A Comparison Of Machine Learning Algorithms to Detect Network Intrusions

Jack Anderson -
40208539

Submitted in partial fulfilment of
the requirements of Edinburgh Napier University
for the Degree of
BEng (Hons) Software Engineering

School of Computing

16th March 2018

Authorship Declaration

I, Jack Anderson, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained informed consent from all people I have involved in the work in this dissertation following the School's ethical guidelines.

Signed:

Date:

Matriculation no:

Data Protection Declaration

Under the 1998 Data Protection Act, The University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below one of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Abstract

Contents

List of Tables

List of Figures

Acknowledgements

Insert acknowledgements here

1 Introduction

1.1 Background

For any major company or business security is of the utmost concern, with a study finding that in the UK in 2016 an estimated 46% of all businesses experienced a cyber security breach or attack K, Jayesh N S, Tom R and Mark B, 2017. These breaches are particularly dangerous as even if a network is compromised a single time a company can have its entire database destroyed, or customer data leaked, leading to legal repercussions. When it comes to preventing these intrusions firewalls alone are insufficient for anything but the most rudimentary of attacks and so a Network Intrusion Detection System (NIDS) is employed to further bolster the security of the network. Traditional NIDS are placed strategically on a network to monitor all incoming traffic. It analyses the passing traffic and then compares it to a large library of known attacks and if it matches will flag the traffic. While these systems do provide some degree of protection they are unable to detect novel attacks or zero-day vulnerabilities and so some other method of identifying suspicious traffic is required. Introducing machine learning to a NIDS is one way of attempting to solve this problem first proposed by Denning, 1987. In using machine learning to detect network intrusions the system can be trained to recognise patterns of intrusive behaviour, allowing it to detect attacks which it may not have seen before but have characteristics of similar attacks. Machine learning also allows systems to be easily retrained to accommodate for new data on attacks as it emerges. There are two main categories of NIDS: misuse detection, and anomaly detection; both of which have their own advantages and disadvantages. This literary review aims to discuss the different kinds of network intrusion detection systems, the algorithms that these systems employ, and the gap in papers which directly compare the performance of single stage and two-stage classifiers in the domain of network intrusion detection.

1.2 Research Questions

For this honours thesis there are a number of research questions which have been collated, and an attempt made to answer them. These questions in which I am interested in answering are the following:

- What is the rate of accurate detection and classification of network intrusions by single stage machine learning classification methods?
- What is the rate of accurate detection and classification of network intrusions by two stage machine learning classification methods?

- Which method of classification i.e. single stage or two stage, is more accurate and by what amount?
- Which configuration of algorithms in the two stage classifier produces the most accurate results?

In this context, accuracy is defined as a high number of true positives and true negatives, and a low number of false positives and false negatives when classifying network intrusions.

First research will be completed in order to gain an understanding of the history and current state of machine learning for network intrusion detection, through reading relevant research papers and articles. Next the individual algorithms and methods which go into detecting network intrusions will be researched and understood. This knowledge will then be put into developing a piece of software capable of running these algorithms and extracting metrics which will be used to answer these research questions.

1.3 Aims and Objectives

The aim of this project is to create a piece of software which is capable of running both single and two stage classification algorithm arrangements on a number of datasets and displaying the results in a clear manner. In doing so this will assist in answering the research questions put forward.

These research questions, specifically whether or not two stage classification provides more accurate results than single stage classification will be answered by meeting the following objectives:

- Perform a review of existing literature.
- Perform a review of existing software.
- Select appropriate machine learning classification algorithms.
- Select appropriate datasets for use in network intrusion detection.
- Implementation of software to assist in comparison of classification algorithms.
- Evaluation of software with regards to specification and other software.
- Evaluation of classification algorithm performance.

1.4 Scope and Limitations

1.4.1 Deliverables

The list of deliverables for this project are the following:

- A review of literature on related topics and a report detailing the findings of such reports.
- A review of existing software which performs a similar role to the proposed software.
- A software requirement specification.
- A description of all testing which will be carried out.
- A report detailing the results of each classification algorithm and configuration of two stage classification algorithms
- An evaluation of the implemented software with regards to how well it meets the proposed specification, and how it compares to existing software.

1.4.2 Boundaries

There exists a large number of different classification algorithms and methods to perform network intrusion detection, therefore this project must focus on a few specifically. This project will focus on performing misuse detection rather than anomaly detection, and will limit the classification algorithms used to: k-Nearest Neighbour, Multi-layer Perceptron, and a Support Vector Machine.

1.4.3 Constraints

The largest constraint facing this project is that of time. There is a large amount of literature to be reviewed, and also a considerable amount of results which must be collected and results drawn from in a short amount of time. Time Knowledge Misuse detection only

1.5 Structure of this Dissertation

This dissertation can be divided into three main parts. Part one is literature review and past projects within the field, and existing software. Second part details the planning design and implementation of the software and the third consists of an analysis and evaluation of results and software.

part one reviews the literature on the subject and examines results of past projects and individual algorithms, also reviews existing software for machine learning comparison.

part two describes the detailed software requirements and specific functionality which should be implemented. Also includes a description of technologies and libraries used, and a testing plan.

part three is an evaluation of the final software product against its design. Analysis and evaluation of classifier results. also has an evaluation of the overall project how well it met its aims and suggestions for future work.

2 Literature Review

2.1 Search Strategy

While searching for relevant papers on the subject of network intrusion detection, a number of key terms were identified which could be used to find papers within the field. The main search term which was used was *'Network Intrusion Detection'* which was used to find papers which were generally related to the topic. A number of suplemental search terms were identified and used in conjuntion with the main search term when attempting to find papers which used a specific technology within network intrusion detection. These included: 'Nearest Neighbor', 'k-NN', 'Neural', 'Self Organising Map', 'SOM', 'Recurrent', 'Hybrid', 'Stage', and 'Ensemble'. Using a combination of the main search term and these additions search terms allowed the discovery of papers directly relevant to the project. If two terms were required to be included within the search then the **AND** operator would be used to ensure that both were matched. Similarly if there were multiple terms for the same technique or technology, i.e. 'Nearest Neighbor' and 'k-NN' then the **OR** operator would be used, to reduce the number of individual searches which were required to be carried out. When papers were found and deemed relevant to the project, a review of citations included within those papers would also be carried out, which allows the finding of papers which are directly related to the subject but which may have been missed by the search terms specified. A seperate search was also performed when researching a relevant dataset by searching for the title of each dataset, such as: 'KDD Cup', 'NSL-KDD', 'DARPA', etc.

2.2 Databases

To find appropriate research papers, Google Scholar and the Edinburgh Napier University Library Search were used to locate articles and other databases which could also be searched. The databases from which the papers were retrieved were:

- ScienceDirect

- IEEE Xplore
- ACM Digital Library
- Society for Industrial and Applied Mathematics

2.3 Paper Selection

At first a large range of papers were collected by title and held on to that could be relevant to the subject matter. Then the abstract and conclusion were read through to determine whether or not the paper was relevant to the research questions. Once the pool of papers had been reduced to a manageable size the entire paper was read through to attempt to make links between the content of the paper and the work which would be carried out and how they could assist in answering the research questions.

During this process, papers would be selected for use if they met all of the following criteria:

- Peer reviewed.
- If the paper is older than five years and has at least 50 citations.
- The paper is directly relevant to the research questions.

Papers would be rejected if they met any of the following critera:

- Too broad.
- Cannot be directly applied to the research questions.
- The paper has less than fifty citations and is older than five years.
- A newer more relevant paper on topic was found.

2.4 Network Intrusion Detection

The field of network intrusion detection was conceived by Denning, 1987, a paper in which the author describes a model for a "real-time intrusion-detection expert system" capable of discerning between normal and abnormal network activity. NIDS can be broadly categorized as performing either: misuse detection or anomaly detection. Misuse detection systems are first trained using some kind of learning algorithm on a set of labelled data where each entry is defined as being either 'normal' or 'intrusive'. Using this the system can create a sophisticated model of attacks, with a very high accuracy in detecting previously observed attacks and variations of such attacks. However, these types of systems perform poorly when faced with novel attacks being unable to accurately detect and classify them.

Anomaly detections systems are trained on a set of data in which every entry is an example of 'normal' network traffic. Training the system in such a way means that *"behavior is flagged as a potential intrusion if it deviates significantly from expected behavior"* Javitz, Valdes, Breen and Patton, 1994. This allows such systems to easily detect novel attacks which have not been observed before. Although these systems perform well at detecting novel attacks they also have a much higher false positive rate than misuse detection systems. The reason for this is that *"previously unseen (yet legitimate) system behaviors are also recognized as anomalies, and hence flagged as potential intrusions"* Lazarevic, Ertoz, Kumar, Ozgur and Srivastava, 2003. Anomaly detection systems also suffer from a so called *"semantic gap"* where anomalies can be detected yet there is no further information on what type of attack has been performed Sommer and Paxson, 2010.

In the beginning of network intrusion detection research methods were proposed for an expert system NIDS Ilgun, Kemmerer and Porras, 1995, in which took knowledge from experts within the network security field and encoded them into rules with which the system could use to check traffic with to determine if it was intrusive. Then in W. Lee, Stolfo and Mok, 1999 a framework was proposed for the use of data mining for building NIDS. This was the first to employ machine learning in its approach to detecting intrusions, a method which today has been explored extensively. In using this approach to network intrusion detection the NIDS can detect attacks which have not been seen before, and can also be retrained quickly on data of new which have appeared, whereas an expert system would need updated in an expensive and slow process.

More recently hybrid approaches to network intrusion detection have been proposed in papers such as; Powers and He, 2008 and, Panda, Abraham and Patra, 2012, which employ a two stage classification approach. This two stage classification functions by first classifying network traffic as either 'normal' or 'intrusive'. Once intrusive traffic has been identified it is fed into a second stage which then determines the type of intrusion. Performing the detection and classification in this manner is beneficial as two separate classifiers can be used, allowing each to specialise, producing a higher rate of intrusion detection and a lower false positive rate.

The research questions which this dissertation aims to answer are surrounding the comparison of these hybrid approaches in comparison to single stage monolithic detectors, which is discussed in more detail in section 2.6.

2.5 Machine Learning Algorithms

This section will detail the Machine learning algorithms which have been chosen to be evaluated and compared in this thesis.

2.5.1 k-Nearest Neighbours

The k-nearest neighbours (k-NN) is one of the most simple machine learning classification and regression algorithms. k-NN is a lazy algorithm meaning that it uses the training dataset directly and therefore does not require any training time. The algorithm functions by first taking a training set of labelled vectors. A vector which is to be classified is then input, and the algorithm calculates the distance from the input vector to each other point in the training set and selects the k nearest points from the training set. For each of these k nearest points their classifications are totalled and the classification which makes up the most of these neighbours is then output as the class of the input vector. In both Liao and Vemuri, 2002, and in Hautamaki, Karkkainen and Franti, 2004, it has been demonstrated that k-NN based network intrusion detection methods can produce good results. The k-NN algorithm has been selected for this thesis mainly because of its simplicity and according to Jain, Duin and Mao, 2000 it *"can be conveniently used as a benchmark for all the other classifiers since it appears to always provide a reasonable classification performance inmost applications."*.

