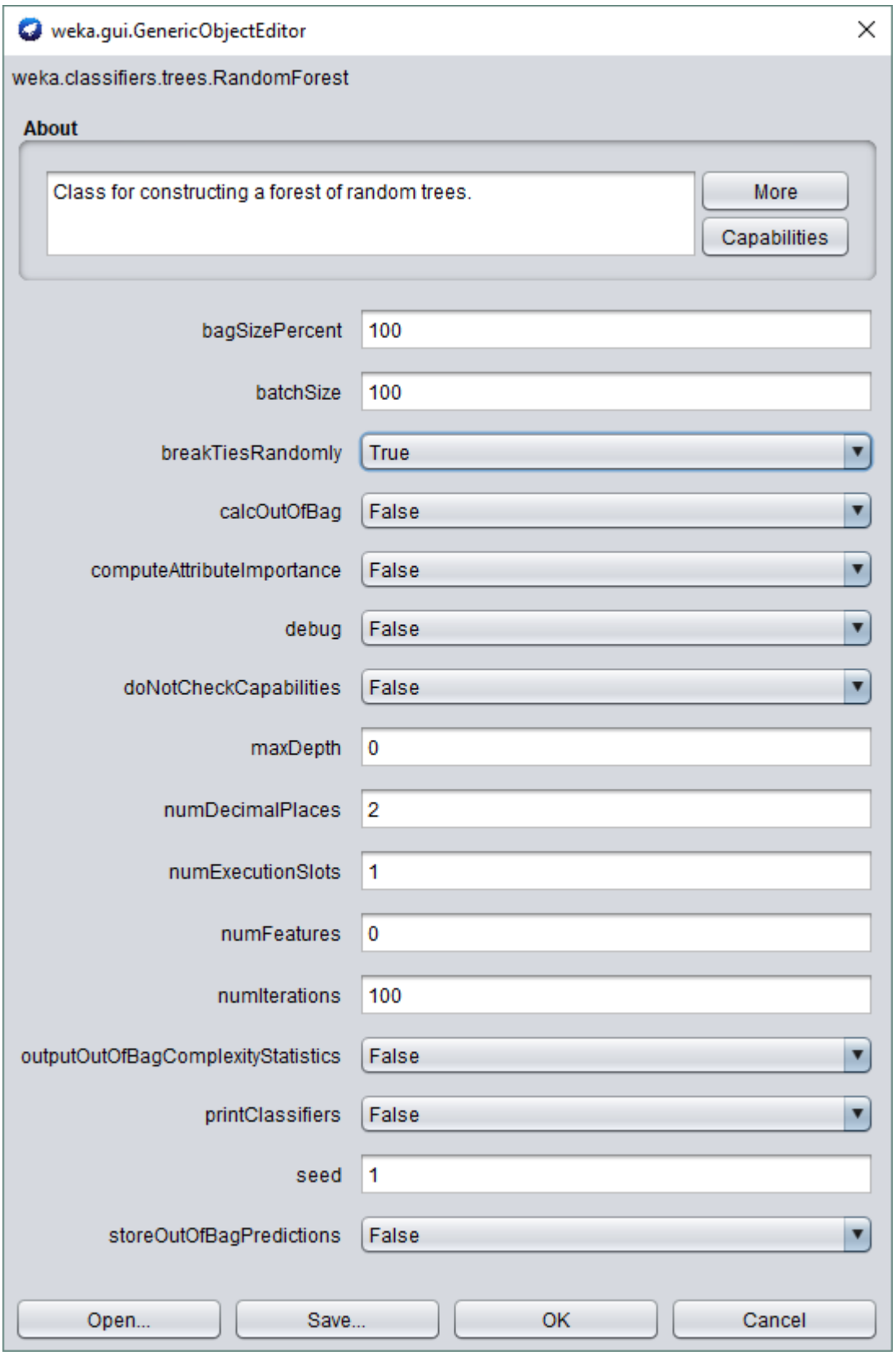


Figure 3.6: Parameter Settings for Random Forest Classifier with Tie-Breaking Capability

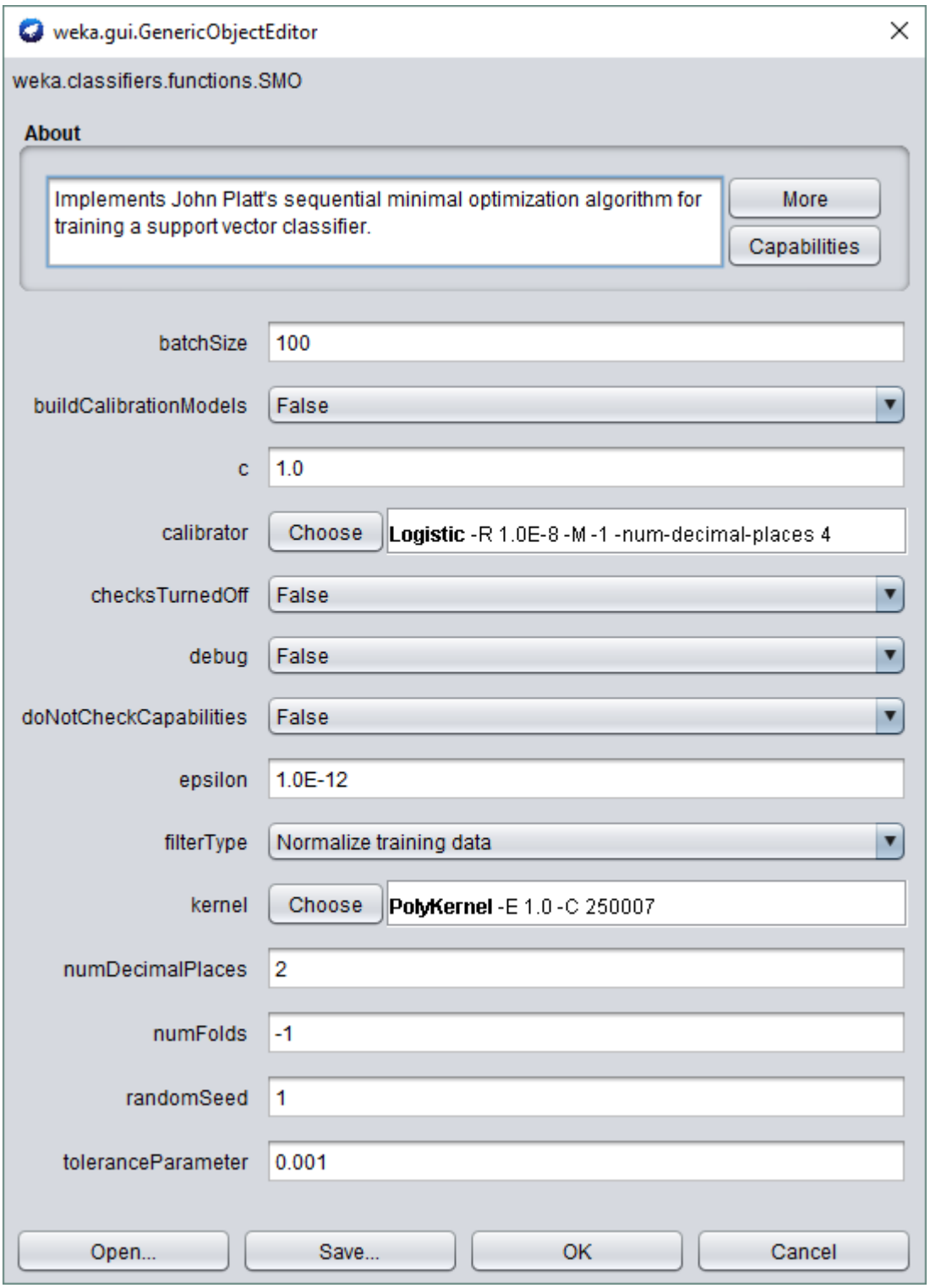


Figure 3.7: Parameter Settings for Support Vector Machine Classifier

3.4 Initial 10-fold Cross Validation and Hold-out Test

Initial cross validation and hold-out test consider all features for making prediction. These steps set a benchmark before any optimisation techniques such as feature selection and ensemble learning. This benchmark score is useful for comparison

purposes after optimisation. The purpose of cross validation is to verify how well each algorithm in handling the training data. Furthermore, the purpose of hold-out test is to perform real evaluation based on the unseen data.

3.5 Integration of Feature Selection

Next, the wrapper-based feature selection technique is performed in WEKA using best-first search algorithm. The feature selection module is placed in "weka.attributeSelection.WrapperSubsetEval". It aims to get rid of the redundant features, to prevent the curse of dimensionality in high feature space and to enhance the classification accuracy. Figure 3.8 shows the settings of using wrapper-based feature selection for Naïve Bayes classifier in WEKA.

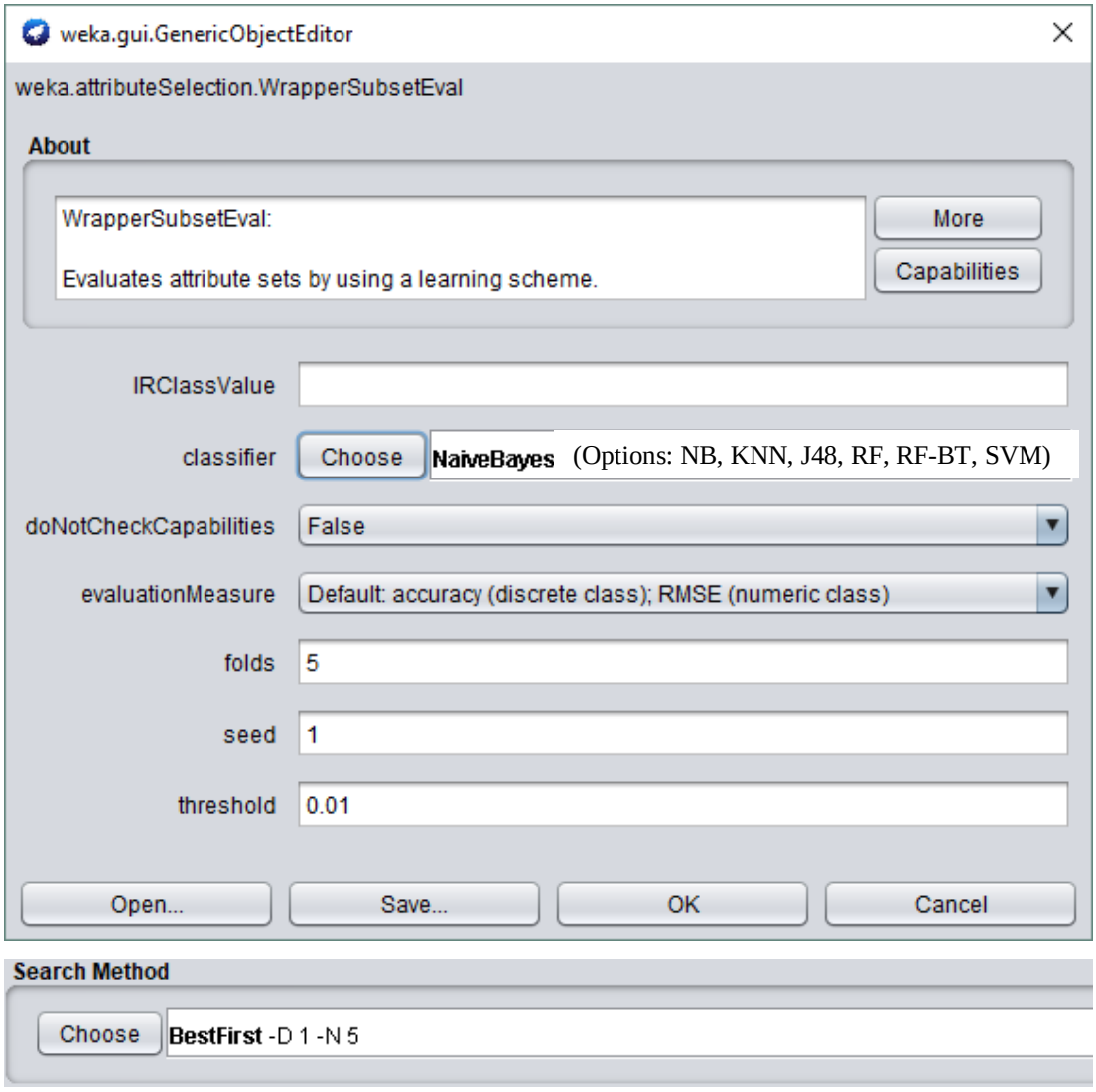


Figure 3.8: Wrapper-based Feature Selection with Best-First Search in WEKA

After feature selection, subsets of useful features are obtained exclusively for the six algorithms.

3.6 Final 10-fold Cross Validation and Hold-out Test

Final cross validation and hold-out test only consider the extracted important features for making prediction. They are expected to obtain better results as compared to those before feature selection. For the final cross validation and hold-out test, "FilteredClassifier" in "weka.classifiers.meta.FilteredClassifier" is used instead of the original machine learning classifier. Specifically, the selected filter function is called "Remove" in "weka.filters.unsupervised.attribute.Remove". Figure 3.9 shows the parameter settings of using "FilteredClassifier". Figure 3.10 shows the example settings of using "Remove" filter function.