2.5.2 Artificial Neural Network

According to Caudill, 1987 an artificial neural network (ANN) is *"a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs"*. The ANN is inspired by and aims to recreate the operation of a biological neural network, found within the brains of living animals. The neural network consists of several layers of neurons or nodes, which are interconnected by axioms each of which has its own associated weight which is altered over the course of the training of the network. These weights are adjusted through a method called backpropagation. A network will typically consist of three layers; the input layer, the hidden layer(s), and the output layer, shown in Figure 1.

The input layer is where the neural network receives the data which it is to process. This input layer is then connected via weighted axioms to the hidden layer(s). The hidden layer may consist of many layers and is responsible for the recognition of patterns within the input data. These hidden layers are then connected to the output layer, which gives the result

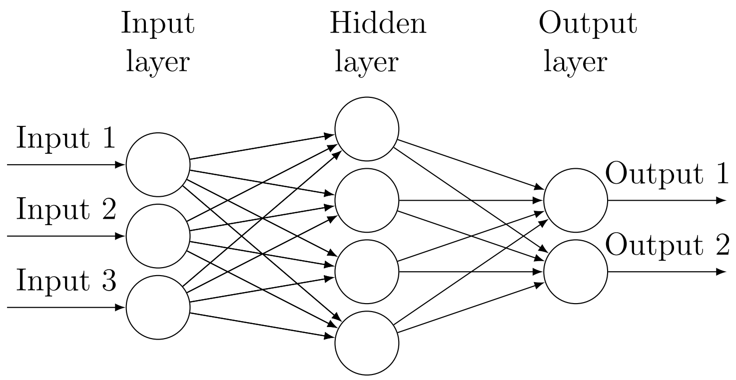

Figure 1: Artificial Neural Network Layers

or classification of the input data. The ANN is trained by feeding in a labelled dataset containing inputs and expected output(s). Each data entry which is fed into the network slightly alters the weights of all axioms within the network depending on the inputs and outputs, which when performed a large number of times allows the network to recognise patterns within input data and attempt to predict the correct output. Neural networks were first proposed for use within the field of network detection intrusion by Herve Debar, Becker and Siboni, 1992, who found them to be a promising method. ANNs are especially good for use with data classification problems and in turn with network intrusion detection as they can recognise complex patterns within datasets with a large number of features, such as network traffic Sung, 1998. Some early methods of network intrusion detection were also unable to make any sort of classification beyond a binary one, i.e. normal or intrusive, whereas ANNs can classify data into any number of categories allowing for a greater amount of information about an attack to be gained from the detection Moradi and Zulkernine, 2004.

However, while neural networks excel at classifying information and recognising complex patterns within data, they do suffer from a semantic gap. Garcia-Teodoro, Diaz-Verdejo, Maciá-Fernández and Vázquez, 2009 explain that *"they do not provide a descriptive model that explains why a particular detection decision has been taken."* meaning that the network is essentially a black box, and without the details of the operation of the network it is difficult to determine what features of a connection identify it as intrusive and how effectively it may perform on other tasks and in other areas. The remainder of this section will discuss several different kinds of ANN which have been successfully applied to network detection intrusion.

This basic form of neural network is also called a feed forward neural network (FFNN) and is the most simple class of neural network available in which the connections between neurons do not form a cycle unlike in a RNN, i.e. information only travels forward from input neurons to output neurons. These kinds of neural networks have been employed in a number of research papers to great effect. Mukkamala, Janoski and Sung, 2002 found that with the use of a neural network intrusions could be accurately detected more than 99% of the time. The same result was also found in Z. Zhang, Li, Manikopoulos, Jorgenson and Ucles, 2001 with a 99% detection rate. Linda, Vollmer and Manic, 2009, S. C. Lee and Heinbuch, 2001, and Moradi and Zulkernine, 2004 also all achieved the same results as other studies. Using these results it is clear that using a FFNN is a viable method detecting network intrusions, as well as being simple to implement within a limit time frame. S. C. Lee and Heinbuch, 2001 also found that these networks are extremely adept at detecting novel attacks which is a desirable trait in a NIDS. It is for these reasons that a FFNN has been chosen to be implemented in this thesis.

2.5.3 Self Organising Map

The self organising map (SOM) is a type of ANN first proposed in the paper Kohonen, 1982 and is used to map high dimensional data into lower dimensions. The SOM is unlike the other neural networks which are to be examined as it can learn to classify data without supervision. This means that whereas a regular neural network will require an input vector and and output vector, the SOM will learn to classify data without the need for an output vector. This lack of need for supervised training can be extremely useful in the context of network intrusion detection as described by Rhodes, Mahaffey and Cannady, 2000 *"This approach is particularly powerful because the self-organizing map never needs to be told what intrusive behaviour looks like. By learning to characterize normal behaviour, it implicitly prepares itself to detect any aberrant network activity."* both in anomaly detection and especially in misuse detection as normal behaviour can be continuously fed into it without the need for examples of intrusive behaviour.

The SOM has been implemented and tested within a number of research papers with great success. In Powers and He, 2008 and Depren, Topallar, Anarim and Ciliz, 2005 it was found that the use of an SOM showed favourable false positive rates and attack classification results over other intrusion detection methods using the KDD 1999 Cup dataset. Similarly, in Lichodzijewski, Zincir-Heywood and Heywood, 2002 the researchers found that a SOM produced good results on the DARPA 1998 dataset with a low false positive rate and a correct classification rate of more than 95%. Kayacik, Zincir-Heywood and Heywood, 2003 too found SOM to be an effective method of classification with results much similar to the previous studies performed on the KDD 1999 Cup dataset however with a much higher rate of false positives.

These papers demonstrate that the SOM is a viable method for the detection of network intrusions giving high rate of correct classifications and low false positive rate, however this algorithm will not be implemented as part of this thesis. This algorithm has been chosen to not be implemented due to an unfamiliarity on the part of the researcher and also time constraints of the project associated with this unfamiliarity regarding the research and implementation it, with other methods being less time consuming to implement.

2.5.4 Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural network where the connections within the network connect back to previous neurons in order to form a cycle of nodes as shown in Figure 2. Having the neural network be structured in this manner allows it to process a sequence of input vectors allowing it to process data which relies on other vectors for context. RNNs are typically applied to problems such as handwriting recognition or speech recognition, however they can be particularly effective in network detection intrusion in recognising sequences of network traffic which alone are not suspicious but in a specific order can then be classified as intrusive behaviour.

The first paper to advocate for the use of RNNs within the field of network intrusion detection was Hervé Debar and Dorizzi, 1992 in which it was found RNNs to be a promising method of detecting intrusions. A later paper Ryan, Lin and Miikkulainen, 1998 re-enforces this statement finding this method to be very effective when employed on real worlds data. Two papers using RNNs in order to detect network intrusions, Tong, Wang and Yu, 2009 and Ghosh and Schwartzbard, 1999 found this approach to have high rate of correct detection of 90%-100%, however with a slightly higher false positive rate than other methods of intrusion detection such as the SOM.

There are however some issues in using RNNs for network intrusion detection. In order to train the network for use in detection, sequenced data such as timed network traffic is required. In this thesis, the data set used is the KDD 1999 Cup dataset in which none of the data is sequenced therefore making it unsuitable for testing with a RNN, which is the main reason why this type of neural network will not be implemented.

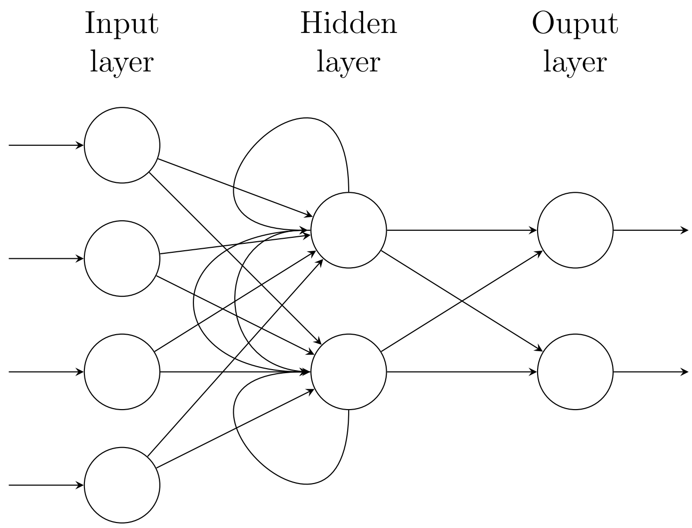

Figure 2: Recurrent Neural Network

2.5.5 Negative Selection

Negative selection is an algorithm in the field of artificial immune systems. Artificial immune systems are a class of algorithm which seek to imitate the immune system of a living creature. De Castro and Timmis, 2002 describes artificial immune systems as *"adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving."*. There are several algorithms related artificial immune systems such as: Clonal Selection, Immune Network, and Dendritic Cell, however the algorithm which will be examined is the negative selection algorithm.

The negative selection algorithm takes its inspiration from the generation of T cells in the immune system. These T cells are capable of distinguishing between the bodies own cells and foreign cells, and are created pseudo-randomly. These cells then undergo a censoring process called negative selection in which cells that recognise the body's own cells are destroyed leaving only ones which detect foreign cells. This process is the basis for the negative selection algorithm where detectors (T cells) are generated by some method, randomly or otherwise, and detectors which detect self are deleted Forrest, Perelson, Allen and Cherukuri, 1994.

In the negative selection algorithm detectors must be represented by some means. In papers such as Dasgupta and Forrest, 1996 and Kim and Bentley, 2001b detectors are represented by fixed length bit string where each portion of the string is a binary representation of some feature of the input data. In order to determine whether or not these detectors match self, the detector is checked against each entry in the training set and if both of the strings contain the same stretch of r uninterrupted bits then the strings are said to match Powers and He, 2008. The detectors which do not match self are then used to detect intrusive behaviour by attempting to match incoming data represented as bit strings. If the incoming bit string matches any of the detectors it is flagged as intrusive. The use of binary strings as detectors however presents similar problems as found in neural networks. When a string is flagged as being intrusive it is difficult to extract semantic meaning from the detectors, i.e. it is apparent that the string is intrusive but information about what features of the connection made it intrusive are not readily available González and Dasgupta, 2003. Kim and Bentley, 2001a also found that the use of bit strings for detectors becomes infeasible when using a dataset which contains a lot of different features for each entry, such as in the KDD 1999 Cup dataset. A solution to this problem lies in the use of real-valued detectors in the place of bit string detectors. Through the use of real values when defining detectors this allows a great amount of domain knowledge to be extracted from subsequent results. For example when a detector is activated and a connection flagged as being intrusive the detector can be examined and the exact features and values of the connection which triggered the detector can be viewed and higher level information can be extracted. In comparison it can be extremely difficult and time consumting to try and extract semantic knowledge from a bit string as it can be unclear what triggered the detector. In order to evolve detectors, there are two main methods of doing so: through random generation, or by means of a genetic algorithm. In the random generation method, a set of self is required, in the form of a dataset of non intrusive behaviour only. Detectors are then generated at random and tested to see whether or not they match the set of self using some method, i.e. r-contiguous bits, r-chunk matching, hamming distance etc. If the generated detector matches self it is discared and a new detector is generated until the detector does not match self and it is stored. Generating detectors by means of a genetic algorithm is described by Powers and He, 2008. In this paper a population of detectors is initialised where each feature of the detector has a 50% chance to be initialised as blank and ignored in detection. Leaving fields blank in this way increases the generality of the detector, allowing it to cover as large an area of non-self as possible and to detect more kinds of attacks. The fields which were not left blank during the initialisation are then randomly assigned a value from a list of allowed values for that feature.