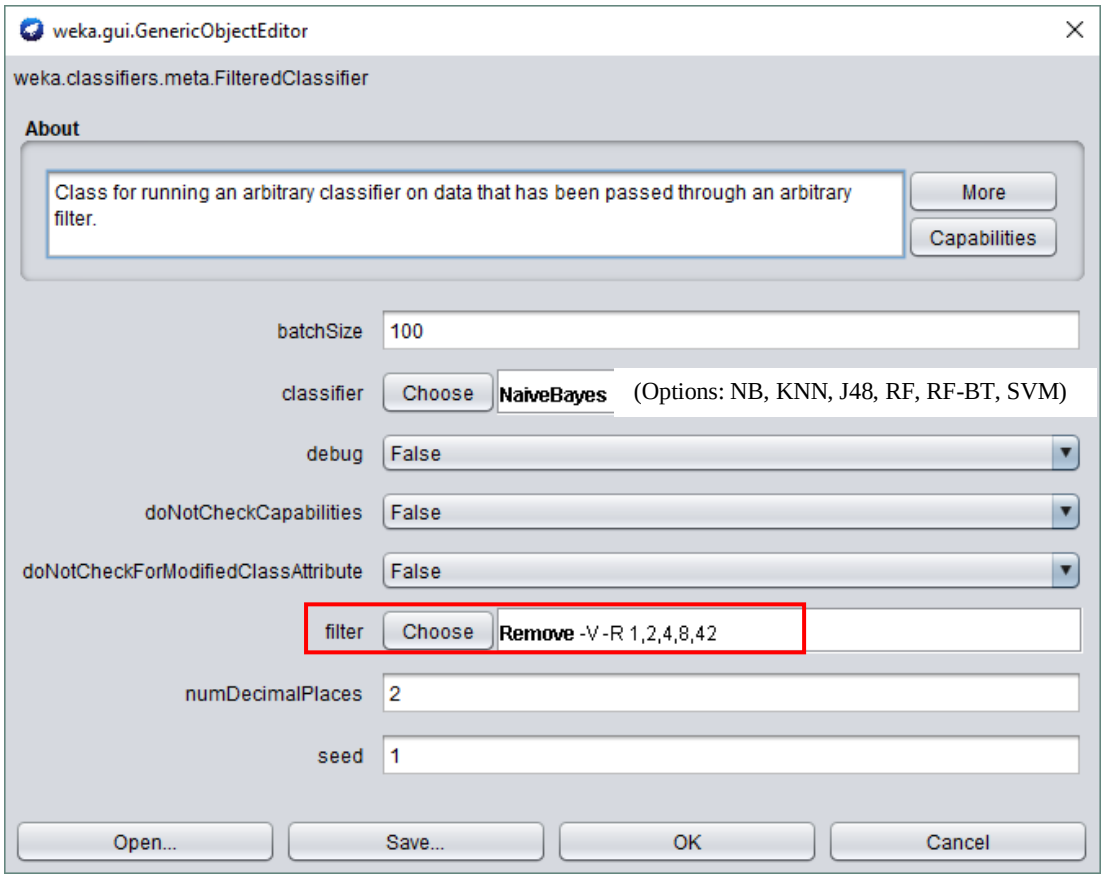


Figure 3.9: Parameter Settings for "FilteredClassifier" in WEKA

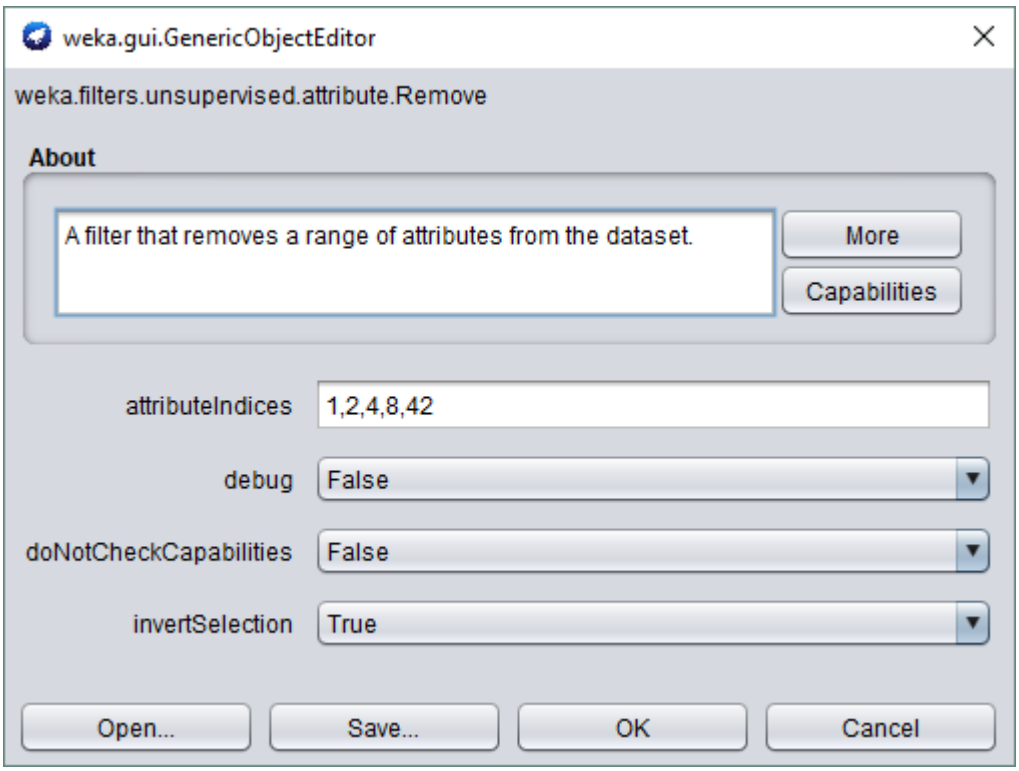


Figure 3.10: Example Settings of Using "Remove" Function

Figure 3.10 illustrates an example of using "Remove" function. Given that the dataset consists of 42 attributes, where the attributes indexing from "1" to "41" are the full features and the attribute with the index of "42" is the labelled class. If the features with indices "1", "2", "4" and "8" form the subset of important features, then the numbers "1, 2, 4, 8, 42" (including the class) are required to be entered in the "attributeIndices" section. In fact, these index numbers will be retained instead of removed, hence the "invertSelection" should be set "True". Eventually, the "Remove" filter will retain attributes at indices "1", "2", "4", "8" and "42" while removing all other attributes.

3.7 Integration of Ensemble Learning

When the final cross validation and hold-out test of the six algorithms are completed, the next step is to perform ensemble learning. The ensemble model consists of two best performing machine learning algorithms. For comparison purposes, the best and the third best algorithms would also be taken for ensemble learning. The aim of having ensemble model is to further enhance classification accuracy by exploiting the advantages of selected individual machine learning algorithms. Figure 3.11

shows the configuration of the ensemble model in WEKA. It is named “Vote” from “weka.classifiers.meta.Vote”. Figure 3.12 shows the selection of the individual classifiers under the parameter settings of “Vote”.

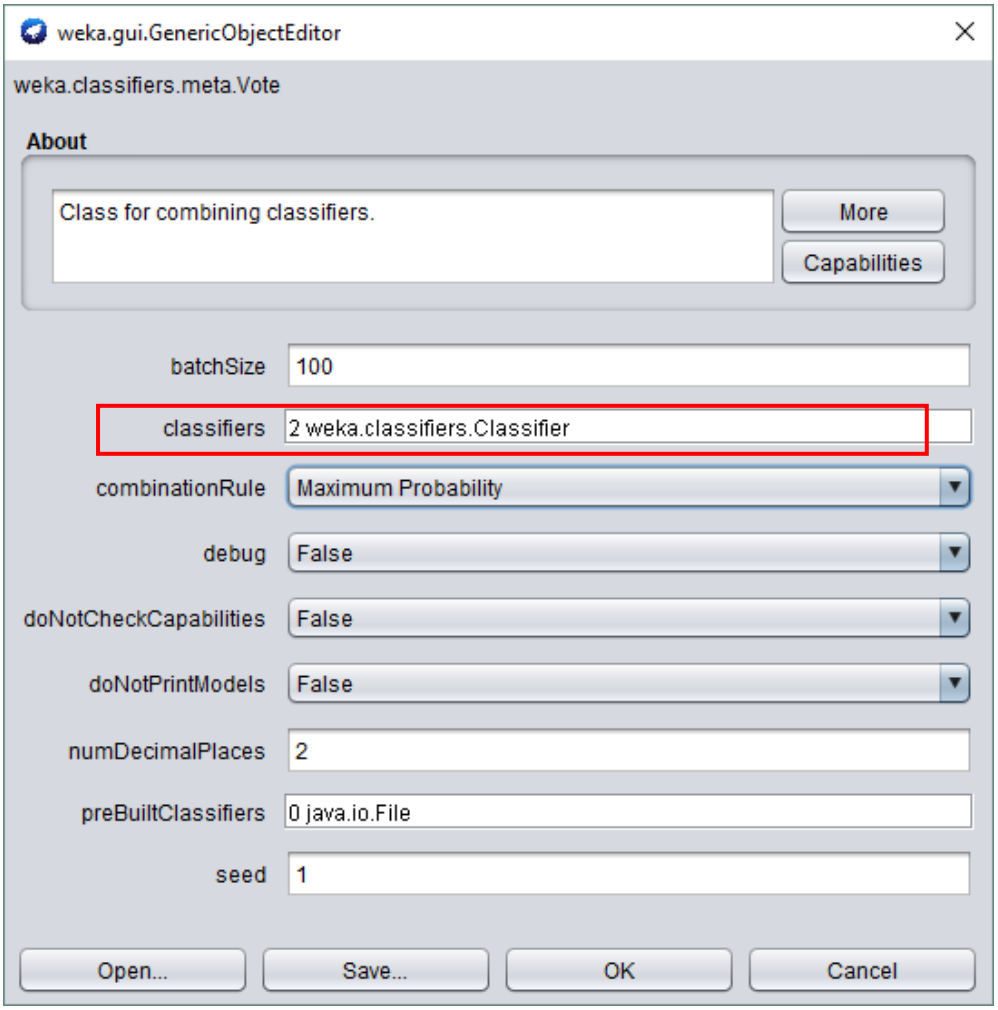


Figure 3.11: Parameter Settings of an Ensemble Model

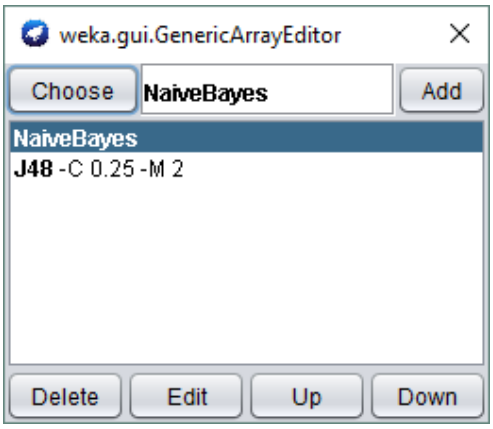


Figure 3.12: Selection of Individual Base Learners for Ensemble Learning

3.8 Project Planning and Resource Allocation

Time, computer's memory resources and processing speed are the main limitations for this project. A meticulous plan is developed by taking into account these limitations. Figure 3.13 shows the Gantt chart for this project. All the tasks were successfully carried out and completed on time.

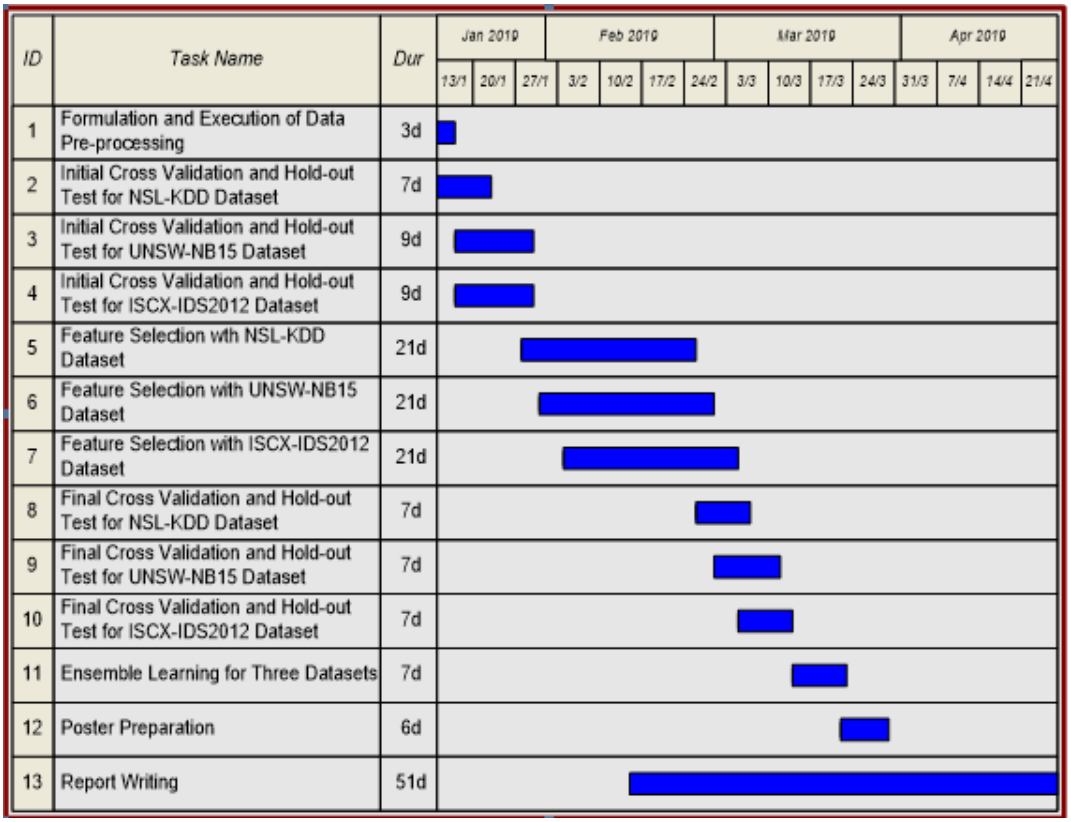


Figure 3.13: Tasks List and Project Planning

3.9 Anticipated Problems and Solutions

It is expected that wrapper-based feature selection technique would take a long duration. For UNSW-NB15 dataset, support vector machine algorithm required more than two weeks to evaluate the important feature subset from the training data. After 15 days, the feature selection method is still unable to conclude the important feature subset. Therefore, the train set is reduced to 50 % with the same class distribution. The reduced train set is adopted to evaluate the important feature subset for support vector machine algorithm. Eventually, feature selection of using the reduced train set is completed within one week due to smaller data. Most importantly, when the preliminary confirmation test is carried out, the classification accuracy would still be improved based on the results from feature selection using the reduced training set.

In the preliminary model training for SVM algorithm using ISCX-IDS2012 dataset, it is found that the physical memory of the computer is insufficient. The payload information in ISCX-IDS2012 has taken up most of the available RAM space. Specifically, four features named “sourcePayloadAsBase64”, “sourcePayloadAsUTF”, “destinationPayloadAsBase64” and “destinationPayloadAsUTF” contain payload information. Thus, the payload information is removed. For ISCX-IDS2012 dataset, the number of attributes is reduced to 16 as shown in Table 3.2. In other words, ISCX-IDS2012 dataset with 15 features became the primary input to the initial cross validation and hold-out test. After the removal of payload information, the performances of all IDS models are still remarkably well.

Table 3.2: Updated Attributes of the ISCX-IDS2012 Dataset

Number of attributes: 16 including 1 class	
appName	source
totalSourceBytes	protocolName
totalDestinationBytes	sourcePort
totalDestinationPackets	destination
totalSourcePackets	destinationPort
direction	startDateTime
sourceTCPFlagsDescription	stopDateTime
destinationTCPFlagsDescription	Tag

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Description of Evaluation Scheme

UNSW-NB15, ISCX-IDS2012 and NSL-KDD datasets contribute to three different network usage scenarios. The measures such as accuracy, precision, recall and F-measure are employed to appraise the classification performance of the machine learning based IDS models. The results from the three scenarios are discussed separately in the following sub-chapters. The confusion matrices for all cross validations and hold-out tests for UNSW-NB15, ISCX-IDS2012 and NSL-KDD datasets are provided in Appendices D, E and F respectively.

4.2 Results and Discussions for UNSW-NB15 Dataset

Table 4.1 shows the important feature subsets for the six algorithms respectively after implementing feature selection technique.

Upon having the subsets of important feature, the final 10-fold cross validation and hold-out test after feature selection could be performed. The performance results after feature selection are then compared to the results of initial 10-fold cross validation and hold-out test before feature selection. Table 4.2 shows the performance results for the cross validations before and after feature selection, whereas Table 4.3 presents the performance results for the hold-out tests before and after feature selection.

In Table 4.2, it can be observed that the F-measure for SVM is not-a-number (NaN), because the precision is also NaN. According to Equation (2.3), precision is the ratio of the number of true positives to the number of conditional positives. To examine the reason for having a NaN precision, Table 4.4 shows the confusion matrix for SVM model for initial cross validation.

Table 4.1: Important Feature Subsets for Different Machine Learning Algorithms using UNSW-NB15 Dataset

Algorithms	Selected Features
NB	proto, service, ct_state_ttl, ct_dst_sport_ltm, ct_flw_http_mthd, ct_src_ltm
KNN	service, spkts, sbytes, dbytes, sttl, dttl, sloss, dloss, response_body_len, ct_state_ttl, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_srv_dst
J48	proto, service, state, spkts, sbytes, dbytes, sttl, sloss, dloss, smean, dmean, trans_depth, response_body_len, ct_srv_src, ct_state_ttl, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, is_ftp_login, ct_flw_http_mthd, ct_src_ltm, ct_srv_dst
RF, RF-BT	service, spkts, sbytes, dbytes, sttl, dload, sloss, dloss, synack, smean, response_body_len, ct_srv_src, ct_state_ttl, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_flw_http_mthd, ct_srv_dst
SVM	dur, proto, service, spkts, dpkts, sbytes, dbytes, rate, sttl, dttl, sload, dload, sloss, sinpkt, djit, swin, stcpb, dwin, tcprtt, synack, ackdat, smean, dmean, trans_depth, ct_srv_src, ct_state_ttl, ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_flw_http_mthd, ct_src_ltm, ct_srv_dst, is_sm_ips_ports

Table 4.2: Performance Results of Various IDS Models for Cross Validation using UNSW-NB15 Dataset

	Metrics	NB	KNN	J48	RF	RF-BT	SVM
Before Feature Selection	**Accuracy**	52.90	76.72	83.28	82.73	82.72	77.13
	Precision	73.19	75.31	83.18	82.47	82.46	NaN
	Recall	52.90	76.72	83.28	82.73	82.72	77.13
	F-measure	58.44	75.54	81.29	81.78	81.76	NaN
After Feature Selection	**Accuracy**	70.19	82.73	83.91	84.10	84.11	77.08
	Precision	NaN	82.13	84.45	84.03	84.05	NaN
	Recall	70.19	82.73	83.91	84.10	84.11	77.08
	F-measure	NaN	81.36	81.82	82.67	82.70	NaN

Table 4.3: Performance Results of Various IDS Models for Hold-out Test using UNSW-NB15 Dataset

	Metrics	NB	KNN	J48	RF	RF-BT	SVM
Before Feature Selection	**Accuracy**	48.08	70.19	75.26	75.58	75.65	69.01
	Precision	77.14	77.22	81.71	83.77	84.09	NaN
	Recall	48.08	70.19	75.26	75.58	75.65	69.01
	F-measure	56.22	72.89	76.82	77.82	77.87	NaN
After Feature Selection	**Accuracy**	54.42	75.01	76.31	77.14	77.12	69.09
	Precision	NaN	80.37	82.28	83.75	83.78	NaN
	Recall	54.42	75.01	76.31	77.14	77.12	69.09
	F-measure	NaN	76.46	77.51	78.36	78.40	NaN

Table 4.4: Confusion Matrix for SVM Model for Initial Cross Validation

		Predicted									
		N	B	A	F	S	R	E	D	W	G
Actual	N	45827	0	39	8871	0	468	790	4	0	1
	B	10	3	0	64	0	98	1571	0	0	0
	A	202	0	131	2	0	0	1665	0	0	0
	F	703	0	0	14167	0	1290	1976	2	0	46
	S	6	0	0	428	0	699	0	0	0	0
	R	57	0	0	1096	0	5908	3395	0	0	35
	E	359	8	0	2405	0	516	30053	29	0	23
	D	114	3	0	444	0	215	11388	37	0	63
	W	2	0	0	11	0	6	111	0	0	0
	G	14	0	0	156	0	39	675	0	0	39116

*N: Normal, B:Backdoor, A:Analysis, F:Fuzzers, S:Shellcode, R:Reconnaissance, E:Exploits, D: Denial-of-Service, W:Worms, G:Generic

According to Table 4.4, the respective IDS model fails to predict the classes “Shellcode” and “Worms”. Therefore, there is no conditional positive to those classes, causing the division-by-zero error when calculating precision. Therefore, it can be concluded that if F-measure or precision is NaN, then the IDS model is unable to predict instance of at least one class. In short, the model which obtains NaN for precision or F-measure should be avoided because the inability to discriminate attack traffic may possibly lead to catastrophic failure in real-life practices.