When evolving the population two parents are bred using a uniform crossover to produce a single child which then has a small probability of mutation. This mutation takes a field from the child and replaces its value with another value which is randomly chosen from the list of allowed values. This newly created child will then replace the parent which it is most similar to if it has a greater fitness. At the end of the evolution process detectors which match self are removed by comparing each detector with the self-set. Generating detectors in such a manner is advantageous as it allows detectors to be created quickly when compared to a purely random generation process.

This method of intrusion detection through the use of the negative selection algorithm has been implemented in a number of papers to great effect. Powers and He, 2008 shows that the negative selection algorithm evolved using a genetic algorithm can acheive detection rates of up to 98%. Dasgupta and González, 2002 found similar results similarly using a genetic algorithm with a self detection rate of 96% and a non-self detection rate of up to 86%. This demonstrates that negative selection is an effective and viable method of network intrusion detection.

2.6 Research Contribution

During this dissertation the there are a number of goals which are aimed to be met and questions answered in order to ultimatley contribute to the research of network intrusion detection is some way. The main question which is to be answered is what is the difference in performance between a single stage classifier and a two stage classifier in the context of network intrusion detection, and to determine what the configuration of machine learning classifiers discussed within this project produces the highest accuracy network intrusion detection and classification of intrusive network connections, where accuracy is defined as a high number of true positives and a low number of false positives.

Within the field of network intrusion detection there have been numerous papers describing methods of detecting intrusion using single stage classifiers such as citations wherein network traffic data is fed into the classifier and the single classifier will determine whether or not the connection is intrusive, and if it is intrusive what class of intrusion it is. Hybrid intrusion detection systems function similarly however the main difference is that there are two stages to the classification process. The first stage will be some form of anomaly detector which reads in network data and determines whether or not a connection is intrusive or not without specifying the type of attack that the intrusive connection is. Then the first stage of the classifier will send all data deemed as intrusive to the second stage of the classifier which will then determine the type of attack. Building an intrusion detection system in a manner such as this allows each stage of the system to specialise in detecting different aspects of a connection and therefore producing a higher detection and classification accuracy. While there have been several papers on hybrid classifiers in the domain of network intrusion detection such as Powers and He, 2008, Panda et al., 2012, and J. Zhang and Zulkernine, 2006, there is an apparent distinct lack of papers which directly compare the performance of a single stage monolithic classifier to that of a two-stage stage classifier and investigate different arrangements of machine learning algorithms to evaluate their performance. This gap in research papers is what this dissertation aims to fill.

The project will achieve that goal in the following manner: A number of machine learning algorithms will be chosen to be implemented, i.e. k-nearest neighbors, artificial neural networks, and negative selection. These algorithms will be housed by a peice of software which will be written to run them, obtain metrics and display information on the performance of each algorithm, in the form of tables and graphs, etc. Once this information has been extracted from each algorithm, an analysis and evauluation will be performed on the accuracy of each method, and a conclusion drawn.

2.7 Dataset

The dataset which was to perform the evaluation of the network intrusion detection methods is the KDD Cup 1999 dataset. This dataset comes from the Third International Knowledge Discovery and Data Mining Tools Competition and is based on data captured in the DARPA'98 IDS evaluation program Lippmann et al., 2000, Tavallaee, Bagheri, Lu and Ghorbani, 2009.

This dataset has been widely criticized by researchers and experts due to several factors such as; having a large number of redundant records *"which cause the learning algorithm to be biased towards the most frequent records, thus prevent it from recognizing rare attack records"* Panda et al., 2012 , and in Vasudevan, Harshini and Selvakumar, 2011, for being outdated meaning that it does not account for new developments in network attacks. This is not of concern during this project as the dataset is used purely as a proof of concept and will not be deployed into an actual network intrusion detection role. There are other datasets which have been proposed for use such as the NSL-KDD which is a revised version of the KDD99 dataset which has been shown to have eliminated many of the faults of the KDD99 dataset while also achieving better performance in the training of network intrusion detection systems Dhanabal and Shantharajah, 2015. While all of the criticisms against the KDD Cup 1999 dataset are well founded and a legitimate cause for concern it has been chosen as it is the single most used dataset and researched dataset in the entire network intrusion detection field. With this large amount of research having been performed on this dataset the results from such studies can be used as confirmation that the algorithms which are to be implemented are performing correctly, and also can be used to make an accurate comparisons of results. The KDD Cup 99 dataset also has a version released which has all redundant entries removed while maintaining the correct ratio of non-intrusive entries to attacks, allowing for much faster training of algorithms and collection of data, which is a priority as there is a time constraint which must be followed during this project.

2.8 Literature Conclusion

The research field of Network Intrusion Detection is one of upmost important due to the widespread prevelance of network security breaches and attacks. There have been a large number of studies performed in this area, with each resulting in varying levels of success. The algorithms chosen for this investigation have been proven to be effective at detecting network intrusions, and studies have also shown that through the use of hybrid network intrusion detection systems which make use of multiple classification stages and methods an increase in accurate classification rate. However, machine learning techniques have not yet proven accurate enough in their classification and detection rates to be deployed in a commercial setting, without substantial supervision. There is also a distinct lack within the research in making direct comparisons between different configurations of multiple stage classifiers, and also in making direct comparisons to their single stage counterparts. As a result of this, the conclusion has been reached that conducting an investigation into this area may prove beneficial for future studies, and may provide an insight into what configurations of multiple stage classifiers are effective in the context of network intrusion detection.

3 Existing Software

3.1 WEKA

3.2 SPLUNK

4 Software

This section of the report will provide a high level description of the functionality and requirements of the software as well as describing the role of the user which will interact with the system. Detailed descriptions of functionality will also be included, and a list of hardware and software constraints, dependencies and assumptions made about the system and users. A detailed testing description will then be given outlining any and all unit testing and functional testing which will take place. Finally an evaluation will take place assessing how well the produced software meets its initial requirements and design.

4.1 Overall Description

4.1.1 Product Perspective

4.1.2 Product Functions

4.1.3 User Characteristics

This software is primarily focused towards researchers and students interested in comparing machine learning classification algorithms with a single stage against those with multiple stages. The software may be used by them in order to quickly make comparisons between different configurations of classifiers and to assess the effectiveness of these configurations. They will be able to enter their own datasets and classifiers and receive results in a standard format, and to view these results in a graph.

4.1.4 Operating Environment

In order to run the software the following requirements must be met:

- Python 3.6.4+ with the following packages:
  - NumPy 1.14.0+
  - Pandas 0.22.0+
  - Pandas-ml 0.5.0+
  - Scikit-learn 0.19.1+
  - Scipy 1.0.0
  - Matplotlib 2.1.1+
  - PyQt 5.9.1+
- One of the following operating systems:
  - Windows 7 (x86) or greater
  - macOS 10.11 or greater
  - Linux with X11
- A system capable of running one of the above operating systems.