The classes of “Worms” and “Shellcode” are overlooked by the respective machine learning IDS model because they are minor classes. According to the class distribution of UNSW-NB15 training data in Figure 2.5, “Worms” and “Shellcode”

are the two classes with the least instances. This also proves that a balanced dataset in model training is needed.

From Tables 4.2 and 4.3, it can be observed that implementing feature selection consistently increases the classification accuracy for all models. This shows the effectiveness of feature selection in increasing the classification accuracy of the machine learning IDS model.

According to Table 4.3, the best performing algorithm in term of accuracy is RF, followed by RF-BT and J48. To study the effect of ensemble learning, the best two pair of algorithms (RF and RF-BT) are combined, trained and tested. The same procedure is applied to the best and third best algorithms (RF and J48). Table 4.5 tabulates the results of the individual learning models and ensemble models.

Table 4.5: Performance Results of the Best Three Individual Models and Ensemble Models for UNSW-NB15 Dataset

Metrics	Individual Models			Ensemble Models	
	Best	Second Best	Third Best	Best + Second Best	Best + Third Best
	RF	RF-BT	J48	RF + RF-BT	RF + J48
Accuracy	77.14	77.12	76.31	77.20	76.24
Precision	83.75	83.78	82.28	83.91	82.58
Recall	77.14	77.12	76.31	77.20	76.24
F-measure	78.36	78.40	77.51	78.46	77.52

Table 4.5 shows that the ensemble model of having the best two individual models (RF and RF-BT) improves in terms of accuracy and F-measure. On the other hand, the ensemble model which combines RF and J48 shows a decrease in accuracy and F-measure. Therefore, this observation ascertains that the ensemble model will only yield better results than each of the individual base learners, provided that they are combined correctly. For UNSW-NB15 dataset, the combination of RF and RF-BT is able to further enhance the prediction performance of the IDS model (e.g., up to 77.20 % in accuracy and 78.46 % in F-measure).

4.3 Results and Discussions for ISCX-IDS2012 Dataset

Table 4.6 shows the important feature subsets for the six algorithms respectively after implementing feature selection technique.

Table 4.6: Important Feature Subsets for Different Machine Learning Algorithms using ISCX-IDS2012 Dataset

Algorithms	Selected Features
NB	appName, totalSourcePackets, direction, sourceTCPFlagsDescription, source, sourcePort, destination, startDateTime, stopDateTime
KNN	appName, totalSourceBytes, direction, sourceTCPFlagsDescription, source, destination, startDateTime
J48	appName, totalSourceBytes, totalDestinationBytes, totalSourcePackets, direction, sourceTCPFlagsDescription, destinationTCPFlagsDescription, source, protocolName, sourcePort, destinationPort, stopDateTime
RF, RF-BT	totalSourceBytes,direction, sourceTCPFlagsDescription, source, sourcePort, destinationPort
SVM	appName, direction, source, sourcePort, destination, startDateTime

With the abovementioned important feature subsets, the final cross validation and hold-out test are performed and compared with the initial cross validation and hold-out test. Table 4.7 shows the performance results of cross validations, whereas Table 4.8 shows the performance results of hold-out tests.

Table 4.7: Performance Results of Various IDS Models for Cross Validation using ISCX-IDS2012 Dataset

	Metrics	NB	KNN	J48	RF	RF-BT	SVM
Before Feature Selection	**Accuracy**	98.44	99.70	99.61	97.94	98.91	99.37
	Precision	98.46	99.70	99.61	98.00	98.92	99.38
	Recall	98.44	99.70	99.61	97.94	98.91	99.37
	F-measure	98.44	99.70	99.61	97.93	98.90	99.37
After Feature Selection	**Accuracy**	99.00	99.77	99.68	99.69	99.73	99.64
	Precision	99.02	99.77	99.68	99.69	99.74	99.64
	Recall	99.00	99.77	99.68	99.69	99.73	99.64
	F-measure	99.00	99.77	99.68	99.69	99.73	99.64

Table 4.8: Performance Results of Various IDS Models for Hold-out Test using ISCX-IDS2012 Dataset

	Metrics	NB	KNN	J48	RF	RF-BT	SVM
Before Feature Selection	**Accuracy**	98.28	99.64	99.57	97.87	98.67	98.60
	Precision	98.32	99.64	99.58	97.93	98.69	98.62
	Recall	98.28	99.64	99.57	97.87	98.67	98.60
	F-measure	98.30	99.64	99.57	97.86	98.67	98.60
After Feature Selection	**Accuracy**	99.08	99.73	99.62	99.61	99.60	99.54
	Precision	99.09	99.73	99.62	99.61	99.61	99.54
	Recall	99.08	99.73	99.62	99.61	99.60	99.54
	F-measure	99.08	99.73	99.62	99.61	99.61	99.54

In general, the performance results using the ISCX-IDS dataset are surprisingly well for the six abovementioned algorithms. The accuracies are more than 97 % even before any optimisation. As expected, the implementation of feature selection consistently increased the models' accuracies and F-measures as shown in Tables 4.7 and 4.8.

According to Table 4.8, the algorithm with the best accuracy is KNN, followed by J48 and RF. Table 4.9 tabulates the results of the individual KNN, J48 and RF models, together with the ensemble models of KNN + J48 and KNN + RF.

Table 4.9: Performance Results of the Best Three Individual Models and Ensemble Models for ISCX-IDS2012 Dataset

Metrics	Individual Models			Ensemble Models	
	Best	Second Best	Third Best	Best + Second Best	Best + Third Best
	KNN	J48	RF	KNN + J48	KNN + RF
Accuracy	99.73	99.62	99.61	99.74	99.71
Precision	99.73	99.62	99.61	99.74	99.71
Recall	99.73	99.62	99.61	99.74	99.71
F-measure	99.73	99.62	99.61	99.74	99.71

The integration of KNN and J48 models yields better performance. The feature-ranking and pruning properties in J48 enabled high prediction accuracy. Although pruning helps in mitigating overfitting issue, there are still unavoidable redundant tree branches which lead to inaccurate prediction, unless the pruning parameters are defined correctly in exact numerical values. For KNN, shorter Euclidean distance would claim higher confidence for correct prediction. However, noisy data would negatively affect the prediction performance. In this case, the complementary effect of KNN and J48 decision tree take place positively. The overfitting issue is overcame when KNN directly classifies the test instance by calculating the shortest Euclidean distance with the highest predictive confidence. The noisy data in KNN are mitigated through the pruning of J48 decision tree. Eventually, when these two classifiers were combined, the ensemble model is able to outperform these two base learners.

On the other hand, the combination of KNN and RF produced weaker prediction accuracy. It could be due to the fact that the RF generates noisy trees which adversely affects the performance of the ensemble model.

4.4 Results and Discussions for NSL-KDD Dataset

Table 4.10 shows the important feature subsets for the six algorithms respectively after implementing feature selection technique.

Table 4.10: Important Feature Subsets for Different Machine Learning Algorithms using NSL-KDD Dataset

Algorithms	Selected Features
NB	service, flag, num_file_creations
KNN	service, flag, src_bytes, dst_bytes, wrong_fragment, urgent, hot, logged_in, num_compromised, num_root, num_file_creations, num_access_files, is_host_login, is_guest_login, count, same_srv_rate
J48	duration, service, src_bytes, dst_bytes, logged_in, num_file_creations, count, srv_count, serror_rate, srv_serror_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate
RF, RF-BT	duration, service, flag, src_bytes, dst_bytes, num_failed_logins, logged_in, root_shell, num_root, num_file_creations, num_outbound_cmds, count, srv_rerror_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_rerror_rate
SVM	duration, service, flag, src_bytes, dst_bytes, land, hot, num_compromised, su_attempted, num_file_creations, num_shells, count, srv_count, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate

After obtaining the important feature subsets, the next step is to perform final cross validation and hold-out test. The results are then compared to those obtained from the initial cross validation and hold-out test. Table 4.11 shows the performance results of cross validations, while Table 4.12 presents the performance results of hold-out tests.

Table 4.11: Performance Results of Various IDS Models for Cross Validation using NSL-KDD Dataset

	Metrics	NB	KNN	J48	RF	RF-BT	SVM
Before Feature Selection	**Accuracy**	90.38	99.66	99.78	99.92	99.92	95.95
	Precision	90.48	99.66	99.78	99.92	99.92	95.99
	Recall	90.38	99.66	99.78	99.92	99.92	95.95
	F-measure	90.36	99.66	99.78	99.92	99.92	95.95
After Feature Selection	**Accuracy**	96.22	99.70	99.85	99.92	99.92	97.18
	Precision	96.22	99.70	99.85	99.92	99.92	97.19
	Recall	96.22	99.70	99.85	99.92	99.92	97.18
	F-measure	96.22	99.70	99.85	99.92	99.92	97.18

Table 4.12: Performance Results of Various IDS Models for Hold-out Test using NSL-KDD Dataset

	Metrics	NB	KNN	J48	RF	RF-BT	SVM
Before Feature Selection	**Accuracy**	76.12	79.36	81.53	80.45	80.19	75.39
	Precision	80.90	84.10	85.79	85.19	85.03	80.20
	Recall	76.12	79.36	81.53	80.45	80.19	75.39
	F-measure	75.94	79.23	81.47	80.34	80.06	75.20
After Feature Selection	**Accuracy**	78.39	80.74	85.30	81.45	81.43	75.92
	Precision	81.16	82.89	87.59	85.48	85.46	80.49
	Recall	78.39	80.74	85.30	81.45	81.43	75.92
	F-measure	78.41	80.79	85.35	81.40	81.37	75.76

In Table 4.11, the results obtained using cross validation are generally good, with accuracies of at least 90 %. It means that the six algorithms are able to handle NSL-KDD training data remarkably well. However, when it comes to the hold-out test, the results are not as good as those in cross validations. The accuracy of the best algorithm (J48 decision tree) is only 81.5 %. It means that the unseen NSL-KDD test data are far more distinct than the training data, and the classifiers are unable to classify those new data correctly. In spite of the distinctive nature of the NSL-KDD training and test sets, it can be observed that after feature selection, the performance results of the six algorithms are all improved in both cross validations and hold-out tests.

According to Table 4.12, the best three algorithms are J48, RF and RF-BT consecutively. Table 4.13 tabulates the results of the individual J48, RF and RF-BT models, together with the ensemble models of J48 + RF and J48 + RF-BT.

Table 4.13: Performance Results of the Best Three Individual Models and Ensemble Models for NSL-KDD Dataset

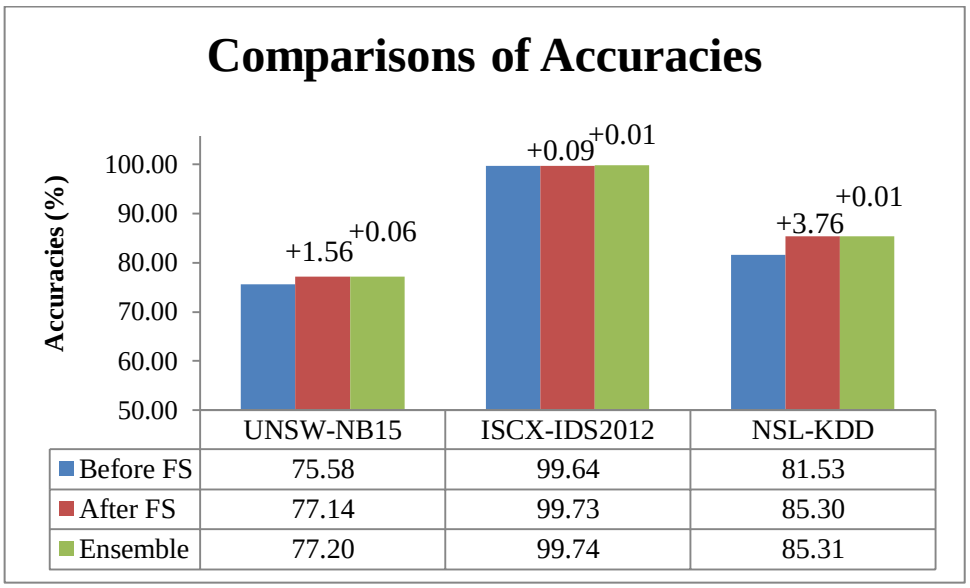
Metrics	Individual Models			Ensemble Models	
	Best	Second Best	Third Best	Best + Second Best	Best + Third Best
	J48	RF	RF-BT	J48 + RF	J48 + RF-BT
Accuracy	85.30	81.45	81.43	85.31	81.89
Precision	87.59	85.48	85.46	87.61	83.62
Recall	85.30	81.45	81.43	85.31	81.89
F-measure	85.35	81.40	81.37	85.36	81.96

The application of ensemble learning which combines the best two algorithms (J48 + RF) shows improvement. It is because the overfitting issue of a J48 decision tree is overcame by the RF by generating more random trees. The higher number of trees results in less variance, as the decisions of individual trees are aggregated in the prediction process. In this case, although J48 alone performs better than RF by about 4 % in accuracy, the combination of both algorithms is capable to enhance the performance of a single decision tree.

Nevertheless, the combination of J48 and RF-BT with tie-breaking capability does not outperform the J48 algorithm. It is because when the features in a random tree are equally good, the tie-breaking capability would only select one feature and discard the rest. In NSL-KDD dataset, the interactions between features could be an important factor in predicting correct output. Hence, the classification accuracy decreases when the interactions of features are disregarded.

4.5 Summary of Results

In this project, the best algorithms for UNSW-NB15, ISCX-IDS2012 and NSL-KDD datasets are random forest, *k*-nearest neighbour and J48 decision tree respectively. Figure 4.1 shows the gradual improvements of these best algorithms in term of accuracy. Specifically, it compares the original datasets without feature selection (i.e., before feature selection), the datasets with feature selection (i.e., after feature selection) and the ensemble model.



	UNSW-NB15	ISCX-IDS2012	NSL-KDD
Before FS	75.58	99.64	81.53
After FS	77.14	99.73	85.30
Ensemble	77.20	99.74	85.31

Figure 4.1: Comparisons of Accuracies Before and After Feature Selection and Ensemble Learning for UNSW-NB15, ISCX-IDS2012 and NSL-KDD Datasets

It can be observed that feature selection technique is able to enhance the classification accuracy for all IDS models. Besides, ensemble learning is also able to further improve the classification accuracy only if the best two machine learning algorithms are provided as the base learners.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

This project has identified several problems in designing an intrusion detection system. These problems include biased evaluation and the use of questionable datasets. This project has successfully addressed these issues by achieving the aim and objectives as follows.

1. This project had investigated the impact of feature selection on the designed intrusion detection system. To ensure the effectiveness of feature selection, the prediction performance of five conventional machine learning algorithms, namely Naïve Bayes, *K*-nearest neighbour, J48 decision tree, random forest and support vector machine against three datasets (i.e., UNSW-NB15, ISCX-IDS2012 and NSL-KDD), before and after performing wrapper-based feature selection with best-first search algorithm, had been compared. Important feature subset had been identified, and subsequently the prediction performance of all machine learning algorithms had been unanimously improved. This had proved the effectiveness of feature selection to improve the prediction performance of a machine learning based intrusion detection system.
2. This project had investigated the impact of feature selection and ensemble learning on the designed intrusion detection system. Two best well-performed algorithms were identified during achieving the first objectives. These two best well-performed algorithms had been combined using ensemble learning to improve the prediction performance of a machine learning based intrusion detection system. The experimental results against three aforementioned datasets have demonstrated that the ensemble model which combines the two best algorithms is able to improve the prediction performance. However, no improvement has been observed when the ensemble model combined the best and the third best machine learning models. Therefore, it can be deduced that ensemble learning is not always guaranteed to improve the classification

performance of the intrusion detection systems. Ensemble learning will only improve classifier performance if the correct base learners are selected.