4.2 Specific Requirements

4.2.1 User Interface

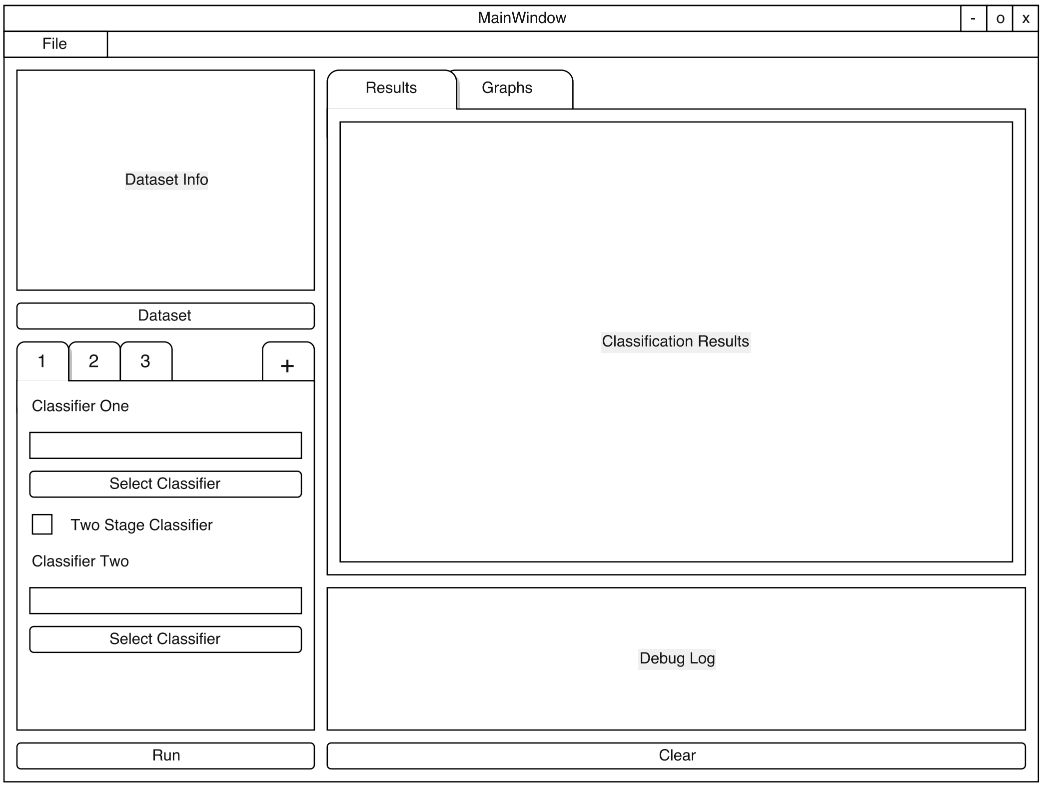

Figure 3: Main Window Result Tab Wireframe

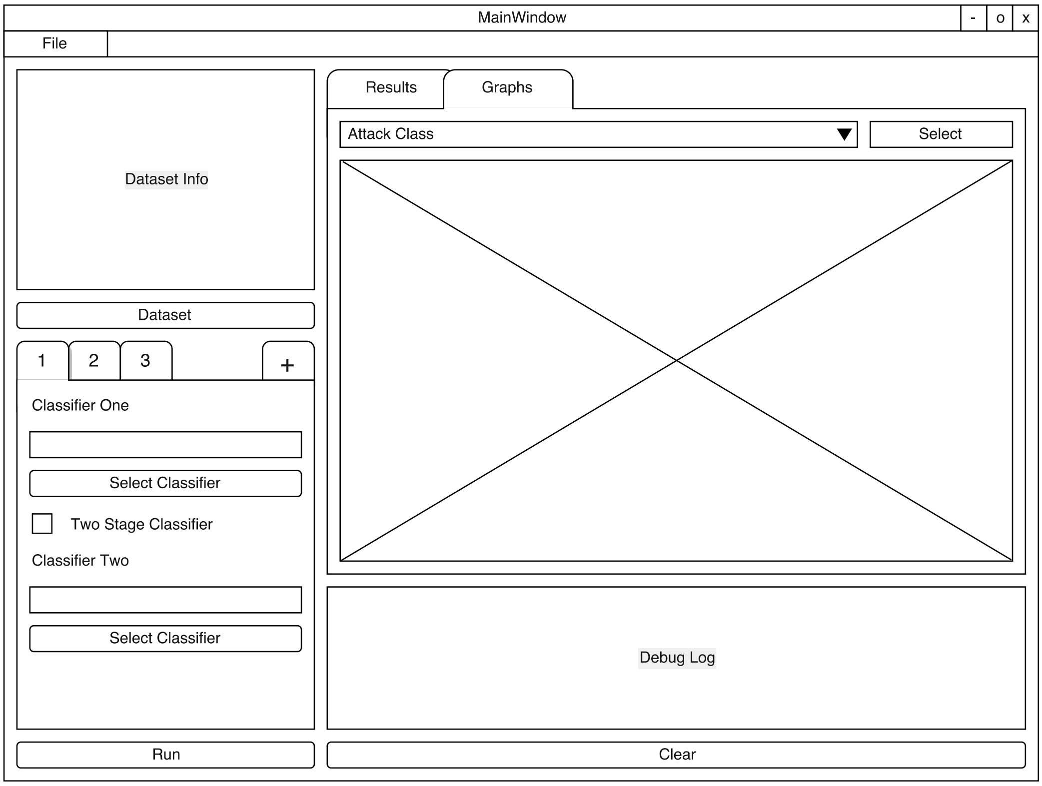

Figure 4: Main Window Graph Tab Wireframe

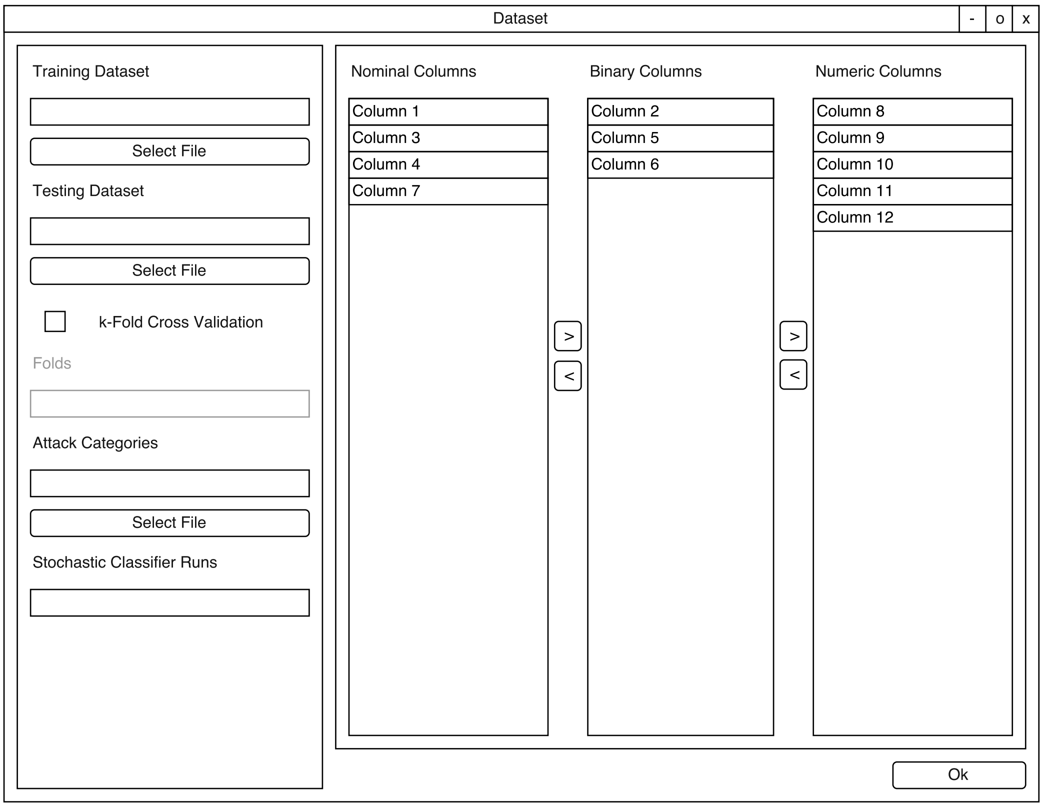

Figure 5: Dataset Window Wireframe

4.2.2 Functional Requirements

4.2.2.1 FR1 - Select Classifiers

ID: FR1
TITLE: Select Classifiers
PRIORITY: High
DESC:
DEP: N/A

4.2.2.2 FR2 - Two Stage Classification

ID: FR2
TITLE: Two Stage Classification
PRIORITY: High
DESC:
DEP: N/A

4.2.2.3 FR3 - Run Classifiers

ID: FR3
TITLE: Run Classifiers
PRIORITY: High
DESC:
DEP: FR1, FR2

4.2.2.4 FR4 - Create New Classifier

ID: FR4
TITLE: Create New Classifier
PRIORITY: Low
DESC:
DEP: N/A

4.2.2.5 FR5 - Debug Output

ID: FR5
TITLE: Debug Output
PRIORITY: Medium
DESC:
DEP: N/A

4.2.2.6 FR6 - Multiple Classifier Configurations

ID: FR6
TITLE: Multiple Classifier Configurations
PRIORITY: Medium
DESC:
DEP: FR1, FR2

4.2.2.7 FR7 - Aggregate Results by Attack

ID: FR7
TITLE: Aggregate Results by Attack
PRIORITY: High
DESC:
DEP: FR3

4.2.2.8 FR8 - Write Results to File

ID: FR8
TITLE: Write Results to File
PRIORITY: Medium

DESC:
DEP: N/A

4.2.2.9 FR9 - Get Classification Results

ID: FR9
TITLE: Get Classification Results
PRIORITY: High
DESC:
DEP: FR3

4.2.2.10 FR10 - Graph Results

ID: FR10
TITLE: Graph Results
PRIORITY: High
DESC:
DEP: FR9

4.2.2.11 FR11 - Select Training Dataset

ID: FR11
TITLE: Select Training Dataset
PRIORITY: High
DESC:
DEP: N/A

4.2.2.12 FR12 - Select Testing Dataset

ID: FR12
TITLE: Select Testing Dataset
PRIORITY: High
DESC:
DEP: N/A

4.2.2.13 FR13 - k-Fold Cross Validation

ID: FR13
TITLE: k-Fold Cross Validation
PRIORITY: High
DESC:
DEP: FR11

4.2.2.14 FR14 - Select Dataset Column Labels

ID: FR14
TITLE: Select Dataset Column Labels
PRIORITY: High
DESC:
DEP: N/A

4.2.2.15 FR15 - Select Dataset Attack Categories

ID: FR15
TITLE: Select Dataset Attack Categories
PRIORITY: High
DESC:
DEP: N/A

4.2.2.16 FR16 - Categorise Dataset Fields

ID: FR16
TITLE: Categorise Dataset Fields
PRIORITY: High
DESC:
DEP: FR14

4.2.2.17 FR17 - One Hot Encoding

ID: FR17
TITLE: One Hot Encoding
PRIORITY: High
DESC:
DEP: FR18

4.2.2.18 FR18 - Load Data

ID: FR18
TITLE: Load Data
PRIORITY: High
DESC:
DEP: FR11, FR12, FR14, FR15

4.2.2.19 FR19 - k-Nearest Neighbour Classifier

ID: FR19
TITLE: k-Nearest Neighbour Classifier
PRIORITY: High

DESC:
DEP: N/A

4.2.2.20 FR20 - Multi-layer Perceptron Classifier

ID: FR20
TITLE: Multi-layer Perceptron Classifier
PRIORITY: High
DESC:
DEP: N/A

4.2.2.21 FR21 - Negative Selection Classifier

ID: FR21
TITLE: Negative Selection Classifier
PRIORITY: High
DESC:
DEP: N/A

4.2.2.22 FR22 - Support Vector Machine Classifier

ID: FR22
TITLE: Support Vector Machine Classifier
PRIORITY: High
DESC:
DEP: N/A

4.2.2.23 FR23 - Stochastic Classifier Averaging

ID: FR23
TITLE: Stochastic Classifier Averaging
PRIORITY: High
DESC:
DEP: FR1, FR2

4.2.2.24 FR24 - Feature Selection

ID: FR24
TITLE: Feature Selection
PRIORITY: Low
DESC:
DEP: N/A

4.2.3 Non-Functional Requirements

4.2.3.1 QR1 - Robustness

ID: QR1
TITLE: Robustness
PRIORITY: High
DESC:

4.2.3.2 QR2 - Responsiveness

ID: QR2
TITLE: Responsiveness
PRIORITY: Medium
DESC: UI should not hang during stuff.

4.2.3.3 QR3 - Usability

ID: QR3
TITLE: Usability
PRIORITY: Medium
DESC:

4.2.3.4 QR4 - Maintainability

ID: QR4
TITLE: Maintainability
PRIORITY: High
DESC:

4.2.3.5 QR5 - Portability

ID: QR5
TITLE: Portability
PRIORITY: Low
DESC: Work on different systems and OS's

4.2.3.6 QR6 - Correctness

ID: QR6
TITLE: Correctness
PRIORITY: High
DESC: Results MUST be correct.

4.3 Testing

Paragraph about testing, unit testing and why

Table 1: Test Case Descriptions

ID	Objective	Precondition	Steps	Test Data	Expected Result
TC1	Create new classifier tab	There is only one tab in the classifier tab box.	- Press the '+' button above the classifier tab box.	N/A	A new tab is created with the number 2 on it.
TC2	Open dataset window.	N/A	- Press the 'Select Dataset..' button in the main window.	N/A	Dataset window is shown.
TC3	Select training dataset.	Dataset window is open.	- Press the 'Select Training Set..' button. - Select a file.	Any .csv file.	Absolute path of the file appears in the textbox below 'Training Dataset' label.
TC4	Deselect k-Fold Cross Validation.	Dataset window is open and k-Fold Cross Validation checkbox is checked.	- Press the k-Fold Cross Validation checkbox.	N/A	Testing dataset label, textbox and button become enabled and folds spinbox is disabled.
TC5	Select testing dataset.	Dataset window is open.	- Press the 'Select Testing Set...' button. - Select a file.	Any valid .csv file.	Absolute path of the file appears in the textbox below 'Training Dataset' label.
TC6	Select dataset fields.	Dataset window is open.	- Press the 'Select Dataset Fields' button. - Select a file.	A .csv file containing two columns, the second of which is one of the values: Numeric, Nominal, Binary.	Absolute path of the file appears in the textbox below 'Dataset Fields' label, and the 'Numeric', 'Nominal', and 'Binary' list views are populated.

TC7	Select attack categories.	Dataset window is open.	- Press the 'Select Attack Categories' button. - Select a file.	Any valid .csv file.	Absolute path of the file appears in the textbox below 'Attack Categories' label.
TC8	Display dataset info in main window.	Valid dataset files have been selected and dataset window is open.	- Press the OK button in the bottom right of the dataset window.	N/A	The dataset window closes, showing no error message and the textbox in the top left of the main window is populated with information regarding the dataset.
TC9	Select classifier.	N/A	- Press the 'Select Classifier' button below the 'Classifier One' label. - Select a file.	Any .py file.	Absolute path of the file appears in the textbox below the 'Classifier One' label.
TC10	Select two stage classification.	Two stage classification checkbox is unchecked.	- Press the 'Two Stage Classification' checkbox.	N/A	The 'Second Classifier' label, textbox, and 'Select Classifier' button are enabled.
TC11	Select second classifier.	The 'Two Stage Classification' checkbox is checked.	- Press the 'Select Classifier' button below the 'Classifier Two' label. - Select a file.	Any .py file.	Absolute path of the file appears in the textbox below the 'Classifier One' label.
TC12	Error message on invalid training dataset filename.	N/A	- Enter a filepath which does not exist into the textbox below the 'Training Dataset' label. - Supply every other textbox with valid filepaths. - Press the OK button in the bottom right of the dataset window.	N/A	An error message is shown and the window does not exit.

TC13	Error message on invalid testing dataset filename.	N/A	- Enter a filepath which does not exist into the textbox below the 'Testing Dataset' label. - Supply every other textbox with valid filepaths. - Press the OK button in the bottom right of the dataset window.	N/A	An error message is shown and the window does not exit.
TC14	Error message on invalid field names filename.	N/A	- Enter a filepath which does not exist into the textbox below the 'Dataset Fields' label. - Supply every other textbox with valid filepaths. - Press the OK button in the bottom right of the dataset window.	N/A	An error message is shown and the window does not exit.
TC15	Error message on invalid attack categories filename.	N/A	- Enter a filepath which does not exist into the textbox below the 'Attack Categories' label. - Supply every other textbox with valid filepaths. - Press the OK button in the bottom right of the dataset window.	N/A	An error message is shown and the window does not exit.
TC16	Error message on invalid training dataset.	Valid dataset files have been selected, except from training dataset which is invalid. A valid classifier configuration has been selected.	- Press the 'Run' button on the main window.	N/A	An error message is displayed and the program does not crash.
TC17	Error message on invalid testing dataset.	Valid dataset files have been selected, except from testing dataset which is invalid. A valid classifier configuration has been selected.	- Press the 'Run' button on the main window.	N/A	An error message is displayed and the program does not crash.

TC18	Error message on invalid field names.	Valid dataset files have been selected, except from dataset fields which do not correspond to the dataset. A valid classifier configuration has been selected.	- Press the ’Run’ button on the main window.	N/A	An error message is displayed and the program does not crash.
TC19	Error message on invalid attack categories.	Valid dataset files have been selected, except from attack categories which do not correspond to the dataset. A valid classifier configuration has been selected.	- Press the ’Run’ button on the main window.	N/A	An error message is displayed and the program does not crash.
TC20	Dataset window is populated with existing information.	Dataset file paths have been previously selected and the OK button pressed on the dataset window.	- Press the ’Select Dataset..’ button on the main window.	N/A	The text boxes containing the paths to files should all be filled with the information which was entered previously, as well as fold count and stochastic classifier run count.
TC21	Error message on invalid classifier.	N/A	- Press the ’Select Classifier’ button the main window. - Select an invalid .py file. - Press the ’Run’ button.	An invalid .py file.	An error message is displayed and the program does not crash.
TC22	Delete classifier tab.	There is more than one tab in the classifier tab.	- Press the ’x’ button next to the tab number in the classifiers tabs in any tab except the first.	N/A	The tab in which the ’x’ button was pressed should close.
TC23	Show results for all classifiers run.	Valid dataset has been selected and valid classifier configuration has been selected.	- Press the ’Run’ button.	N/A	A tab should be created on the main tab container for each classifier configuration run, and populated with classification information.

TC24	Show graph of classification results.	Classifiers have been run and results retrieved.	- Select the 'Graphs' tab on the main window. - Select a class from the drop down in the 'Graphs' tab. - Press the 'Show' button.	N/A	A graph is shown which has a series for each classifier configuration specified.

5 Methodology

5.1 Implementation

5.1.1 PyQt

What is PyQT (PSF, 2018c) Why PyQT Cross platform not java so allows dev in all same language Extensive documentation

5.1.2 Scikit-Learn

What is scikit (PSF, 2018d) Why Pre implemented algorithms ensure correctness and save time Provides methods for scaling data Provides methods for generating Confusion matrices and producing classification reports quickly.

5.1.3 Pandas

hat is pandas (PSF, 2018b) Why Its the best

5.1.4 Matplotlib

What is matplot (PSF, 2018a) why works with pyqt powerfull lots of usefull info online

5.1.5 Software

The software which was written is described with a class diagram, seen in Appendix E specifying each distinct class created along with its methods, properties and relationship with other classes, and with a use case diagram showing the general function of the software from the perspective of the user, as seen in Appendix F. A description of all of the software classes functions is as follows.

5.1.5.1 MainWindow

The MainWindow class is the class which is responsible for all of the logic behind the user interface of the main window. It inherits the QMainWindow class; a PyQt type which provides a main application window, and from Ui_MainWindow, which provides the specification and initialisation of the user interface. The classes main function is handling user input, creating new user interface elements and windows, and allowing the user to view classification results.

5.1.5.2 DatasetWindow

The DatasetWindow class is primarily responsible for defining what dataset the user wishes to use, to specify the data type of each column of said dataset. The window also allows users to choose between a testing dataset or k-fold cross validation and specify a number of folds, and to specify the number of runs a stochastic classifier should be run for. This class inherits from QDialog which provides the class with buttons allowing the user to accept or cancel the information they have input. It also inherits from Ui_DatasetWindow which like Ui_MainWindow provides the user interface for the window.

5.1.5.3 UIObject

UIObject is an interface which is used by other classes which implement a user interface for some window. It is required as when a user interface is inherited by a window class a number of methods are expected to be available in order to create and render the user interface at runtime.

5.1.5.4 Classifier

Classifier is an interface which any classification algorithm which is to be used must implement. It has one property, 'stochastic', which determines whether or not the classifier must be run multiple times, and a 'run' method which carries out the classification and returns the results.

5.1.5.5 QClfSelector

QClfSelector is a class which inherits from QWidget, the base class of all Qt user interface objects, and who's main function is to act as a widget that can be included within MainWindow multiple times. It provides the functionality of accepting user classifier arrangements and then fetching and running those classifiers, using whatever method has been specified by the user, be it k-fold cross validation or using a test dataset, or running stochastic classifiers multiple times and gathering the results.

5.1.5.6 QBarChart

QBarChart is a class which inherits from 'FigureCanvas', a Matplotlib class which interfaces with PyQt in order to create widget which can be displayed within a PyQt application. The class accepts a set of classification results and then constructs a Matplotlib chart which can be displayed in the MainWindow class for the user to view.

5.1.5.7 ErrorMessage

The ErrorMessage Class is the class responsible for displaying an error message popup to users, in a standard manner and encapsulates boiler plate code required for creating and showing a message box. The class inherits from QMessageBox which is a PyQt class provides a modal dialog for informing the user.

5.2 Datasets

5.2.1 NSL-KDD

- Denial of Service (DoS) - Attacker tries to preventlegitimate users from using a service.
- Remote to Local (r2l) - Attacker does not have an account on the victim machine, hence tries to gain access.
- User to Root (u2r) - Attacker has local access to the victim machine and tries to gain super user privileges.
- Probe - Attacker tries to gain information about the target host.

•Remote to Local (r2l):.•User to Root (u2r):•Probe:

Table 2: NSL-KDD Dataset Attack Samples.

Attack	Samples	Category
back	956	dos
buffer_overflow	30	u2r
ftp_write	8	r2l
guess_passwd	53	r2l
imap	11	r2l
ipsweep	3599	r2l
land	18	dos
loadmodule	9	u2r

multihop	7	r2l
neptune	41214	dos
nmap	1493	probe
normal	67343	normal
perl	3	u2r
phf	4	r2l
pod	201	dos
portsweep	2931	probe
rootkit	10	u2r
satan	3633	probe
smurf	2646	dos
spy	2	r2l
teardrop	892	dos
warezclient	890	r2l
warezmaster	20	r2l

Table 3: NSL-KDD Dataset Categories Samples.

DoS	Probe	u2r	r2l	Normal
45927	11656	52	995	67343

5.2.2 UNSW-NB15

Table 4: UNSW-NB15 Dataset Attack Samples.

Attack	Samples
Analysis	677
Backdoor	583
DoS	4089
Exploits	11132
Fuzzers	6062
Generic	18871
Reconnaissance	3496
Shellcode	378
Worms	44
Normal	37000

5.3 Data Preprocessing

5.4 Two Stage Classification

5.5 Metrics

5.5.1 Precision

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

5.5.2 Recall

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

5.5.3 f1-Score

$$f_1 = \frac{2tp}{2tp + fp + fn}$$

6 Evaulation

6.1 Software

Evaluate the software How many of the function requirements were met How many of the non functional requirements were met Is the user interface like it was designed How does it compare to existing software

6.2 k-Nearest Neighbours

6.3 Multi-layer Perceptron

6.4 Support Vector Machine

6.5 Single-Stage Classifiers Performance

6.6 Two-stage Classifiers Performance

7 Conclusion

7.1 Research Questions

What is the rate of accurate detection and classification of network intrusions by single stage machine learning classification methods?

What is the rate of accurate detection and classification of network intrusions by two stage machine learning classification methods?

Which method of classification i.e. single stage or two stage, is more accurate and by what amount?

Which configuration of algorithms in the two stage classifier produces the most accurate results?

7.2 Has the Project met it's Aims and Objectives?

7.3 Reflection

7.4 Learning Outcomes

7.5 Further Research

References

Caudill, M. (1987). Neural networks primer, part i. *AI expert*, *2*(12), 46–52.

Dasgupta, D. & Forrest, S. (1996). Novelty detection in time series data using ideas from immunology. In *Proceedings of the international conference on intelligent systems* (pp. 82–87).

Dasgupta, D. & González, F. (2002). An immunity-based technique to characterize intrusions in computer networks. *IEEE transactions on Evolutionary Computation*, *6*(3), 281–291.

De Castro, L. N. & Timmis, J. (2002). *Artificial immune systems: A new computational intelligence approach*. Springer Science & Business Media.

Debar, H. [Herve], Becker, M. & Siboni, D. (1992). A neural network component for an intrusion detection system. In *Research in security and privacy, 1992. proceedings., 1992 ieee computer society symposium on* (pp. 240–250). IEEE.

Debar, H. [Hervé] & Dorizzi, B. (1992). An application of a recurrent network to an intrusion detection system. In *Neural networks, 1992. ijcnn., international joint conference on* (Vol. 2, pp. 478–483). IEEE.

Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on software engineering*, (2), 222–232.

Depren, O., Topallar, M., Anarim, E. & Ciliz, M. K. (2005). An intelligent intrusion detection system (ids) for anomaly and misuse detection in computer networks. *Expert systems with Applications*, *29*(4), 713–722.

Dhanabal, L. & Shantharajah, S. (2015). A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*, *4*(6), 446–452.

Forrest, S., Perelson, A. S., Allen, L. & Cherukuri, R. (1994). Self-nonself discrimination in a computer. In *Research in security and privacy, 1994. proceedings., 1994 ieee computer society symposium on* (pp. 202–212). Ieee.

Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G. & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, *28*(1), 18–28.

Ghosh, A. K. & Schwartzbard, A. (1999). A study in using neural networks for anomaly and misuse detection. In *Usenix security symposium* (Vol. 99, p. 12).

González, F. A. & Dasgupta, D. (2003). Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, *4*(4), 383–403.

Hautamaki, V., Karkkainen, I. & Franti, P. (2004). Outlier detection using k-nearest neighbour graph. In *Pattern recognition, 2004. icpr 2004. proceedings of the 17th international conference on* (Vol. 3, pp. 430–433). IEEE.