In summary, the proposed machine learning models that integrate feature selection and ensemble learning are able to classify normal and attack data up to ten different classes.

5.2 Recommendations for Future Work

This final year project also suggests the following areas that are worth to be explored in the future:

1. It is suggested to use different platforms to build the intrusion detection system such as Microsoft Azure Machine Learning Studio. The platform allows the intrusion detection system model to be published as a web service and be deployed in the virtual network environment.
2. Other improvement techniques in machine learning such as boosting and stacking methods can be integrated to investigate the corresponding classification performance. These methods can later be further combined into a single ensemble model to further improve the prediction accuracy.
3. It is suggested to develop the intrusion detection system using the open source software so that the graphical processing unit (GPU) can be selected as the primary processing unit. GPU allows parallel computing and it has optimised memory bandwidth. Therefore, GPU can provide faster processing speed in large memory operation than conventional central processing unit. For example, large memory operation includes matrix multiplication, which is essential for support vector machine model. Github and Kaggle are the two available open source platforms which contain various machine learning source codes in various programming languages.
4. It is suggested to investigate whether deep learning models can be used to classify structured data (which contains a number of features) effectively. These deep learning models include convolutional neural networks, recurrent neural networks and multi-layer perceptron.

REFERENCES

Ahmad, B., Jian, W. and Anwar Ali, Z., 2018. Role of Machine Learning and Data Mining in Internet Security: Standing State with Future Directions. *Journal of Computer Networks and Communications, 2018*.

Akhtar, T., Gupta, B.B. and Yamaguchi, S., 2018, January. Malware propagation effects on SCADA system and smart power grid. In *2018 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 1-6). IEEE.

Ali, J., Khan, R., Ahmad, N. and Maqsood, I., 2012. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, *9*(5), p.272.

Aljawarneh, S., Yassein, M.B. and Aljundi, M., 2017. An enhanced J48 classification algorithm for the anomaly intrusion detection systems. *Cluster Computing*, pp.1-17.

Al-kasassbeh, M., Al-Naymat, G., Hamadneh, N., Obeidat, I. and Almseidin, M., 2018. Intensive Preprocessing of KDD Cup 99 for Network Intrusion Classification Using Machine Learning Techniques. *arXiv preprint arXiv:1805.10458*.

Atli, B.G., Miche, Y., Kalliola, A., Oliver, I., Holtmanns, S. and Lendasse, A., 2018. Anomaly-based intrusion detection using extreme learning machine and aggregation of network traffic statistics in probability space. *Cognitive Computation*, *10*(5), pp.848-863.

Barbosa, R.R.R., 2014. Anomaly detection in SCADA systems: a network based approach.

Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B. and Wingers, L., 2017. Notes on the design and analysis of SIMON and SPECK. *IACR Cryptology ePrint Archive*, *2017*, p.560.

Bhatia, N., 2010. Survey of nearest neighbor techniques. *arXiv preprint arXiv:1007.0085*.

Boyer, S.A., 2004. SCADA: Supervisory Control And Data Acquisition ISA-The Instrumentation. *Systems and Automation Society, USA*,.

Caruana, R. and Niculescu-Mizil, A., 2006, June. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning* (pp. 161-168). ACM.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M. and Elhadad, N., 2015, August. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1721-1730). ACM.

Catal, C., Tufekci, S., Pirmit, E. and Kocabag, G., 2015. On the use of ensemble of classifiers for accelerometer-based activity recognition. *Applied Soft Computing*, *37*, pp.1018-1022.

Erez, N. and Wool, A., 2015. Control variable classification, modeling and anomaly detection in Modbus/TCP SCADA systems. *International Journal of Critical Infrastructure Protection*, *10*, pp.59-70.

Fawagreh, K., Gaber, M.M. and Elyan, E., 2014. Random forests: from early developments to recent advancements. *Systems Science & Control Engineering: An Open Access Journal*, *2*(1), pp.602-609.

Fogie, S. and Peikari, C. (2002). *Going on the Defensive: Intrusion-Detection Systems | Types of IDSs | InformIT*. [online] Informit.com. Available at: http://www.informit.com/articles/article.aspx?p=29601 [Accessed 2 Mar. 2019].

Folino, G., Pisani, F.S. and Sabatino, P., 2016, March. A distributed intrusion detection framework based on evolved specialized ensembles of classifiers. In *European Conference on the Applications of Evolutionary Computation* (pp. 315-331). Springer, Cham.

Gharaee, H. and Hosseinvand, H., 2016, September. A new feature selection IDS based on genetic algorithm and SVM. In *2016 8th International Symposium on Telecommunications (IST)* (pp. 139-144). IEEE.

Gheyas, I.A. and Smith, L.S., 2010. Feature subset selection in large dimensionality domains. *Pattern recognition*, *43*(1), pp.5-13.

Haider, W., Hu, J. and Moustafa, N., 2017, December. Designing Anomaly Detection System for Cloud Servers by Frequency Domain Features of System Call Identifiers and Machine Learning. In *International Conference on Mobile Networks and Management* (pp. 137-149). Springer, Cham.

Haider, W., Hu, J., Slay, J., Turnbull, B.P. and Xie, Y., 2017. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Applications*, *87*, pp.185-192.

Hira, Z.M. and Gillies, D.F., 2015. A review of feature selection and feature extraction methods applied on microarray data. *Advances in bioinformatics*, *2015*.

Hssina, B., Merbouha, A., Ezzikouri, H. and Erritali, M., 2014. A comparative study of decision tree ID3 and C4. 5. *International Journal of Advanced Computer Science and Applications*, *4*(2), pp.0-0.

Hu, Z., Bao, Y., Xiong, T. and Chiong, R., 2015. Hybrid filter–wrapper feature selection for short-term load forecasting. *Engineering Applications of Artificial Intelligence*, *40*, pp.17-27.

Jabbar, M.A. and Aluvalu, R., 2017. RFAODE: A novel ensemble intrusion detection system. *Procedia computer science*, *115*, pp.226-234.

Jadhav, S.D. and Channe, H.P., 2016. Comparative study of K-NN, naive Bayes and decision tree classification techniques. *International Journal of Science and Research*, *5*(1), pp.1842-1845.

Kabir, E. and Zhang, Y., 2016. Epileptic seizure detection from EEG signals using logistic model trees. *Brain informatics*, *3*(2), pp.93-100.

Kaur, R., Sachdeva, M. and Kumar, G., 2016. Study and comparison of feature selection approaches for intrusion detection. *Int. J. Comput. Appl*, *7*, p.6.

Kelleher, J.D., Namee, B. M. and D'arcy, A., 2015. *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press.

Kim, T., Chung, B.D. and Lee, J.S., 2017. Incorporating receiver operating characteristics into naive Bayes for unbalanced data classification. *Computing*, *99*(3), pp.203-218.

Kohavi, R. and John, G.H., 1997. Wrappers for feature subset selection. *Artificial intelligence*, *97*(1-2), pp.273-324.

Kohavi, R., 1995, August. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (Vol. 14, No. 2, pp. 1137-1145).

Liao, Y. and Vemuri, V.R., 2002. Use of k-nearest neighbor classifier for intrusion detection. *Computers & security*, *21*(5), pp.439-448.

Liaw, A. and Wiener, M., 2002. Classification and regression by randomForest. *R news*, *2*(3), pp.18-22.

Longadge, R. and Dongre, S., 2013. Class imbalance problem in data mining review. *arXiv preprint arXiv:1305.1707*.

Lowd, D. and Domingos, P., 2005, August. Naive Bayes models for probability estimation. In *Proceedings of the 22nd international conference on Machine learning* (pp. 529-536). ACM.

Malli, R.C., Aygun, M. and Ekenel, H.K., 2016. Apparent age estimation using ensemble of deep learning models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 9-16).

Marsland, S., 2011. *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC.

Maynard, P., McLaughlin, K. and Haberler, B., 2014, September. Towards Understanding Man-in-the-middle Attacks on IEC 60870-5-104 SCADA Networks. In *ICS-CSR*.

McHugh, J., 2000. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, *3*(4), pp.262-294.

McLachlan, G., Do, K.A. and Ambroise, C., 2005. *Analyzing microarray gene expression data* (Vol. 422). John Wiley & Sons.

Mirza, A.H. and Cosan, S., 2018, May. Computer network intrusion detection using sequential LSTM Neural Networks autoencoders. In *2018 26th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE.

Mitchell, R. and Chen, I.R., 2014. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys (CSUR)*, *46*(4), p.55.

Mo, Y., Chabukswar, R. and Sinopoli, B., 2014. Detecting integrity attacks on SCADA systems. *IEEE Transactions on Control Systems Technology*, *22*(4), pp.1396-1407.

Mocherla, S., Danehy, A. and Impey, C., 2017. Evaluation of Naive Bayes and Support Vector Machines for Wikipedia. *Applied Artificial Intelligence*, *31*(9-10), pp.733-744.

Mohammed, M., Khan, M.B. and Bashier, E.B.M., 2016. *Machine learning: algorithms and applications*. Crc Press.

Moustafa, N. and Slay, J., 2015, November. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 military communications and information systems conference (MilCIS)*(pp. 1-6). IEEE.

Park, K., Song, Y. and Cheong, Y.G., 2018, March. Classification of Attack Types for Intrusion Detection Systems Using a Machine Learning Algorithm. In *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)* (pp. 282-286). IEEE.

Patel, N. and Upadhyay, S., 2012. Study of various decision tree pruning methods with their empirical comparison in WEKA. *International journal of computer applications*, *60*(12).

Poria, S., Cambria, E., Hazarika, D., Majumder, N., Zadeh, A. and Morency, L.P., 2017, July. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 873-883).

Rao, U.H. and Nayak, U., 2014. Intrusion Detection and Prevention Systems. In *The InfoSec Handbook* (pp. 225-243). Apress, Berkeley, CA.

Rennie, J.D., Shih, L., Teevan, J. and Karger, D.R., 2003. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)* (pp. 616-623).

Revathi, S. and Malathi, A., 2013. A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology (IJERT)*, *2*(12), pp.1848-1853.

Rish, I., 2001, August. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (Vol. 3, No. 22, pp. 41-46).

Ruggieri, S., 2002. Efficient C4. 5 [classification algorithm]. *IEEE transactions on knowledge and data engineering*, *14*(2), pp.438-444.

Safavian, S.R. and Landgrebe, D., 1991. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, *21*(3), pp.660-674.

Shiravi, A., Shiravi, H., Tavallaee, M. and Ghorbani, A.A., 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, *31*(3), pp.357-374.

Shitharth, S. and Winston, D.P., 2015. A comparative analysis between two countermeasure techniques to detect DDoS with sniffers in a SCADA network. *Procedia Technology*, *21*, pp.179-186.

Skiena, S.S., 1998. *The algorithm design manual: Text* (Vol. 1). Springer Science & Business Media.

Strobl, C., Boulesteix, A.L., Zeileis, A. and Hothorn, T., 2007. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, *8*(1), p.25.

Suleiman, M. and Issac, B., 2018. Performance Comparison of Intrusion Detection Machine Learning Classifiers on Benchmark and New Datasets.

Sung, A.H. and Mukkamala, S., 2003, January. Identifying important features for intrusion detection using support vector machines and neural networks. In *2003 Symposium on Applications and the Internet, 2003. Proceedings.* (pp. 209-216). IEEE.

Swarnkar, M. and Hubballi, N., 2016. OCPAD: One class Naive Bayes classifier for payload based anomaly detection. *Expert Systems with Applications*, *64*, pp.330-339.

Syarif, A.R. and Gata, W., 2017, October. Intrusion detection system using hybrid binary PSO and K-nearest neighborhood algorithm. In *2017 11th International Conference on Information & Communication Technology and System (ICTS)*(pp. 181-186). IEEE.

Tang, B., Kay, S. and He, H., 2016. Toward optimal feature selection in naive Bayes for text categorization. *IEEE transactions on knowledge and data engineering*, *28*(9), pp.2508-2521.

Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A., 2009, July. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications* (pp. 1-6). IEEE.

Wang, H., Khoshgoftaar, T.M. and Gao, K., 2010, August. A comparative study of filter-based feature ranking techniques. In *2010 IEEE International Conference on Information Reuse & Integration* (pp. 43-48). IEEE.

Witten, I.H., Frank, E., Hall, M.A. and Pal, C.J., 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Yassin, W., Udzir, N.I., Muda, Z. and Sulaiman, M.N., 2013, August. Anomaly-based intrusion detection through k-means clustering and naives bayes classification. In *Proc. 4th Int. Conf. Comput. Informatics, ICOCI* (Vol. 49, pp. 298-303).

Yu, L. and Liu, H., 2003. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)* (pp. 856-863).

Zeng, Z., Zhang, H., Zhang, R. and Yin, C., 2015. A novel feature selection method considering feature interaction. *Pattern Recognition*, *48*(8), pp.2656-2666.

Zhang, Y. and Yang, Y., 2015. Cross-validation for selecting a model selection procedure. *Journal of Econometrics*, *187*(1), pp.95-112.

Zhang, Y., Wang, L., Xiang, Y. and Ten, C.W., 2016. Inclusion of SCADA cyber vulnerability in power system reliability assessment considering optimal resources allocation. *IEEE Transactions on Power Systems*, *31*(6), pp.4379-4394.

Zhou, X., Wang, S., Xu, W., Ji, G., Phillips, P., Sun, P. and Zhang, Y., 2015, April. Detection of pathological brain in MRI scanning based on wavelet-entropy and naive Bayes classifier. In *International conference on bioinformatics and biomedical engineering* (pp. 201-209). Springer, Cham.

Zhou, Z.H., 2012. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.

Zou, Q., Guo, J., Ju, Y., Wu, M., Zeng, X. and Hong, Z., 2015. Improving tRNAscan-SE Annotation Results via Ensemble Classifiers. *Molecular informatics*, *34*(11-12), pp.761-770.

APPENDICES

APPENDIX A: Dataset Downloads

- UNSW-NB15
  https://cloudstor.aarnet.edu.au/plus/index.php/s/2DhnLGDdEECo4ys

- ISCX-IDS2012
  https://iscxdownloads.cs.unb.ca/iscxdownloads/ISCX-IDS-2012/#ISCX-IDS-2012

- NSL-KDD
  https://iscxdownloads.cs.unb.ca/iscxdownloads/NSL-KDD/#NSL-KDD

APPENDIX B: Python Codes (XML-to-CSV converter for ISCX-IDS2012)

```
import re

def conversion (raw_feature):
    searchObj = re.search(r'\>.*\<\/', raw_feature, re.M|re.I)
    searchStr = searchObj.group()
    processedStr = searchStr[1:len(searchStr)-2]
    return processedStr;

###       Calculate number of features
###       Remember to change here
f = open('C:\Users\User\Desktop\Papers\
        Dataset\ISCXIDS\TestbedWedJun16-3Flows.xml', 'r')   # VARIABLE: INPUT
f2 = open('C:\Users\User\Desktop\Papers\Dataset\
          ISCXIDS\TestbedWedJun16-3Flows.csv', 'w')     # VARIABLE: OUTPUT
nObject = 0
totalObject = 196924.0                                  # VARIABLE: INSTANCE

#   Ensure no of features here
f2.write("appName, totalSourceBytes, totalDestinationBytes, \
totalDestinationPackets, totalSourcePackets, sourcePayloadAsBase64, \
sourcePayloadAsUTF, \
destinationPayloadAsBase64, destinationPayloadAsUTF, direction, \
sourceTCPFlagsDescription, destinationTCPFlagsDescription, source, \
protocolName,sourcePort, destination, destinationPort, startDateTime, \
stopDateTime, Tag\n")

EOF = "0";
while EOF != "":
    ###       Remember to change here
    if f.readline() == "<TestbedWedJun16-3Flows>\n":    # VARIABLE: SUB-HEADER
        nObject = nObject + 1
        print ("%.2f" % round(nObject/totalObject*100,2))
        # No of features
        featureListRaw = [f.readline(),f.readline(),f.readline(),f.readline(),
                          f.readline(),f.readline(),f.readline(),f.readline(),
                          f.readline(),f.readline(),f.readline(),f.readline(),
                          f.readline(),f.readline(),f.readline(),f.readline(),
                          f.readline(),f.readline(),f.readline(),f.readline()]

        #featureListRaw.pop(13)     # pop unnecessary feature, if any

        featureListProcessed = []

        for i in range(len(featureListRaw)):
            feature_buffer = conversion(featureListRaw[i])
            gotComma = re.search(r'\,', feature_buffer, re.M|re.I)
            if (gotComma != None):
                feature_buffer = "\""+feature_buffer+"\""
            featureListProcessed.append(feature_buffer)

        for i in range(len(featureListProcessed)):
            f2.write(featureListProcessed[i]+",",)
        f2.write("\n")
    EOF = f.readline()
f.close()
f2.close()
```

APPENDIX C: Python Codes (ISCX-IDS2012: Removal of Duplicates, Undersampling of Normal Majority Class and Train-Test Split of 70:30)

```
import re
import random

#   Pre-record the data into lists
percent_of_train_data_to_all = 70.0
num_of_normal_sample = 20000.00

f = open(r'C:\Users\User\Desktop\Papers\
         Dataset\ISCXIDS\csv_mix\raw\TestbedAll.csv','r')
bruteForce_list = []
httpDos_list = []
infiltration_list = []
ddos_list = []
normal_list = []
total_num = 2071657.0
i = 1.0

header = f.readline()
EOF = f.readline()
while EOF != "":
    buff = re.search(r'Normal$', EOF, re.M|re.I)
    if buff == None:
        buff2 = re.search(r'bruteForce$', EOF, re.M|re.I)
        if buff2 != None:
            bruteForce_list.append(EOF)
        else:
            buff3 = re.search(r'httpDos$', EOF, re.M|re.I)
            if buff3 != None:
                httpDos_list.append(EOF)
            else:
                buff4 = re.search(r'infiltration$', EOF, re.M|re.I)
                if buff4 != None:
                    infiltration_list.append(EOF)
                else:
                    buff5 = re.search(r'DDos$', EOF, re.M|re.I)
                    if buff5 != None:
                        ddos_list.append(EOF)
    else :

        normal_list.append(EOF)
    print ("%.2f" % round(i/total_num*100,2))+" %"
    i=i+1
    EOF = f.readline()
f.close()
print ("Finish recording the data. \
       Removing duplicates if any. Generating train set...")

bruteForce_list = list(set(bruteForce_list))
httpDos_list = list(set(httpDos_list))
infiltration_list = list(set(infiltration_list))
ddos_list = list(set(ddos_list))
normal_list = list(set(normal_list))
normal_list = random.sample(normal_list, int(num_of_normal_sample))
```

APPENDIX C (continued)

```
no_of_bruteForce = len(bruteForce_list)
no_of_httpDos = len(httpDos_list)
no_of_infiltration = len(infiltration_list)
no_of_ddos = len(ddos_list)
no_of_normal = len(normal_list)
total_datapoint = no_of_bruteForce + no_of_httpDos \
+ no_of_infiltration + no_of_ddos + no_of_normal

no_of_bruteForce_train = int(no_of_bruteForce*
                             percent_of_train_data_to_all/100.0)
no_of_httpDos_train = int(no_of_httpDos*percent_of_train_data_to_all/100.0)
no_of_infiltration_train = int(no_of_infiltration*
                               percent_of_train_data_to_all/100.0)
no_of_ddos_train = int(no_of_ddos*percent_of_train_data_to_all/100.0)
no_of_normal_train = int(no_of_normal*percent_of_train_data_to_all/100.0)

no_of_bruteForce_test = no_of_bruteForce - no_of_bruteForce_train
no_of_httpDos_test = no_of_httpDos - no_of_httpDos_train
no_of_infiltration_test = no_of_infiltration - no_of_infiltration_train
no_of_ddos_test = no_of_ddos - no_of_ddos_train
no_of_normal_test = no_of_normal - no_of_normal_train

bruteForce_train_list = random.sample(bruteForce_list, no_of_bruteForce_train)
httpDos_train_list = random.sample(httpDos_list, no_of_httpDos_train)
infiltration_train_list = random.sample(infiltration_list,
                                        no_of_infiltration_train)
ddos_train_list = random.sample(ddos_list, no_of_ddos_train)
normal_train_list = random.sample(normal_list, no_of_normal_train)

train_list = bruteForce_train_list + httpDos_train_list + \
infiltration_train_list + ddos_train_list + normal_train_list
print "Completed train set first built. Randomizing sequence..."
train_list = random.sample(train_list, len(train_list))
print "Sample training dataset: Complete. Generating test set..."
```

```
bruteForce_test_list = list(set(bruteForce_list)^set(bruteForce_train_list))
httpDos_test_list = list(set(httpDos_list)^set(httpDos_train_list))
infiltration_test_list = list(set(infiltration_list)^
                              set(infiltration_train_list))
ddos_test_list = list(set(ddos_list)^set(ddos_train_list))
normal_test_list = list(set(normal_list)^set(normal_train_list))

test_list = bruteForce_test_list + httpDos_test_list \
+ infiltration_test_list + ddos_test_list + normal_test_list
print "Completed test set first built. Randomizing sequence..."
test_list = random.sample(test_list, len(test_list))
print "Sample testing dataset: Complete. Output train and test sets into files"

print "Exporting to train sample csv file"
fTrain = open(r'C:\Users\User\Desktop\
              Papers\Dataset\ISCXIDS\csv_mix\train_set.csv','a')
fTrain.write(header)
for item in range(len(train_list)):
    fTrain.write(train_list[item])
    print ("Train: %.2f" % round(float(float(item)*100/len(train_list)),2))+"%"
fTrain.close()
```

APPENDIX C (continued)

```
print "Exporting to test sample csv file"
fTest = open(r'C:\Users\User\Desktop\Papers\
             Dataset\ISCXIDS\csv_mix\test_set.csv','a')
fTest.write(header)
for item in range(len(test_list)):
    fTest.write(test_list[item])
    print ("Test: %.2f" % round(float(float(item)*100/len(test_list)),2))+" %"
fTest.close()
```

APPENDIX D: Confusion Matrices for UNSW-NB15 Dataset

N: Normal	B: Backdoor	A: Analysis	F: Fuzzers	S: Shellcode
R: Recon	E: Exploits	D: DoS	W: Worm	G: Generic

Appendix D.1: Initial Cross Validation for Naïve Bayes

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**38363**	2025	2596	3201	6228	411	2519	89	137	431	56000
	B	0	**1003**	58	11	585	4	10	4	22	49	1746
	A	13	964	**537**	7	401	6	4	3	8	57	2000
	F	132	1888	78	**4160**	9973	1327	135	12	48	431	18184
	S	0	0	0	22	**1106**	0	0	0	0	5	1133
	R	1	1238	60	148	8850	**26**	18	4	17	129	10491
	E	562	8509	2208	1171	8607	428	**8513**	120	2831	444	33393
	D	35	6471	564	213	3350	168	560	**64**	471	368	12264
	W	1	0	2	7	91	2	0	0	**27**	0	130
	G	19	295	34	49	312	46	114	13	161	**38957**	40000
Σ(Predicted)		39126	22393	6137	8989	39503	2418	11873	309	3722	40871	175341
												Weighted Average
Accuracy												52.90
Precision		98.05	4.48	8.75	46.28	2.80	1.08	71.70	20.71	0.73	95.32	73.19
Recall		68.51	57.45	26.85	22.88	97.62	0.25	25.49	0.52	20.77	97.39	52.90
F-measure		80.66	8.31	13.20	30.62	5.44	0.40	37.61	1.02	1.40	96.34	58.44

Example:

$$\text{Accuracy} = \frac{38363 + 1003 + 537 + \cdots + 38957}{175341} \times 100\ \% = 52.90\ \%$$

$$\text{Precision (Normal)} = \frac{38363}{39126} \times 100\ \% = 98.05\ \%$$

$$\text{Precision}_{\text{avg.}} = \frac{(98.05 \cdot 56000 + 4.48 \cdot 1746 + \cdots + 95.32 \cdot 40000)\%}{175341} = 73.19\ \%$$

$$\text{Recall(Normal)} = \frac{38363}{56000} \times 100\ \% = 68.51\ \%$$

$$\text{Recall}_{\text{avg.}} = \frac{(68.51 \cdot 56000 + 57.45 \cdot 1746 + \cdots + 97.39 \cdot 40000)\%}{175341} = 52.90\ \%$$

$$\text{F} - \text{measure(Normal)} = 2 \times \frac{98.05 \times 68.51}{98.05 + 68.51}\% = 80.66\ \%$$

$$\text{F} - \text{measure}_{\text{avg.}} = \frac{(80.66 \cdot 56000 + 8.31 \cdot 1746 + \cdots + 96.34 \cdot 40000)\%}{175341} = 58.44\ \%$$

Appendix D.2: Initial Cross Validation for *K*-Nearest Neighbour

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**50861**	9	121	3829	56	343	618	142	1	20	56000
	B	25	**62**	79	47	6	86	1122	310	4	5	1746
	A	116	89	**327**	6	0	0	1184	278	0	0	2000
	F	4315	46	9	**10409**	158	786	2028	383	12	38	18184
	S	72	8	0	184	**357**	394	73	38	0	7	1133
	R	374	65	1	654	232	**6118**	2523	493	11	20	10491
	E	735	65	91	1002	112	1280	**25709**	4091	71	237	33393
	D	163	17	29	191	56	189	10075	**1481**	3	60	12264
	W	0	0	0	11	1	20	73	4	**16**	5	130
	G	23	5	5	59	10	43	457	209	4	**39185**	40000
Σ(Predicted)		56684	366	662	16392	988	9259	43862	7429	122	39577	175341
												Weighted Average
Accuracy												76.72
Precision		89.73	16.94	49.40	63.50	36.13	66.08	58.61	19.94	13.11	99.01	75.31
Recall		90.82	3.55	16.35	57.24	31.51	58.32	76.99	12.08	12.31	97.96	76.72
F-measure		90.27	5.87	24.57	60.21	33.66	61.95	66.56	15.04	12.70	98.48	75.54

Appendix D.3: Initial Cross Validation for J48 Decision Tree

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**51783**	5	132	3637	32	28	326	51	1	5	56000
	B	3	**245**	22	9	10	9	1419	25	1	3	1746
	A	171	58	**313**	8	0	0	1418	32	0	0	2000
	F	2839	4	11	**13415**	116	21	1695	57	5	21	18184
	S	37	12	0	100	**812**	16	117	33	0	6	1133
	R	13	6	2	17	5	**7907**	2469	67	3	2	10491
	E	315	25	56	298	143	674	**30994**	728	39	121	33393
	D	60	15	19	84	72	69	10774	**1127**	1	43	12264
	W	1	2	0	2	0	1	38	2	**82**	2	130
	G	11	6	2	30	9	5	507	79	5	**39346**	40000
Σ(Predicted)		55233	378	557	17600	1199	8730	49757	2201	137	39549	175341
												Weighted Average
Accuracy												83.28
Precision		93.75	64.81	56.19	76.22	67.72	90.57	62.29	51.20	59.85	99.49	83.18
Recall		92.47	14.03	15.65	73.77	71.67	75.37	92.82	9.19	63.08	98.37	83.28
F-measure		93.11	23.07	24.48	74.98	69.64	82.27	74.55	15.58	61.42	98.92	81.29

Appendix D.4: Initial Cross Validation for Random Forest without Tie-Breaking Capability

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**52138**	0	26	3413	34	36	340	11	0	2	56000
	B	3	**208**	78	25	12	13	1049	357	0	1	1746
	A	184	82	**313**	9	0	0	1062	350	0	0	2000
	F	2584	13	15	**13719**	98	22	1382	336	1	14	18184
	S	31	0	0	149	**771**	11	152	16	0	3	1133
	R	7	6	0	17	3	**7905**	2025	526	0	2	10491
	E	216	4	13	323	135	566	**28497**	3582	14	43	33393
	D	25	5	7	93	96	60	9751	**2212**	4	11	12264
	W	0	0	0	6	0	1	93	3	**26**	1	130
	G	8	5	0	33	10	2	453	217	1	**39271**	40000
Σ(Predicted)		55196	323	452	17787	1159	8616	44804	7610	46	39348	175341
												Weighted Average
Accuracy												82.73
Precision		94.46	64.40	69.25	77.13	66.52	91.75	63.60	29.07	56.52	99.80	82.47
Recall		93.10	11.91	15.65	75.45	68.05	75.35	85.34	18.04	20.00	98.18	82.73
F-measure		93.78	20.11	25.53	76.28	67.28	82.74	72.89	22.26	29.55	98.98	81.78

Appendix D.5: Initial Cross Validation for Random Forest with Tie-Breaking Capability

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**52131**	0	21	3430	33	34	340	9	0	2	56000
	B	1	**210**	78	27	10	11	1050	357	1	1	1746
	A	185	83	**309**	9	0	0	1065	349	0	0	2000
	F	2586	13	14	**13707**	96	21	1392	340	1	14	18184
	S	28	0	0	149	**774**	12	149	18	0	3	1133
	R	6	6	0	16	5	**7909**	2019	527	0	3	10491
	E	218	6	16	328	128	559	**28492**	3580	16	50	33393
	D	24	7	5	97	100	62	9748	**2206**	4	11	12264
	W	0	0	0	6	0	0	90	4	**29**	1	130
	G	10	5	0	37	11	2	448	215	1	**39271**	40000
Σ(Predicted)		55189	330	443	17806	1157	8610	44793	7605	52	39356	175341
												Weighted Average
Accuracy												82.72
Precision		94.46	63.64	69.75	76.98	66.90	91.86	63.61	29.01	55.77	99.78	82.46
Recall		93.09	12.03	15.45	75.38	68.31	75.39	85.32	17.99	22.31	98.18	82.72
F-measure		93.77	20.23	25.30	76.17	67.60	82.81	72.88	22.21	31.87	98.97	81.76

Appendix D.6: Initial Cross Validation for Support Vector Machine

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**45827**	0	39	8871	0	468	790	4	0	1	56000
	B	10	**3**	0	64	0	98	1571	0	0	0	1746
	A	202	0	**131**	2	0	0	1665	0	0	0	2000
	F	703	0	0	**14167**	0	1290	1976	2	0	46	18184
	S	6	0	0	428	**0**	699	0	0	0	0	1133
	R	57	0	0	1096	0	**5908**	3395	0	0	35	10491
	E	359	8	0	2405	0	516	**30053**	29	0	23	33393
	D	114	3	0	444	0	215	11388	**37**	0	63	12264
	W	2	0	0	11	0	6	111	0	**0**	0	130
	G	14	0	0	156	0	39	675	0	0	**39116**	40000
Σ(Predicted)		47294	14	170	27644	0	9239	51624	72	0	39284	175341
												Weighted Average
Accuracy												77.13
Precision		96.90	21.43	77.06	51.25	NaN	63.95	58.22	51.39	NaN	99.57	NaN
Recall		81.83	0.17	6.55	77.91	0.00	56.31	90.00	0.30	0.00	97.79	77.13
F-measure		88.73	0.34	12.07	61.83	NaN	59.89	70.70	0.60	NaN	98.67	NaN

Appendix D.7: Final Cross Validation for Naïve Bayes

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**43186**	45	238	9094	2	1311	1840	64	0	220	56000
	B	66	**3**	8	166	0	0	928	547	0	28	1746
	A	15	5	**295**	0	0	0	1090	559	0	36	2000
	F	668	5	128	**11372**	0	2613	2636	533	0	229	18184
	S	0	0	0	591	**0**	494	38	0	0	10	1133
	R	90	7	42	3782	0	**3000**	2743	701	0	126	10491
	E	636	53	215	3577	20	213	**23675**	4575	0	429	33393
	D	415	29	52	566	21	137	7193	**3588**	0	263	12264
	W	2	0	8	4	0	11	104	0	**0**	1	130
	G	1194	0	12	124	8	79	549	77	0	**37957**	40000
Σ(Predicted)		46272	147	998	29276	51	7858	40796	10644	0	39299	175341
												Weighted Average
Accuracy												70.19
Precision		93.33	2.04	29.56	38.84	0.00	38.18	58.03	33.71	NaN	96.59	NaN
Recall		77.12	0.17	14.75	62.54	0.00	28.60	70.90	29.26	0.00	94.89	70.19
F-measure		84.45	0.32	19.68	47.92	0.00	32.70	63.82	31.33	NaN	95.73	NaN

Appendix D.8: Final Cross Validation for *K*-Nearest Neighbour

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**52270**	4	156	3038	38	66	375	42	1	10	56000
	B	4	**216**	19	18	9	16	1316	147	0	1	1746
	A	167	31	**318**	12	0	3	1340	125	0	4	2000
	F	3278	9	12	**12893**	87	56	1624	194	5	26	18184
	S	48	13	0	92	**856**	37	66	19	1	1	1133
	R	34	3	3	49	15	**7902**	2192	284	4	5	10491
	E	383	39	68	462	146	774	**29174**	2154	39	154	33393
	D	72	14	12	127	49	82	9847	**2029**	2	30	12264
	W	0	0	0	4	0	5	36	2	**82**	1	130
	G	16	6	3	31	10	7	536	70	8	**39313**	40000
Σ(Predicted)		56272	335	591	16726	1210	8948	46506	5066	142	39545	175341
												Weighted Average
Accuracy												82.73
Precision		92.89	64.48	53.81	77.08	70.74	88.31	62.73	40.05	57.75	99.41	82.13
Recall		93.34	12.37	15.90	70.90	75.55	75.32	87.37	16.54	63.08	98.28	82.73
F-measure		93.11	20.76	24.55	73.86	73.07	81.30	73.03	23.42	60.29	98.84	81.36