Ilgun, K., Kemmerer, R. A. & Porras, P. A. (1995). State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, *21*(3), 181–199. doi:10.1109/32.372146

Jain, A. K., Duin, R. P. W. & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(1), 4–37.

Javitz, H. S., Valdes, A., Breen, G. & Patton, R. D. (1994). The nides statistical component description and justification.

K, R., Jayesh N S, P. S., Tom R, G. P. & Mark B, V. W. (2017). *Cyber security breaches survey.* Ipsos MORI Social Research Institute.

Kayacik, H. G., Zincir-Heywood, A. N. & Heywood, M. I. (2003). On the capability of an som based intrusion detection system. In *Proceedings of the international joint conference on neural networks, 2003.* (Vol. 3, 1808–1813 vol.3). doi:10.1109/IJCNN.2003.1223682

Kim, J. & Bentley, P. J. (2001a). An evaluation of negative selection in an artificial immune system for network intrusion detection. In *Proceedings of the 3rd annual conference on genetic and evolutionary computation* (pp. 1330–1337). Morgan Kaufmann Publishers Inc.

Kim, J. & Bentley, P. J. (2001b). Towards an artificial immune system for network intrusion detection: An investigation of clonal selection with a negative selection operator. In *Evolutionary computation, 2001. proceedings of the 2001 congress on* (Vol. 2, pp. 1244–1252). IEEE.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetics*, *43*(1), 59–69.

Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A. & Srivastava, J. (2003). A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 siam international conference on data mining* (pp. 25–36). SIAM.

Lee, S. C. & Heinbuch, D. V. (2001). Training a neural-network based intrusion detector to recognize novel attacks. *IEEE Transactions on systems, man, and Cybernetics-Part A: Systems and Humans*, *31*(4), 294–299.

Lee, W., Stolfo, S. J. & Mok, K. W. (1999). A data mining framework for building intrusion detection models. In *Proceedings of the 1999 ieee symposium on security and privacy (cat. no.99cb36344)* (pp. 120–132). doi:10.1109/SECPRI.1999.766909

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

Lichodzijewski, P., Zincir-Heywood, A. N. & Heywood, M. I. (2002). Dynamic intrusion detection using self-organizing maps. In *The 14th annual canadian information technology security symposium (citss)*.

Linda, O., Vollmer, T. & Manic, M. (2009). Neural network based intrusion detection system for critical infrastructures. In *Neural networks, 2009. ijcnn 2009. international joint conference on* (pp. 1827–1834). IEEE.

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., . . . Zissman, M. A. (2000). Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Darpa information survivability conference and exposition, 2000. discex '00. proceedings* (Vol. 2, 12–26 vol.2). doi:10.1109/DISCEX.2000.821506

Moradi, M. & Zulkernine, M. (2004). A neural network based system for intrusion detection and classification of attacks. In *Proceedings of the ieee international conference on advances in intelligent systems-theory and applications* (pp. 15–18).

Mukkamala, S., Janoski, G. & Sung, A. (2002). Intrusion detection using neural networks and support vector machines. In *Neural networks, 2002. ijcnn'02. proceedings of the 2002 international joint conference on* (Vol. 2, pp. 1702–1707). IEEE.

Panda, M., Abraham, A. & Patra, M. R. (2012). A hybrid intelligent approach for network intrusion detection. *Procedia Engineering*, *30*, 1–9.

Powers, S. T. & He, J. (2008). A hybrid artificial immune system and self organising map for network intrusion detection. *Information Sciences*, *178*(15), 3024–3042.

PSF. (2018a). Matplotlib. Retrieved March 14, 2018, from https://pypi.python.org/pypi/matplotlib

PSF. (2018b). Pandas. Retrieved March 14, 2018, from https://pypi.python.org/pypi/pandas

PSF. (2018c). Pyqt5. Retrieved March 14, 2018, from https://pypi.python.org/pypi/PyQt5

PSF. (2018d). Scikit-learn. Retrieved March 14, 2018, from https://pypi.python.org/pypi/scikit-learn/0.19.1

Rhodes, B. C., Mahaffey, J. A. & Cannady, J. D. (2000). Multiple self-organizing maps for intrusion detection. In *Proceedings of the 23rd national information systems security conference* (pp. 16–19).

Ryan, J., Lin, M.-J. & Miikkulainen, R. (1998). Intrusion detection with neural networks. In *Advances in neural information processing systems* (pp. 943–949).

Sommer, R. & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *Security and privacy (sp), 2010 ieee symposium on* (pp. 305–316). IEEE.

Sung, A. (1998). Ranking importance of input parameters of neural networks. *Expert Systems with Applications*, *15*(3), 405–411.

Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *Computational intelligence for security and defense applications, 2009. cisda 2009. ieee symposium on* (pp. 1–6). IEEE.

Tong, X., Wang, Z. & Yu, H. (2009). A research using hybrid rbf/elman neural networks for intrusion detection system secure model. *Computer physics communications*, *180*(10), 1795–1801.

Vasudevan, A., Harshini, E. & Selvakumar, S. (2011). Ssenet-2011: A network intrusion detection system dataset and its comparison with kdd cup 99 dataset. In *Internet (ah-ici), 2011 second asian himalayas international conference on* (pp. 1–5). IEEE.

Zhang, J. & Zulkernine, M. (2006). A hybrid network intrusion detection technique using random forests. In *Availability, reliability and security, 2006. ares 2006. the first international conference on* (8–pp). IEEE.

Zhang, Z., Li, J., Manikopoulos, C., Jorgenson, J. & Ucles, J. (2001). Hide: A hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In *Proc. ieee workshop on information assurance and security* (pp. 85–90).

Appendices

A Project Overview

Initial Project Overview

SOC10101 Honours Project (40 Credits)

Title of Project: A Comparison of Machine Learning Algorithms to Detect Network Intrusions

Overview of Project Content and Milestones

For this project, the main objective is to research different machine learning strategies for detecting network intrusions and to develop a piece of software which can obtain metrics from these algorithms. The focus of this project will be on a comparison of two-stage classifiers versus single-stage classifiers for detecting and classifying network intrusions. The dataset upon which these methods will be tested is the KDD Cup 1999 dataset, with the possibility of testing upon other data sets should time allow it.

The different strategies for machine learning will be gathered by reading relevant research papers and articles within the field of machine learning and network intrusion detection and selecting appropriate algorithms/strategies which are applicable for the chosen data set, and which are also feasible to implement within the timescale.

The software will be a desktop application and will take either a predetermined algorithm or a user submitted algorithm, and a data set of network traffic. The software should then run the algorithm and extract metrics from it such as rates for false positives, true positives, false negatives, true negatives, overall accuracy of classification, etc. These results can then be used to plot graphs and charts to visualise this information to a user in a useful way.

A dissertation will be delivered at the end of the project and should contain a detailed design and plan of the software as well as complete testing strategy and results. Also included in the final report should be a comparison of several of the implemented algorithms to determine which is the most suitable for use if any at all.

A list of milestones for this project goes as the following:

- Initial Project Overview Submitted
- Relevant Algorithms Selected
- Project timescale Completed
- Literature Review Complete
- Software Design Completed
- Algorithms Implemented
- Software Implemented
- Software Testing Plan
- Software Tested
- Algorithm Comparison Completed
- Dissertation Written
- Poster Presentation Completed
- Project Submitted

The Main Deliverable(s):

A list of the main deliverables for the project is as follows:

- Initial Project Overview
- Gantt Chart
- Literary Review
- Interim Report
- Requirement Specification
- Software Design Document
- Software Test Plan
- Software Test Results
- Software Implementation
- Algorithm Implementations
- Meeting Diary
- Algorithm Experimental Results
- Algorithm Comparisons
- Software User Documentation
- Dissertation
- Poster Presentation

The Target Audience for the Deliverable(s):

The target audience for this project could include, machine learning and network intrusion researchers, and computer science students. Researchers may find the comparison of algorithms to be highly useful when carrying out preliminary research and could save time on selecting or discounting an algorithm. Computer science students may also find this project of use for experimenting with different network intrusion methods and different machine learning methods, giving them an insight on what they can be used for and how effective they are.

The Work to be Undertaken:

During this project, the work which must be undertaken is first extensive research of the subject area and collection of relevant sources. The project must then be planned and a timeline of work set out to be completed, with deadlines for each deliverable. Algorithms such as k-nearest neighbour, Artificial neural network, negative selection genetic algorithm, etc, will be implemented, compared and contrasted, specifically the performance of single stage against two-stage classifiers using these algorithms. At the same time as implementing these algorithms a requirement specification and then a design document will be created for the software package as well as a test plan. The design will then be implemented and then be tested according to the test plan. Once fully tested and proven correct the software can then be used to compare the algorithms and obtain metrics from then. The final dissertation will then be written which will include an analysis of the results for each algorithm.

Additional Information / Knowledge Required:

Additional research is required on network intrusion and two stage classifiers to best select the methods which will be implemented and explored. Research into similar software products such as WEKA ("Weka 3 - Data Mining with Open Source Machine Learning Software in Java", 2017) will also be conducted to source ideas and to see in which areas these pieces of software are lacking. Research on each individual algorithm to be implemented will also be required to ensure a correct implementation. And finally, an investigation into relevant libraries which may be used to assist with GUI creation, Graphing, and algorithm implementations.

Information Sources that Provide a Context for the Project:

1. Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, *36*(10), 11994-12000.

2. Sommer, R., & Paxson, V. (2010, May). Outside the closed world: On using machine learning for network intrusion detection. In Security and Privacy (SP), 2010 IEEE Symposium on (pp. 305-316). IEEE.

3. Denning, D. E. (1987). An intrusion-detection model. IEEE Transactions on software engineering, (2), 222-232.

4. Powers, S. T., & He, J. (2008). A hybrid artificial immune system and Self Organising Map for network intrusion detection. Information Sciences, 178(15), 3024-3042.

5. Shon, T., & Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. Information Sciences, 177(18), 3799-3821.

6. Mukkamala, S., Janoski, G., & Sung, A. (2002). Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on* (Vol. 2, pp. 1702-1707). IEEE.

7. Frank, J. (1994, October). Artificial intelligence and intrusion detection: Current and future directions. In *Proceedings of the 17th national computer security conference* (Vol. 10, pp. 1-12).

8. Weka 3 - Data Mining with Open Source Machine Learning Software in Java. (2017). Cs.waikato.ac.nz. Retrieved 21 September 2017, from http://www.cs.waikato.ac.nz/ml/weka/

The Importance of the Project:

This project has importance as there has not been many papers directly comparing implementations of different machine learning approaches to network intrusion detection. While there are software packages which deal with gaining metrics from and comparing algorithms there is not one which is focused solely on network intrusion detection. Making a piece of software which is focused on one area of research may prove to provide greater insights, through more focussed results or through ease of use regarding comparing single and multiple stage classifiers, whereas a more complicated piece of software may take a long time to become acquainted with and to produce results.

The Key Challenge(s) to be Overcome:

The key challenges to be overcome in this project are the implementations of the machine learning algorithms themselves. This is due to a personal lack of experience in implementing machine learning algorithms. Experience is also lacked in understanding formal descriptions of algorithms which may hinder my understanding of techniques when reading research papers. Another challenge will be creating a method of accommodating algorithms by creating interfaces which they will communicate with the main piece of software allowing for any algorithm to be entered by a user.