Appendix D.9: Final Cross Validation for J48 Decision Tree

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**52149**	2	93	3297	35	34	345	39	0	6	56000
	B	3	**262**	20	8	14	12	1410	13	1	3	1746
	A	182	48	**300**	5	0	1	1438	26	0	0	2000
	F	2681	6	8	**13575**	102	17	1735	44	2	14	18184
	S	29	8	0	94	**863**	24	102	9	0	4	1133
	R	14	2	1	15	6	**7958**	2447	42	3	3	10491
	E	254	12	44	292	126	567	**31474**	488	36	100	33393
	D	46	12	9	89	62	76	10826	**1099**	1	44	12264
	W	0	0	0	2	0	1	29	2	**96**	0	130
	G	17	6	3	26	11	4	522	53	4	**39354**	40000
Σ(Predicted)		55375	358	478	17403	1219	8694	50328	1815	143	39528	175341
												Weighted Average
Accuracy												83.91
Precision		94.17	73.18	62.76	78.00	70.80	91.53	62.54	60.55	67.13	99.56	84.45
Recall		93.12	15.01	15.00	74.65	76.17	75.86	94.25	8.96	73.85	98.39	83.91
F-measure		93.65	24.90	24.21	76.29	73.38	82.96	75.19	15.61	70.33	98.97	81.82

Appendix D.10: Final Cross Validation for Random Forest without Tie-Breaking Capability

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**52526**	1	42	3015	36	35	323	20	0	2	56000
	B	1	**218**	28	14	15	11	1300	157	1	1	1746
	A	196	22	**318**	6	0	0	1311	147	0	0	2000
	F	2351	7	11	**13911**	105	13	1596	171	2	17	18184
	S	24	5	0	98	**865**	14	111	14	1	1	1133
	R	16	6	1	9	4	**7920**	2213	318	1	3	10491
	E	222	7	17	266	131	640	**30278**	1742	33	57	33393
	D	41	4	7	69	69	57	9996	**2002**	4	15	12264
	W	0	0	0	4	0	1	46	2	**75**	2	130
	G	6	5	1	31	9	2	533	57	3	**39353**	40000
Σ(Predicted)		55383	275	425	17423	1234	8693	47707	4630	120	39451	175341
												Weighted Average
Accuracy												84.10
Precision		94.84	79.27	74.82	79.84	70.10	91.11	63.47	43.24	62.50	99.75	84.03
Recall		93.80	12.49	15.90	76.50	76.35	75.49	90.67	16.32	57.69	98.38	84.10
F-measure		94.32	21.57	26.23	78.14	73.09	82.57	74.67	23.70	60.00	99.06	82.67

Appendix D.11: Final Cross Validation for Random Forest with Tie-Breaking Capability

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**52544**	1	39	3001	30	35	327	21	0	2	56000
	B	2	**219**	26	10	15	12	1300	160	1	1	1746
	A	196	25	**312**	6	0	0	1311	150	0	0	2000
	F	2344	7	9	**13920**	102	10	1590	180	3	19	18184
	S	26	4	0	101	**863**	12	106	19	1	1	1133
	R	14	6	1	8	4	**7934**	2195	324	2	3	10491
	E	236	7	17	262	135	621	**30249**	1775	32	59	33393
	D	38	4	7	72	69	58	9976	**2022**	4	14	12264
	W	0	0	0	4	1	0	50	1	**72**	2	130
	G	8	4	0	31	9	3	532	56	4	**39353**	40000
Σ(Predicted)		55408	277	411	17415	1228	8685	47636	4708	119	39454	175341
												Weighted Average
Accuracy												84.11
Precision		94.83	79.06	75.91	79.93	70.28	91.35	63.50	42.95	60.50	99.74	84.05
Recall		93.83	12.54	15.60	76.55	76.17	75.63	90.58	16.49	55.38	98.38	84.11
F-measure		94.33	21.65	25.88	78.20	73.10	82.75	74.66	23.83	57.83	99.06	82.70

Appendix D.12: Final Cross Validation for Support Vector Machine

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**45785**	0	43	8873	0	470	829	0	0	0	56000
	B	9	**3**	0	64	0	126	1544	0	0	0	1746
	A	187	0	**130**	2	0	0	1681	0	0	0	2000
	F	691	0	0	**14175**	0	1289	1981	1	0	47	18184
	S	6	0	0	426	**0**	701	0	0	0	0	1133
	R	51	0	0	1119	0	**5889**	3397	0	0	35	10491
	E	328	8	0	2423	0	520	**30062**	12	0	40	33393
	D	93	1	0	482	0	227	11385	**0**	0	76	12264
	W	2	0	0	12	0	4	112	0	**0**	0	130
	G	14	0	0	157	0	38	675	0	0	**39116**	40000
Σ(Predicted)		47166	12	173	27733	0	9264	51666	13	0	39314	175341
												Weighted Average
Accuracy												77.08
Precision		97.07	25.00	75.14	51.11	NaN	63.57	58.19	0.00	NaN	99.50	NaN
Recall		81.76	0.17	6.50	77.95	0.00	56.13	90.02	0.00	0.00	97.79	77.08
F-measure		88.76	0.34	11.97	61.74	NaN	59.62	70.69	0.00	NaN	98.64	NaN

Appendix D.13: Initial Hold-out Test for Naïve Bayes

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**16283**	1958	4033	3387	9199	549	1227	77	69	218	37000
	B	0	**363**	15	2	52	0	2	0	3	146	583
	A	2	410	**57**	2	60	0	0	0	0	146	677
	F	64	1024	3	**1298**	2765	417	36	1	15	439	6062
	S	0	0	0	5	**373**	0	0	0	0	0	378
	R	0	234	7	29	3181	**8**	7	0	5	25	3496
	E	248	2234	787	453	2281	192	**3569**	31	948	389	11132
	D	17	1968	250	97	934	76	249	**31**	200	267	4089
	W	0	1	0	2	32	0	0	0	**9**	0	44
	G	31	415	34	57	403	52	107	7	172	**17593**	18871
Σ(Predicted)		16645	8607	5186	5332	19280	1294	5197	147	1421	19223	82332
												Weighted Average
Accuracy												48.08
Precision		97.83	4.22	1.10	24.34	1.93	0.62	68.67	21.09	0.63	91.52	77.14
Recall		44.01	62.26	8.42	21.41	98.68	0.23	32.06	0.76	20.45	93.23	48.08
F-measure		60.71	7.90	1.94	22.78	3.79	0.33	43.71	1.46	1.23	92.37	56.22

Appendix D.14: Initial Hold-out Test for *K*-Nearest Neighbour

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**27184**	39	585	6779	152	823	1114	278	4	42	37000
	B	6	**97**	0	4	3	18	372	81	1	1	583
	A	1	201	**0**	0	0	0	387	85	0	3	677
	F	1609	389	0	**2574**	45	286	967	181	4	7	6062
	S	27	4	0	56	**100**	136	36	16	1	2	378
	R	131	261	4	245	118	**2238**	416	70	5	8	3496
	E	316	1824	81	396	73	478	**6962**	888	20	94	11132
	D	77	1809	27	110	26	95	1518	**397**	3	27	4089
	W	1	0	0	3	1	7	25	1	**4**	2	44
	G	28	8	0	84	15	37	359	101	7	**18232**	18871
Σ(Predicted)		29380	4632	697	10251	533	4118	12156	2098	49	18418	82332
												Weighted Average
Accuracy												70.19
Precision		92.53	2.09	0.00	25.11	18.76	54.35	57.27	18.92	8.16	98.99	77.22
Recall		73.47	16.64	0.00	42.46	26.46	64.02	62.54	9.71	9.09	96.61	70.19
F-measure		81.90	3.72	0.00	31.56	21.95	58.79	59.79	12.83	8.60	97.79	72.89

Appendix D.15: Initial Hold-out Test for J48 Decision Tree

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**27608**	15	749	7644	144	19	716	77	1	27	37000
	B	18	**88**	0	4	4	1	456	6	0	6	583
	A	17	89	**0**	1	0	0	566	4	0	0	677
	F	1317	210	0	**2866**	189	14	1359	86	0	21	6062
	S	9	3	0	23	**287**	4	37	14	0	1	378
	R	18	105	1	21	21	**2831**	478	20	0	1	3496
	E	171	875	11	156	76	214	**9321**	231	4	73	11132
	D	40	840	4	50	33	28	2539	**510**	1	44	4089
	W	2	0	0	0	0	0	10	0	**31**	1	44
	G	18	4	0	23	17	1	316	64	5	**18423**	18871
Σ(Predicted)		29218	2229	765	10788	771	3112	15798	1012	42	18597	82332
												Weighted Average
Accuracy												75.26
Precision		94.49	3.95	0.00	26.57	37.22	90.97	59.00	50.40	73.81	99.06	81.71
Recall		74.62	15.09	0.00	47.28	75.93	80.98	83.73	12.47	70.45	97.63	75.26
F-measure		83.39	6.26	0.00	34.02	49.96	85.68	69.22	20.00	72.09	98.34	76.82

Appendix D.16: Initial Hold-out Test for Random Forest without Tie-Breaking Capability

		Prediction										Σ(Actual)
		N	B	A	F	S	R	E	D	W	G	
Actual	N	**28191**	0	403	7533	134	5	706	20	1	7	37000
	B	0	**52**	40	11	4	0	449	27	0	0	583
	A	3	129	**40**	8	0	0	472	25	0	0	677
	F	892	185	130	**3328**	223	6	1231	66	0	1	6062
	S	6	0	0	39	**267**	1	55	9	1	0	378
	R	7	164	58	28	28	**2802**	396	12	0	1	3496
	E	82	1170	506	212	75	168	**8773**	124	3	19	11132
	D	8	1250	477	98	44	29	1738	**438**	0	7	4089
	W	1	0	0	3	0	0	32	0	**7**	1	44
	G	3	7	0	36	18	3	436	38	3	**18327**	18871
Σ(Predicted)		29193	2957	1654	11296	793	3014	14288	759	15	18363	82332
												Weighted Average
Accuracy												75.58
Precision		96.57	1.76	2.42	29.46	33.67	92.97	61.40	57.71	46.67	99.80	83.77
Recall		76.19	8.92	5.91	54.90	70.63	80.15	78.81	10.71	15.91	97.12	75.58
F-measure		85.18	2.94	3.43	38.35	45.60	86.08	69.02	18.07	23.73	98.44	77.82

Appendix D.17: Initial Hold-out Test for Random Forest with Tie-Breaking Capability

		Prediction										Σ(Actual)
		N	B	A	F	S	R	E	D	W	G	
Actual	N	**28144**	0	414	7560	139	7	712	17	1	6	37000
	B	0	**63**	34	11	4	0	455	16	0	0	583
	A	2	139	**34**	8	0	0	479	15	0	0	677
	F	889	204	120	**3402**	206	8	1185	46	0	2	6062
	S	4	0	0	48	**267**	2	47	9	1	0	378
	R	9	164	60	28	27	**2814**	382	11	0	1	3496
	E	88	1184	508	196	70	165	**8797**	102	2	20	11132
	D	10	1248	487	85	45	34	1742	**431**	0	7	4089
	W	0	0	0	3	0	0	32	0	**7**	2	44
	G	5	6	0	41	18	3	436	36	3	**18323**	18871
Σ(Predicted)		29151	3008	1657	11382	776	3033	14267	683	14	18361	82332
												Weighted Average
Accuracy												75.65
Precision		96.55	2.09	2.05	29.89	34.41	92.78	61.66	63.10	50.00	99.79	84.09
Recall		76.06	10.81	5.02	56.12	70.63	80.49	79.02	10.54	15.91	97.10	75.65
F-measure		85.09	3.51	2.91	39.00	46.27	86.20	69.27	18.06	24.14	98.43	77.87

Appendix D.18: Initial Hold-out Test for Support Vector Machine

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**22970**	0	157	11215	0	1348	1306	1	0	3	37000
	B	2	**0**	0	17	0	25	539	0	0	0	583
	A	58	0	**0**	0	0	0	619	0	0	0	677
	F	212	0	0	**4270**	0	165	1407	0	0	8	6062
	S	1	0	0	142	**0**	234	1	0	0	0	378
	R	18	0	0	421	0	**2249**	796	0	0	12	3496
	E	615	2	16	1036	0	304	**9152**	3	0	4	11132
	D	72	2	0	238	0	113	3627	**13**	0	24	4089
	W	0	0	0	9	0	1	34	0	**0**	0	44
	G	16	0	0	224	0	46	418	0	0	**18167**	18871
Σ(Predicted)		23964	4	173	17572	0	4485	17899	17	0	18218	82332
												Weighted Average
Accuracy												69.01
Precision		95.85	0.00	0.00	24.30	NaN	50.14	51.13	76.47	NaN	99.72	NaN
Recall		62.08	0.00	0.00	70.44	0.00	64.33	82.21	0.32	0.00	96.27	69.01
F-measure		75.36	0.00	0.00	36.13	NaN	56.36	63.05	0.63	NaN	97.96	NaN

Appendix D.19: Final Hold-out Test for Naïve Bayes

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**19478**	75	560	11758	2	2410	2617	38	0	62	37000
	B	5	**0**	0	45	0	0	182	311	0	40	583
	A	0	0	**0**	58	0	0	259	320	0	40	677
	F	131	0	6	**3346**	0	736	1091	682	0	70	6062
	S	1	0	0	209	**0**	165	3	0	0	0	378
	R	26	0	8	1458	0	**1204**	697	93	0	10	3496
	E	157	0	54	2545	3	112	**6861**	1269	0	131	11132
	D	117	0	6	373	5	95	2351	**1030**	0	112	4089
	W	0	0	0	4	0	6	34	0	**0**	0	44
	G	5285	0	1	249	7	109	320	18	0	**12882**	18871
Σ(Predicted)		25200	75	635	20045	17	4837	14415	3761	0	13347	82332
												Weighted Average
Accuracy												54.42
Precision		77.29	0.00	0.00	16.69	0.00	24.89	47.60	27.39	NaN	96.52	NaN
Recall		52.64	0.00	0.00	55.20	0.00	34.44	61.63	25.19	0.00	68.26	54.42
F-measure		62.63	0.00	0.00	25.63	0.00	28.90	53.71	26.24	NaN	79.97	NaN

Appendix D.20: Final Hold-out Test for *K*-Nearest Neighbour

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**28171**	13	644	6864	117	95	921	127	6	42	37000
	B	78	**30**	0	327	6	8	122	12	0	0	583
	A	66	1	**0**	321	1	5	265	16	0	2	677
	F	1575	58	0	**3030**	241	152	868	115	3	20	6062
	S	12	4	0	27	**289**	12	28	4	0	2	378
	R	35	58	5	39	32	**2786**	521	18	0	2	3496
	E	321	304	62	880	86	258	**8727**	412	6	76	11132
	D	139	285	14	436	26	43	2514	**616**	1	15	4089
	W	1	0	0	1	0	0	12	0	**30**	0	44
	G	31	4	0	402	16	20	263	53	5	**18077**	18871
Σ(Predicted)		30429	757	725	12327	814	3379	14241	1373	51	18236	82332
												Weighted Average
Accuracy												75.01
Precision		92.58	3.96	0.00	24.58	35.50	82.45	61.28	44.87	58.82	99.13	80.37
Recall		76.14	5.15	0.00	49.98	76.46	79.69	78.40	15.06	68.18	95.79	75.01
F-measure		83.56	4.48	0.00	32.95	48.49	81.05	68.79	22.56	63.16	97.43	76.46

Appendix D.21: Final Hold-out Test for J48 Decision Tree

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**28043**	6	583	7375	123	12	750	96	1	11	37000
	B	15	**96**	0	4	4	2	455	1	0	6	583
	A	18	88	**0**	0	0	0	569	2	0	0	677
	F	1319	199	0	**2973**	155	22	1328	50	0	16	6062
	S	3	4	0	28	**307**	9	25	1	0	1	378
	R	10	86	1	7	22	**2847**	512	10	0	1	3496
	E	157	732	9	144	71	198	**9581**	176	3	61	11132
	D	43	702	4	58	28	29	2672	**506**	1	46	4089
	W	0	1	0	0	0	0	11	0	**31**	1	44
	G	19	4	0	25	15	3	302	56	5	**18442**	18871
Σ(Predicted)		29627	1918	597	10614	725	3122	16205	898	41	18585	82332
												Weighted Average
Accuracy												76.31
Precision		94.65	5.01	0.00	28.01	42.34	91.19	59.12	56.35	75.61	99.23	82.28
Recall		75.79	16.47	0.00	49.04	81.22	81.44	86.07	12.37	70.45	97.73	76.31
F-measure		84.18	7.68	0.00	35.66	55.67	86.04	70.10	20.29	72.94	98.47	77.51