A.A Project Timeline

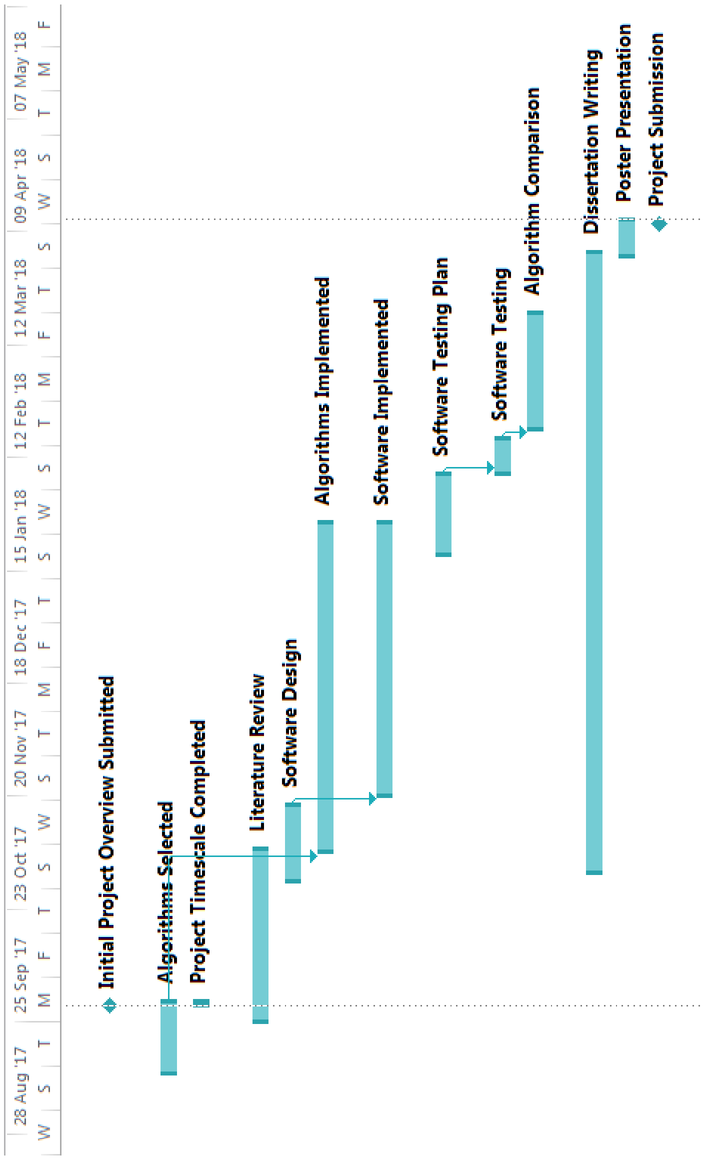

Figure 6: Project Timeline Gantt Chart

B Second Formal Review Output

SOC10101 Honours Project (40 Credits)

Week 9 Report

Student Name: JACK ANDERSON

Supervisor: SIMON POWERS

Second Marker: NAGHMEH MORADPOOR

Date of Meeting: 9/11/17

Can the student provide evidence of attending supervision meetings by means of project diary sheets or other equivalent mechanism? **yes** ✓ **no***

If not, please comment on any reasons presented

Please comment on the progress made so far

- weekly meetings & keeping track of them

Is the progress satisfactory? **yes** ✓ **no***

Can the student articulate their aims and objectives? **yes** ✓ **no***

If yes then please comment on them, otherwise write down your suggestions.

- More recent publication.
- Introduction section: more recent attack
- Justification on: ML-based IDS
  : chosen algorithms
  : chosen dataset: NSL-KDD, UNSW-NB15, PU-IDS, ADFA-Linux
- Expansion on search term
- study: Splunk
- Research question: hypothesis to add

* Please circle one answer; if **no** is circled then this **must** be amplified in the space provided

Does the student have a plan of work? yes no*

If yes then please comment on that plan otherwise write down your suggestions.

- work plan has been discussed during the meeting

Does the student know how they are going to evaluate their work? yes no*

If yes then please comment otherwise write down your suggestions.

- comparison with current products: Splunk in terms of performance for 3 algorithms.

Any other recommendations as to the future direction of the project

N/A

Signatures: Supervisor Simon Powell Second Marker Naghmeh Moradpoor

Student Jack Anderson

Please give the student a photocopy of this form immediately after the review meeting; the original should be lodged in the School Office with Leanne Clyde

* Please circle one answer; if **no** is circled then this **must** be amplified in the space provided

C Diary Sheets (or other project management evidence)

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson **Supervisor:** Simon Powers

Date: 15/09/2017 **Last diary date:** N/A

Objectives:

- Complete first draft of IPO.
- Continue reading about Network Intrusion Detection and Two Stage Classifiers
- Investigate WEKA software.

Progress:

- Decided upon language
- Picked several algorithms to implementation
- Read some research papers

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson

Supervisor: Simon Powers

Date: 21/09/2017

Last diary date: 15/09/2017

Objectives:

- Correct issues with IPO
- Continue reading about network intrusion detection and two stage classifiers
- Investigate WEKA software more
- Begin outlining structure of the literature review

Progress:

- Wrote first draft of IPO
- Explored WEKA
- Read research papers

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson

Supervisor: Simon Powers

Date: 28/09/2017

Last diary date: 21/09/2017

Objectives:

- Continue collecting citations
- Begin writing literature review

Progress:

- Finalized IPO
- Collected more citations to do with intrusion detection and ensemble classifiers.
- Investigated WEKA
- Outlined literature review structure

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson **Supervisor:** Simon Powers

Date: 05/10/2017 **Last diary date:** 28/09/2017

Objectives:

- Continue writing literature review

Progress:

- Collected more citations
- Began writing literature review

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson **Supervisor:** Simon Powers

Date: 12/10/2017 **Last diary date:** 05/10/2017

Objectives:

- Continue writing literature review
- Begin implementing the k-nn algorithm

Progress:

- Made progress on literature review about history of intrusion detection
- Collected references regarding specific algorithms

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson

Supervisor: Simon Powers

Date: 20/10/2017

Last diary date: 12/10/2017

Objectives:

- Go into more detail regarding the negative selection algorithm and neural networks in literature review
- Back up some points made within the literature review

Progress:

- Began section regarding algorithms within the literature review

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson **Supervisor:** Simon Powers

Date: 26/10/2017 **Last diary date:** 20/10/2017

Objectives:

- Add a conclusion to literature review
- Finish search strategy section in literature review
- Begin implementation of k-nn algorithm

Progress:

- Finished literature review section on algorithms
- Added a section in literature review about research contribution

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson | **Supervisor:** Simon Powers

Date: 02/11/2017 | **Last diary date:** 26/10/2017

Objectives:

- Finish first draft of literature review
- Create the outline for the entire dissertation
- Update project schedule Gantt Chart

Progress:

- No progress was made this week as I had two coursework deadlines at the conclusion of the week

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson **Supervisor:** Simon Powers

Date: 09/11/2017 **Last diary date:** 02/11/2017

Objectives:

- At some point in the future update my literature review using all of the feedback within the interim report this includes
  - More up to date references
  - Justifications for different algorithms and machine learning in general
  - Switching datasets from KDD 99 to a more recent/amended one
- Begin work on a k-nn classifier
- General research on python machine learning and GUI libraries

Progress:

- Finished first draft of the literature review
- Updated Gantt Chart to better reflect current project schedule.
- Created outline for dissertation

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson **Supervisor:** Simon Powers

Date: 16/11/2017 **Last diary date:** 09/11/2017

Objectives:

- Improve k-nn classifier by finding alternative to onehot encoding.
- Begin creating the GUI

Progress:

- Set up Github repository
- Researched and implemented preparing a dataset for processing
- Implemented a basic k-nn classifier
- Researched different GUI options

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson

Supervisor: Simon Powers

Date: 23/11/2017

Last diary date: 16/11/2017

Objectives:

- Continue to attempt to improve knn performance
- Continue work on creating GUI
- Implement multi-layer perceptron classifier

Progress:

- Began creating a simple GUI to start learning PyQt
- Began writing simple unit tests

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson

Supervisor: Simon Powers

Date: 09/01/2018

Last diary date: 23/11/2017

Objectives:

- Implement a multi-layer perceptron classifier
- Continue creation of a GUI

Progress:

- No progress was made over the holiday period

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson

Supervisor: Simon Powers

Date: 16/01/2018

Last diary date: 09/01/2018

Objectives:

- Implement a support vector machine instead of negative selection classifier due to time constraints
- Continue work on the GUI
- Research feature selection and improving classifier performance

Progress:

- Created a simple multi-layer perceptron classifier
- Implemented a large section of the GUI
  - Dataset selection
  - Field type categorising
  - Custom classifier selection
  - Basic window layouts

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson **Supervisor:** Simon Powers

Date: 23/01/2018 **Last diary date:** 16/01/2018

Objectives:

- Research statistical significance testing
- Implement ability to run and compare several classifiers
- finish graphing and displaying results

Progress:

- Implemented support vector machine classifier
- Added basic graph to GUI
- Added K fold cross validation
- User creation of template classifiers
- Linked back end computation to the GUI
- General bug fixes

Supervisor's Comments:

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Jack Anderson **Supervisor:** Simon Powers

Date: 30/01/2018 **Last diary date:** 23/01/2018

Objectives:

- Finish displaying results in GUI
- Implement ability to run classifiers several times and average results for stochastic classifiers such as MLP
- Begin gathering results

Progress:

- Added ability to run several classifier configurations at once
- Added graphing of classification results and selection of specific classes
- Fixed major bug regarding two stage classification results

Supervisor's Comments:

D MoSCoW Analysis

Table 5: MoSCoW Analysis of Requirements

Requirement		Priority			
ID	Title	Must	Should	Could	Wont
FR1	Select Classifiers	x			
FR2	Two Stage Classification	x			
FR3	Run Classifiers	x			
FR4	Create New Classifier			x	
FR5	Debug Output		x		
FR6	Multiple Classifier Configurations		x		
FR7	Aggregate Results By Attack	x			
FR8	Write Results to File		x		
FR9	Get Classification Results	x			
FR10	Graph Results		x		
FR11	Select Training Dataset	x			
FR12	Select Testing Dataset	x			
FR13	k-Fold Cross Validation	x			
FR14	Select Dataset Column Labels	x			
FR15	Select Dataset Attack Categories	x			
FR16	Categorise Dataset Fields	x			
FR17	One Hot Encoding	x			
FR18	Load Data	x			
FR19	k-Nearest Neighbour Classifier	x			
FR20	Multi-layer Perceptron Classifier	x			
FR21	Negative Selection Classifier			x	
FR22	Support Vector Machine Classifier	x			
FR23	Stochastic Classifier Averaging		x		
FR24	Feature Selection				x
QR1	Robustness	x			
QR2	Responsiveness			x	
QR3	Usability		x		
QR4	Maintainability	x			
QR4	Correctness	x			
QR5	Portability			x	