Appendix D.22: Final Hold-out Test for Random Forest without Tie-Breaking Capability

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**28131**	0	464	7578	108	9	667	36	1	6	37000
	B	0	**80**	41	96	7	0	353	6	0	0	583
	A	3	45	**41**	87	3	0	494	4	0	0	677
	F	1052	103	98	**3350**	252	5	1160	32	2	8	6062
	S	4	0	0	21	**307**	2	33	10	0	1	378
	R	11	21	23	26	48	**2804**	542	17	0	4	3496
	E	77	220	246	359	90	182	**9780**	157	3	18	11132
	D	15	175	198	184	36	24	2874	**572**	2	9	4089
	W	0	0	0	0	0	0	12	0	**31**	1	44
	G	10	3	0	30	20	3	352	36	3	**18414**	18871
Σ(Predicted)		29303	647	1111	11731	871	3029	16267	870	42	18461	82332
												Weighted Average
Accuracy												77.14
Precision		96.00	12.36	3.69	28.56	35.25	92.57	60.12	65.75	73.81	99.75	83.75
Recall		76.03	13.72	6.06	55.26	81.22	80.21	87.85	13.99	70.45	97.58	77.14
F-measure		84.86	13.01	4.59	37.66	49.16	85.95	71.39	23.07	72.09	98.65	78.36

Appendix D.23: Final Hold-out Test for Random Forest with Tie-Breaking Capability

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**28187**	0	447	7550	112	9	652	36	1	6	37000
	B	1	**37**	116	106	7	0	310	6	0	0	583
	A	0	3	**116**	96	3	0	455	4	0	0	677
	F	1052	27	241	**3359**	258	6	1071	34	2	12	6062
	S	5	0	0	19	**308**	1	35	9	0	1	378
	R	13	30	13	28	43	**2813**	533	19	0	4	3496
	E	81	201	345	378	87	184	**9679**	155	4	18	11132
	D	31	203	207	190	33	23	2833	**562**	1	6	4089
	W	0	0	0	0	0	0	13	0	**30**	1	44
	G	12	2	0	32	20	3	358	35	3	**18406**	18871
Σ(Predicted)		29382	503	1485	11758	871	3039	15939	860	41	18454	82332
												Weighted Average
Accuracy												77.12
Precision		95.93	7.36	7.81	28.57	35.36	92.56	60.73	65.35	73.17	99.74	83.78
Recall		76.18	6.35	17.13	55.41	81.48	80.46	86.95	13.74	68.18	97.54	77.12
F-measure		84.92	6.81	10.73	37.70	49.32	86.09	71.51	22.71	70.59	98.63	78.40

Appendix D.24: Final Hold-out Test for Support Vector Machine

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**22883**	0	154	11240	0	1333	1389	0	0	1	37000
	B	2	**0**	0	17	0	26	538	0	0	0	583
	A	58	0	**0**	0	0	0	619	0	0	0	677
	F	214	0	0	**4277**	0	161	1402	0	0	8	6062
	S	1	0	0	146	**0**	231	0	0	0	0	378
	R	17	0	0	426	0	**2246**	795	0	0	12	3496
	E	432	2	16	1046	0	312	**9314**	1	0	9	11132
	D	64	0	0	259	0	119	3623	**0**	0	24	4089
	W	0	0	0	9	0	1	34	0	**0**	0	44
	G	17	0	0	226	0	45	421	0	0	**18162**	18871
Σ(Predicted)		23688	2	170	17646	0	4474	18135	1	0	18216	82332
												Weighted Average
Accuracy												69.09
Precision		96.60	0.00	0.00	24.24	NaN	50.20	51.36	0.00	NaN	99.70	NaN
Recall		61.85	0.00	0.00	70.55	0.00	64.24	83.67	0.00	0.00	96.24	69.09
F-measure		75.41	0.00	0.00	36.08	NaN	56.36	63.65	0.00	NaN	97.94	NaN

Appendix D.25: Ensemble Learning of RF + RF-BT

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**28166**	0	438	7572	111	9	667	30	1	6	37000
	B	1	**38**	108	115	7	0	308	6	0	0	583
	A	0	3	**108**	106	3	0	453	4	0	0	677
	F	1008	27	225	**3440**	255	5	1058	32	2	10	6062
	S	4	0	0	20	**307**	1	36	9	0	1	378
	R	13	30	13	27	44	**2812**	536	18	0	3	3496
	E	77	201	328	391	80	181	**9697**	154	4	19	11132
	D	31	203	199	199	32	25	2834	**558**	1	7	4089
	W	0	0	0	0	0	0	13	0	**30**	1	44
	G	11	2	0	30	19	3	365	36	3	**18402**	18871
Σ(Predicted)		29311	504	1419	11900	858	3036	15967	847	41	18449	82332
												Weighted Average
Accuracy												77.20
Precision		96.09	7.54	7.61	28.91	35.78	92.62	60.73	65.88	73.17	99.75	83.91
Recall		76.12	6.52	15.95	56.75	81.22	80.43	87.11	13.65	68.18	97.51	77.20
F-measure		84.95	6.99	10.31	38.30	49.68	86.10	71.57	22.61	70.59	98.62	78.46

Appendix D.26: Ensemble Learning of RF + J48

		Prediction										
		N	B	A	F	S	R	E	D	W	G	Σ(Actual)
Actual	N	**28005**	4	504	7527	135	12	728	74	1	10	37000
	B	15	**95**	0	4	6	2	453	2	0	6	583
	A	18	87	**0**	1	1	0	568	2	0	0	677
	F	1270	194	12	**2998**	164	22	1338	48	0	16	6062
	S	3	2	0	21	**317**	9	24	1	0	1	378
	R	9	75	15	13	31	**2847**	496	7	0	3	3496
	E	136	682	114	162	84	201	**9553**	144	2	54	11132
	D	35	655	112	79	31	26	2614	**498**	1	38	4089
	W	0	0	0	0	0	0	11	0	**32**	1	44
	G	11	4	0	37	15	3	312	58	5	**18426**	18871
Σ(Predicted)		29502	1798	757	10842	784	3122	16097	834	41	18555	82332
												Weighted Average
Accuracy												76.24
Precision		94.93	5.28	0.00	27.65	40.43	91.19	59.35	59.71	78.05	99.30	82.58
Recall		75.69	16.30	0.00	49.46	83.86	81.44	85.82	12.18	72.73	97.64	76.24
F-measure		84.22	7.98	0.00	35.47	54.56	86.04	70.17	20.23	75.29	98.47	77.52

APPENDIX E: Confusion Matrices for ISCX-IDS2012 Dataset

N: Normal	D: DDoS	B: Brute Force	I: Infiltration	H: HTTP DoS

Appendix E.1: Initial Cross Validation for Naïve Bayes

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13782**	66	30	63	59	14000
	D	193	**25958**	56	13	2	26222
	B	3	0	**5117**	1	0	5121
	I	225	0	100	**6775**	21	7121
	H	10	0	2	16	**2602**	2630
Σ(Predicted)		14213	26024	5305	6868	2684	55094
							Weighted Average
Accuracy							98.44
Precision		96.97	99.75	96.46	98.65	96.94	98.46
Recall		98.44	98.99	99.92	95.14	98.94	98.44
F-measure		97.70	99.37	98.16	96.86	97.93	98.44

Appendix E.2: Initial Cross Validation for *K*-Nearest Neighbour

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13897**	59	0	42	2	14000
	D	18	**26201**	0	1	2	26222
	B	0	2	**5119**	0	0	5121
	I	29	0	2	**7089**	1	7121
	H	4	2	0	1	**2623**	2630
Σ(Predicted)		13948	26264	5121	7133	2628	55094
							Weighted Average
Accuracy							99.70
Precision		99.63	99.76	99.96	99.38	99.81	99.70
Recall		99.26	99.92	99.96	99.55	99.73	99.70
F-measure		99.45	99.84	99.96	99.47	99.77	99.70

Appendix E.3: Initial Cross Validation for J48 Decision Tree

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13970**	13	3	10	4	14000
	D	20	**26199**	3	0	0	26222
	B	0	0	**5120**	1	0	5121
	I	84	3	2	**7001**	31	7121
	H	28	0	2	13	**2587**	2630
Σ(Predicted)		14102	26215	5130	7025	2622	55094
							Weighted Average
Accuracy							99.61
Precision		99.06	99.94	99.81	99.66	98.67	99.61
Recall		99.79	99.91	99.98	98.31	98.37	99.61
F-measure		99.42	99.93	99.89	98.98	98.51	99.61

Appendix E.4: Initial Cross Validation for Random Forest without Tie-Breaking Capability

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13076**	886	19	16	3	14000
	D	12	**26196**	11	2	1	26222
	B	2	0	**5119**	0	0	5121
	I	38	40	0	**7043**	0	7121
	H	28	49	2	24	**2527**	2630
Σ(Predicted)		13156	27171	5151	7085	2531	55094
							Weighted Average
Accuracy							97.94
Precision		99.39	96.41	99.38	99.41	99.84	98.00
Recall		93.40	99.90	99.96	98.90	96.08	97.94
F-measure		96.30	98.13	99.67	99.16	97.93	97.93

Appendix E.5: Initial Cross Validation for Random Forest with Tie-Breaking Capability

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13525**	463	10	2	0	14000
	D	2	**26218**	0	1	1	26222
	B	0	6	**5115**	0	0	5121
	I	41	16	1	**7062**	1	7121
	H	12	35	0	11	**2572**	2630
Σ(Predicted)		13580	26738	5126	7076	2574	55094
							Weighted Average
Accuracy							98.91
Precision		99.59	98.06	99.79	99.80	99.92	98.92
Recall		96.61	99.98	99.88	99.17	97.79	98.91
F-measure		98.08	99.01	99.83	99.49	98.85	98.90

Appendix E.6: Initial Cross Validation for Support Vector Machine

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13739**	199	12	30	20	14000
	D	7	**26203**	2	0	7	26219
	B	2	0	**5119**	0	0	5121
	I	47	0	0	**7070**	5	7122
	H	7	0	7	0	**2618**	2632
Σ(Predicted)		13802	26402	5140	7100	2650	55094
							Weighted Average
Accuracy							99.37
Precision		99.54	99.25	99.59	99.58	98.79	99.38
Recall		98.14	99.94	99.96	99.27	99.47	99.37
F-measure		98.83	99.59	99.78	99.42	99.13	99.37

Appendix E.7: Final Cross Validation for Naïve Bayes

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13718**	137	18	38	89	14000
	D	0	**26146**	56	10	10	26222
	B	0	1	**5119**	1	0	5121
	I	110	3	2	**6951**	55	7121
	H	7	0	2	10	**2611**	2630
Σ(Predicted)		13835	26287	5197	7010	2765	55094
							Weighted Average
Accuracy							99.00
Precision		99.15	99.46	98.50	99.16	94.43	99.02
Recall		97.99	99.71	99.96	97.61	99.28	99.00
F-measure		98.57	99.59	99.22	98.38	96.79	99.00

Appendix E.8: Final Cross Validation for *K*-Nearest Neighbour

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13941**	15	0	39	5	14000
	D	10	**26209**	1	0	2	26222
	B	0	1	**5120**	0	0	5121
	I	41	0	2	**7077**	1	7121
	H	5	2	0	0	**2623**	2630
Σ(Predicted)		13997	26227	5123	7116	2631	55094
							Weighted Average
Accuracy							99.77
Precision		99.60	99.93	99.94	99.45	99.70	99.77
Recall		99.58	99.95	99.98	99.38	99.73	99.77
F-measure		99.59	99.94	99.96	99.42	99.71	99.77

Appendix E.9: Final Cross Validation for J48 Decision Tree

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13975**	8	3	9	5	14000
	D	18	**26196**	3	3	2	26222
	B	0	0	**5120**	1	0	5121
	I	70	0	2	**7016**	33	7121
	H	6	0	2	14	**2608**	2630
Σ(Predicted)		14069	26204	5130	7043	2648	55094
							Weighted Average
Accuracy							99.68
Precision		99.33	99.97	99.81	99.62	98.49	99.68
Recall		99.82	99.90	99.98	98.53	99.16	99.68
F-measure		99.58	99.94	99.89	99.07	98.83	99.67

Appendix E.10: Final Cross Validation for Random Forest without Tie-Breaking Capability

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13948**	5	24	19	4	14000
	D	15	**26202**	3	1	1	26222
	B	0	0	**5121**	0	0	5121
	I	42	0	2	**7039**	38	7121
	H	7	1	2	5	**2615**	2630
Σ(Predicted)		14012	26208	5152	7064	2658	55094
							Weighted Average
Accuracy							99.69
Precision		99.54	99.98	99.40	99.65	98.38	99.69
Recall		99.63	99.92	100.00	98.85	99.43	99.69
F-measure		99.59	99.95	99.70	99.25	98.90	99.69

Appendix E.11: Final Cross Validation for Random Forest with Tie-Breaking Capability

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13961**	3	12	21	3	14000
	D	11	**26208**	1	1	1	26222
	B	0	0	**5121**	0	0	5121
	I	40	0	2	**7042**	37	7121
	H	7	1	1	5	**2616**	2630
Σ(Predicted)		14019	26212	5137	7069	2657	55094
							Weighted Average
Accuracy							99.73
Precision		99.59	99.98	99.69	99.62	98.46	99.74
Recall		99.72	99.95	100.00	98.89	99.47	99.73
F-measure		99.65	99.97	99.84	99.25	98.96	99.73

Appendix E.12: Final Cross Validation for Support Vector Machine

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**13885**	78	0	6	31	14000
	D	16	**26204**	0	0	2	26222
	B	1	0	**5120**	0	0	5121
	I	51	0	0	**7069**	1	7121
	H	10	0	1	0	**2619**	2630
Σ(Predicted)		13963	26282	5121	7075	2653	55094
							Weighted Average
Accuracy							99.64
Precision		99.44	99.70	99.98	99.92	98.72	99.64
Recall		99.18	99.93	99.98	99.27	99.58	99.64
F-measure		99.31	99.82	99.98	99.59	99.15	99.64

Appendix E.13: Initial Hold-out Test for Naïve Bayes

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5920**	27	8	19	26	6000
	D	99	**11110**	26	0	3	11238
	B	3	0	**2188**	0	4	2195
	I	108	0	62	**2868**	14	3052
	H	3	0	3	1	**1121**	1128
Σ(Predicted)		6133	11137	2287	2888	1168	23613
							Weighted Average
Accuracy							98.28
Precision		96.53	99.76	95.67	99.31	95.98	98.32
Recall		98.67	98.86	99.68	93.97	99.38	98.28
F-measure		97.59	99.31	97.63	96.57	97.65	98.28

Appendix E.14: Initial Hold-out Test for *K*-Nearest Neighbour

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5959**	27	0	14	0	6000
	D	12	**11223**	0	2	1	11238
	B	0	1	**2192**	0	2	2195
	I	18	0	0	**3033**	1	3052
	H	3	2	1	0	**1122**	1128
Σ(Predicted)		5992	11253	2193	3049	1126	23613
							Weighted Average
Accuracy							99.64
Precision		99.45	99.73	99.95	99.48	99.64	99.64
Recall		99.32	99.87	99.86	99.38	99.47	99.64
F-measure		99.38	99.80	99.91	99.43	99.56	99.64

Appendix E.15: Initial Hold-out Test for J48 Decision Tree

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5995**	3	0	1	1	6000
	D	14	**11223**	0	1	0	11238
	B	0	0	**2193**	2	0	2195
	I	42	1	1	**2987**	21	3052
	H	10	2	2	0	**1114**	1128
Σ(Predicted)		6061	11229	2196	2991	1136	23613
							Weighted Average
Accuracy							99.57
Precision		98.91	99.95	99.86	99.87	98.06	99.58
Recall		99.92	99.87	99.91	97.87	98.76	99.57
F-measure		99.41	99.91	99.89	98.86	98.41	99.57

Appendix E.16: Initial Hold-out Test for Random Forest without Tie-Breaking Capability

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5598**	385	14	3	0	6000
	D	9	**11226**	2	1	0	11238
	B	8	13	**2173**	1	0	2195
	I	20	18	0	**3014**	0	3052
	H	2	26	0	0	**1100**	1128
Σ(Predicted)		5637	11668	2189	3019	1100	23613
							Weighted Average
Accuracy							97.87
Precision		99.31	96.21	99.27	99.83	100.00	97.93
Recall		93.30	99.89	99.00	98.75	97.52	97.87
F-measure		96.21	98.02	99.13	99.29	98.74	97.86

Appendix E.17: Initial Hold-out Test for Random Forest with Tie-Breaking Capability

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5781**	207	12	0	0	6000
	D	3	**11235**	0	0	0	11238
	B	1	6	**2188**	0	0	2195
	I	13	7	0	**3032**	0	3052
	H	34	30	0	0	**1064**	1128
Σ(Predicted)		5832	11485	2200	3032	1064	23613
							Weighted Average
Accuracy							98.67
Precision		99.13	97.82	99.45	100.00	100.00	98.69
Recall		96.35	99.97	99.68	99.34	94.33	98.67
F-measure		97.72	98.89	99.57	99.67	97.08	98.67

Appendix E.18: Initial Hold-out Test for Support Vector Machine

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5905**	65	1	18	11	6000
	D	10	**11197**	27	0	4	11238
	B	0	0	**2195**	0	0	2195
	I	97	7	0	**2882**	66	3052
	H	17	5	3	0	**1103**	1128
Σ(Predicted)		6029	11274	2226	2900	1184	23613
							Weighted Average
Accuracy							98.60
Precision		97.94	99.32	98.61	99.38	93.16	98.62
Recall		98.42	99.64	100.00	94.43	97.78	98.60
F-measure		98.18	99.48	99.30	96.84	95.42	98.60

Appendix E.19: Final Hold-out Test for Naïve Bayes

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5906**	53	3	11	27	6000
	D	2	**11206**	27	0	3	11238
	B	0	1	**2191**	1	2	2195
	I	52	0	0	**2971**	29	3052
	H	3	0	3	1	**1121**	1128
Σ(Predicted)		5963	11260	2224	2984	1182	23613
							Weighted Average
Accuracy							99.08
Precision		99.04	99.52	98.52	99.56	94.84	99.09
Recall		98.43	99.72	99.82	97.35	99.38	99.08
F-measure		98.74	99.62	99.16	98.44	97.06	99.08

Appendix E.20: Final Hold-out Test for *K*-Nearest Neighbour

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5979**	7	0	14	0	6000
	D	6	**11228**	0	1	3	11238
	B	0	2	**2193**	0	0	2195
	I	23	0	0	**3027**	2	3052
	H	2	3	1	0	**1122**	1128
Σ(Predicted)		6010	11240	2194	3042	1127	23613
							Weighted Average
Accuracy							99.73
Precision		99.48	99.89	99.95	99.51	99.56	99.73
Recall		99.65	99.91	99.91	99.18	99.47	99.73
F-measure		99.57	99.90	99.93	99.34	99.51	99.73

Appendix E.21: Final Hold-out Test for J48 Decision Tree

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5994**	4	0	1	1	6000
	D	8	**11224**	0	3	3	11238
	B	0	0	**2193**	2	0	2195
	I	41	0	0	**2988**	23	3052
	H	3	0	2	0	**1123**	1128
Σ(Predicted)		6046	11228	2195	2994	1150	23613
							Weighted Average
Accuracy							99.62
Precision		99.14	99.96	99.91	99.80	97.65	99.62
Recall		99.90	99.88	99.91	97.90	99.56	99.62
F-measure		99.52	99.92	99.91	98.84	98.60	99.62

Appendix E.22: Final Hold-out Test for Random Forest without Tie-Breaking Capability

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5979**	1	16	4	0	6000
	D	12	**11223**	0	2	1	11238
	B	0	0	**2195**	0	0	2195
	I	27	0	0	**3003**	22	3052
	H	3	1	2	1	**1121**	1128
Σ(Predicted)		6021	11225	2213	3010	1144	23613
							Weighted Average
Accuracy							99.61
Precision		99.30	99.98	99.19	99.77	97.99	99.61
Recall		99.65	99.87	100.00	98.39	99.38	99.61
F-measure		99.48	99.92	99.59	99.08	98.68	99.61

Appendix E.23: Final Hold-out Test for Random Forest with Tie-Breaking Capability

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5979**	3	16	2	0	6000
	D	12	**11223**	0	2	1	11238
	B	0	0	**2195**	0	0	2195
	I	28	0	0	**3002**	22	3052
	H	3	1	2	1	**1121**	1128
Σ(Predicted)		6022	11227	2213	3007	1144	23613
							Weighted Average
Accuracy							99.60
Precision		99.29	99.96	99.19	99.83	97.99	99.61
Recall		99.65	99.87	100.00	98.36	99.38	99.60
F-measure		99.47	99.92	99.59	99.09	98.68	99.61

Appendix E.24: Final Hold-out Test for Support Vector Machine

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5957**	31	4	3	5	6000
	D	11	**11223**	0	1	3	11238
	B	0	0	**2195**	0	0	2195
	I	37	0	0	**3013**	2	3052
	H	5	0	3	3	**1117**	1128
Σ(Predicted)		6010	11254	2202	3020	1127	23613
							Weighted Average
Accuracy							99.54
Precision		99.12	99.72	99.68	99.77	99.11	99.54
Recall		99.28	99.87	100.00	98.72	99.02	99.54
F-measure		99.20	99.80	99.84	99.24	99.07	99.54

Appendix E.25: Ensemble Learning of KNN + J48

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5979**	7	0	14	0	6000
	D	5	**11229**	0	1	3	11238
	B	0	0	**2194**	1	0	2195
	I	23	0	0	**3027**	2	3052
	H	3	0	2	0	**1123**	1128
Σ(Predicted)		6010	11236	2196	3043	1128	23613
							Weighted Average
Accuracy							99.74
Precision		99.48	99.94	99.91	99.47	99.56	99.74
Recall		99.65	99.92	99.95	99.18	99.56	99.74
F-measure		99.57	99.93	99.93	99.33	99.56	99.74

Appendix E.26: Ensemble Learning of KNN + RF

		Prediction					
		N	D	B	I	H	Σ(Actual)
Actual	N	**5973**	6	0	20	1	6000
	D	3	**11231**	0	1	3	11238
	B	0	1	**2194**	0	0	2195
	I	26	0	0	**3024**	2	3052
	H	3	2	1	0	**1122**	1128
Σ(Predicted)		6005	11240	2195	3045	1128	23613
							Weighted Average
Accuracy							99.71
Precision		99.47	99.92	99.95	99.31	99.47	99.71
Recall		99.55	99.94	99.95	99.08	99.47	99.71
F-measure		99.51	99.93	99.95	99.20	99.47	99.71

APPENDIX F: Confusion Matrices for NSL-KDD Dataset

N: Normal	A: Anomaly

Appendix F.1: Initial Cross Validation for Naïve Bayes

		Prediction		Σ(Actual)
		N	A	
Actual	N	**63060**	4283	67343
	A	7832	**50798**	58630
Σ(Predicted)		70892	55081	125973
				Weighted Average
Accuracy				90.38
Precision		88.95	92.22	90.48
Recall		93.64	86.64	90.38
F-measure		91.24	89.35	90.36

Appendix F.2: Initial Cross Validation for *K*-Nearest Neighbour

		Prediction		Σ(Actual)
		N	A	
Actual	N	**67127**	216	67343
	A	218	**58412**	58630
Σ(Predicted)		67345	58628	125973
				Weighted Average
Accuracy				99.66
Precision		99.68	99.63	99.66
Recall		99.68	99.63	99.66
F-measure		99.68	99.63	99.66

Appendix F.3: Initial Cross Validation for J48 Decision Tree

		Prediction		Σ(Actual)
		N	A	
Actual	N	**67200**	143	67343
	A	132	**58498**	58630
Σ(Predicted)		67332	58641	125973
				Weighted Average
Accuracy				99.78
Precision		99.80	99.76	99.78
Recall		99.79	99.77	99.78
F-measure		99.80	99.77	99.78

Appendix F.4: Initial Cross Validation for Random Forest without Tie-Breaking Capability

		Prediction		Σ(Actual)
		N	A	
Actual	N	**67319**	24	67343
	A	80	**58550**	58630
Σ(Predicted)		67399	58574	125973
				Weighted Average
Accuracy				99.92
Precision		99.88	99.96	99.92
Recall		99.96	99.86	99.92
F-measure		99.92	99.91	99.92

Appendix F.5: Initial Cross Validation for Random Forest with Tie-Breaking Capability

		Prediction		Σ(Actual)
		N	A	
Actual	N	**67319**	24	67343
	A	81	**58549**	58630
Σ(Predicted)		67400	58573	125973
				Weighted Average
Accuracy				99.92
Precision		99.88	99.96	99.92
Recall		99.96	99.86	99.92
F-measure		99.92	99.91	99.92

Appendix F.6: Initial Cross Validation for Support Vector Machine

		Prediction		Σ(Actual)
		N	A	
Actual	N	**65790**	1553	67343
	A	3545	**55085**	58630
Σ(Predicted)		69335	56638	125973
				Weighted Average
Accuracy				95.95
Precision		94.89	97.26	95.99
Recall		97.69	93.95	95.95
F-measure		96.27	95.58	95.95

Appendix F.7: Final Cross Validation for Naïve Bayes

		Prediction		Σ(Actual)
		N	A	
Actual	N	**65032**	2311	67343
	A	2446	**56184**	58630
Σ(Predicted)		67478	58495	125973
				Weighted Average
Accuracy				96.22
Precision		96.38	96.05	96.22
Recall		96.57	95.83	96.22
F-measure		96.47	95.94	96.22

Appendix F.8: Final Cross Validation for *K*-Nearest Neighbour

		Prediction		Σ(Actual)
		N	A	
Actual	N	**67189**	154	67343
	A	218	**58412**	58630
Σ(Predicted)		67407	58566	125973
				Weighted Average
Accuracy				99.70
Precision		99.68	99.74	99.70
Recall		99.77	99.63	99.70
F-measure		99.72	99.68	99.70

Appendix F.9: Final Cross Validation for J48 Decision Tree

		Prediction		Σ(Actual)
		N	A	
Actual	N	**67253**	90	67343
	A	93	**58537**	58630
Σ(Predicted)		67346	58627	125973
				Weighted Average
Accuracy				99.85
Precision		99.86	99.85	99.85
Recall		99.87	99.84	99.85
F-measure		99.86	99.84	99.85

Appendix F.10: Final Cross Validation for Random Forest without Tie-Breaking Capability

		Prediction		Σ(Actual)
		N	A	
Actual	N	**67314**	29	67343
	A	67	**58563**	58630
Σ(Predicted)		67381	58592	125973
				Weighted Average
Accuracy				99.92
Precision		99.90	99.95	99.92
Recall		99.96	99.89	99.92
F-measure		99.93	99.92	99.92

Appendix F.11: Final Cross Validation for Random Forest with Tie-Breaking Capability

		Prediction		Σ(Actual)
		N	A	
Actual	N	**67315**	28	67343
	A	67	**58563**	58630
Σ(Predicted)		67382	58591	125973
				Weighted Average
Accuracy				99.92
Precision		99.90	99.95	99.92
Recall		99.96	99.89	99.92
F-measure		99.93	99.92	99.92

Appendix F.12: Final Cross Validation for Support Vector Machine

		Prediction		Σ(Actual)
		N	A	
Actual	N	**66042**	1301	67343
	A	2254	**56376**	58630
Σ(Predicted)		68296	57677	125973
				Weighted Average
Accuracy				97.18
Precision		96.70	97.74	97.19
Recall		98.07	96.16	97.18
F-measure		97.38	96.94	97.18

Appendix F.13: Initial Hold-out Test for Naïve Bayes

		Prediction		Σ(Actual)
		N	A	
Actual	N	**9041**	670	9711
	A	4714	**8119**	12833
Σ(Predicted)		13755	8789	22544
				Weighted Average
Accuracy				76.12
Precision		65.73	92.38	80.90
Recall		93.10	63.27	76.12
F-measure		77.06	75.10	75.94

Appendix F.14: Initial Hold-out Test for *K*-Nearest Neighbour

		Prediction		Σ(Actual)
		N	A	
Actual	N	**9342**	369	9711
	A	4285	**8548**	12833
Σ(Predicted)		13627	8917	22544
				Weighted Average
Accuracy				79.36
Precision		68.56	95.86	84.10
Recall		96.20	66.61	79.36
F-measure		80.06	78.60	79.23

Appendix F.15: Initial Hold-out Test for J48 Decision Tree

		Prediction		Σ(Actual)
		N	A	
Actual	N	**9448**	263	9711
	A	3900	**8933**	12833
Σ(Predicted)		13348	9196	22544
				Weighted Average
Accuracy				81.53
Precision		70.78	97.14	85.79
Recall		97.29	69.61	81.53
F-measure		81.95	81.10	81.47

Appendix F.16: Initial Hold-out Test for Random Forest without Tie-Breaking Capability

		Prediction		Σ(Actual)
		N	A	
Actual	N	**9447**	264	9711
	A	4143	**8690**	12833
Σ(Predicted)		13590	8954	22544
				Weighted Average
Accuracy				80.45
Precision		69.51	97.05	85.19
Recall		97.28	67.72	80.45
F-measure		81.09	79.77	80.34

Appendix F.17: Initial Hold-out Test for Random Forest with Tie-Breaking Capability

		Prediction		Σ(Actual)
		N	A	
Actual	N	**9443**	268	9711
	A	4199	**8634**	12833
Σ(Predicted)		13642	8902	22544
				Weighted Average
Accuracy				80.19
Precision		69.22	96.99	85.03
Recall		97.24	67.28	80.19
F-measure		80.87	79.45	80.06

Appendix F.18: Initial Hold-out Test for Support Vector Machine

		Prediction		Σ(Actual)
		N	A	
Actual	N	**8979**	732	9711
	A	4815	**8018**	12833
Σ(Predicted)		13794	8750	22544
				Weighted Average
Accuracy				75.39
Precision		65.09	91.63	80.20
Recall		92.46	62.48	75.39
F-measure		76.40	74.30	75.20

Appendix F.19: Final Hold-out Test for Naïve Bayes

		Prediction		Σ(Actual)
		N	A	
Actual	N	**8751**	960	9711
	A	3911	**8922**	12833
Σ(Predicted)		12662	9882	22544
				Weighted Average
Accuracy				78.39
Precision		69.11	90.29	81.16
Recall		90.11	69.52	78.39
F-measure		78.23	78.56	78.41

Appendix F.20: Final Hold-out Test for *K*-Nearest Neighbour

		Prediction		Σ(Actual)
		N	A	
Actual	N	**8803**	908	9711
	A	3435	**9398**	12833
Σ(Predicted)		12238	10306	22544
				Weighted Average
Accuracy				80.74
Precision		71.93	91.19	82.89
Recall		90.65	73.23	80.74
F-measure		80.21	81.23	80.79

Appendix F.21: Final Hold-out Test for J48 Decision Tree

		Prediction		Σ(Actual)
		N	A	
Actual	N	**9329**	382	9711
	A	2930	**9903**	12833
Σ(Predicted)		12259	10285	22544
				Weighted Average
Accuracy				85.30
Precision		76.10	96.29	87.59
Recall		96.07	77.17	85.31
F-measure		84.92	85.67	85.35

Appendix F.22: Final Hold-out Test for Random Forest without Tie-Breaking Capability

		Prediction		Σ(Actual)
		N	A	
Actual	N	**9389**	322	9711
	A	3860	**8973**	12833
Σ(Predicted)		13249	9295	22544
				Weighted Average
Accuracy				81.45
Precision		70.87	96.54	85.48
Recall		96.68	69.92	81.45
F-measure		81.79	81.10	81.40

Appendix F.23: Final Hold-out Test for Random Forest with Tie-Breaking Capability

		Prediction		Σ(Actual)
		N	A	
Actual	N	**9387**	324	9711
	A	3863	**8970**	12833
Σ(Predicted)		13250	9294	22544
				Weighted Average
Accuracy				81.43
Precision		70.85	96.51	85.46
Recall		96.66	69.90	81.43
F-measure		81.76	81.08	81.37

Appendix F.24: Final Hold-out Test for Support Vector Machine

		Prediction		Σ(Actual)
		N	A	
Actual	N	**8976**	735	9711
	A	4694	**8139**	12833
Σ(Predicted)		13670	8874	22544
				Weighted Average
Accuracy				75.92
Precision		65.66	91.72	80.49
Recall		92.43	63.42	75.92
F-measure		76.78	74.99	75.76

Appendix F.25: Ensemble Learning of J48 + RF

		Prediction		Σ(Actual)
		N	A	
Actual	N	**9333**	378	9711
	A	2933	**9900**	12833
Σ(Predicted)		12266	10278	22544
				Weighted Average
Accuracy				85.31
Precision		76.09	96.32	87.61
Recall		96.11	77.14	85.31
F-measure		84.93	85.67	85.36

Appendix F.26: Ensemble Learning of J48 + RF-BT

		Prediction		Σ(Actual)
		N	A	
Actual	N	**8777**	934	9711
	A	3148	**9685**	12833
Σ(Predicted)		11925	10619	22544
				Weighted Average
Accuracy				81.89
Precision		73.60	91.20	83.62
Recall		90.38	75.47	81.89
F-measure		81.13	82.59	81.96