E Class Diagram

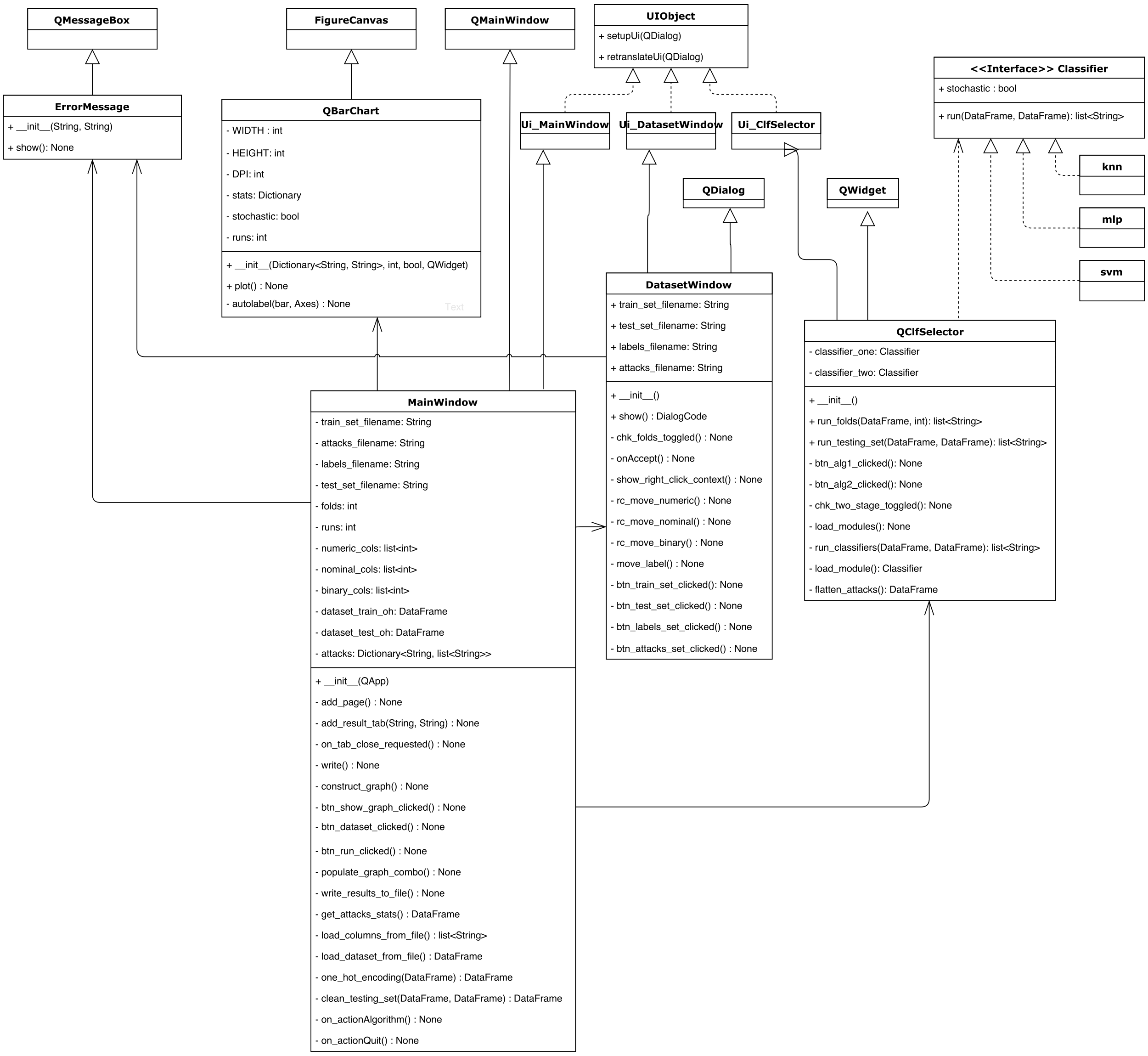

Figure 7: UML Class Diagram

F Use Case Diagram

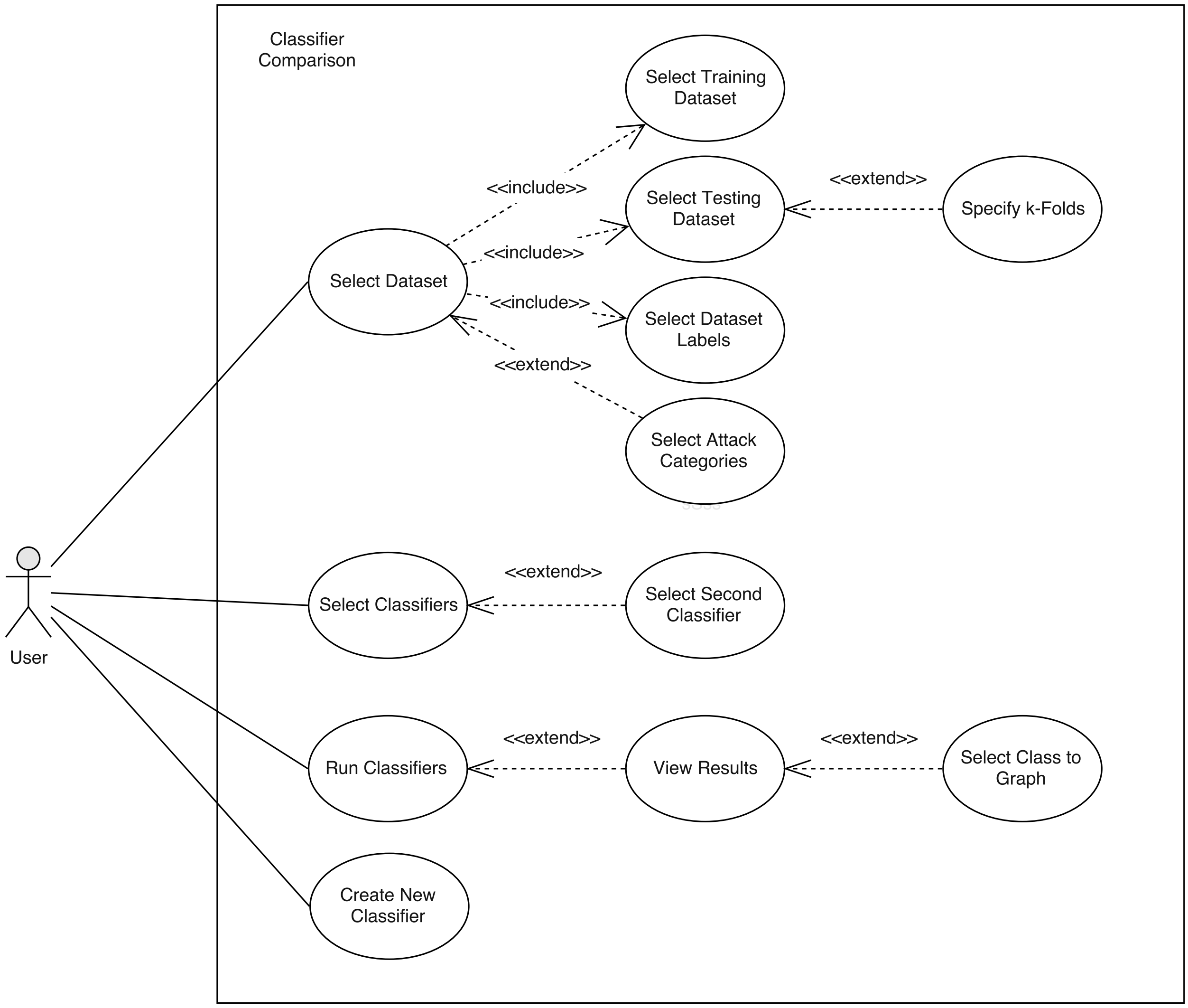

Figure 8: UML Use Case Diagram

G NSL-KDD Precision

Table 6: NSL-KDD Classifiers Precision Table

	dos	u2r	r2l	probe	normal
K-NN	0.91	0.17	0.69	0.9	0.99
MLP	0.94	0.38	0.79	0.98	0.99
SVM	0.93	0.64	0.83	0.99	0.99
K-NN/K-NN	0.9	0.18	0.73	0.91	0.99
K-NN/MLP	0.94	0.3	0.51	0.98	0.99
K-NN/SVM	0.93	0.58	0.64	0.96	0.99
MLP/K-NN	0.89	0.33	0.51	0.91	0.99
MLP/MLP	0.94	0.49	0.71	0.98	0.99
MLP/SVM	0.93	0.73	0.8	0.97	0.99
SVM/K-NN	0.9	0.21	0.44	0.91	0.99
SVM/MLP	0.94	0.37	0.49	0.98	0.99
SVM/SVM	0.93	0.65	0.83	0.96	0.99

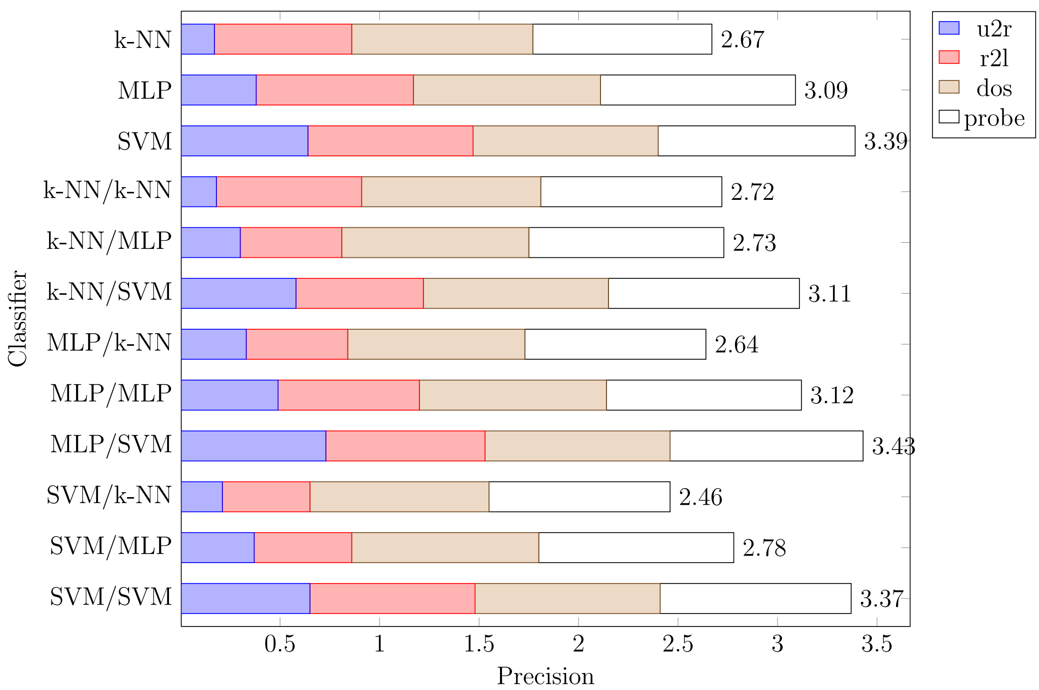

Figure 9: Classifiers Precision Graph

H NSL-KDD Recall

Table 7: NSL-KDD Classifiers Recall Table

	dos	u2r	r2l	probe	normal
K-NN	0.92	0.18	0.59	0.88	0.99
MLP	0.95	0.34	0.57	0.98	0.99
SVM	0.93	0.36	0.6	0.95	0.99
K-NN/K-NN	0.9	0.18	0.58	0.88	0.99
K-NN/MLP	0.94	0.19	0.57	0.98	0.99
K-NN/SVM	0.9	0.21	0.55	0.97	0.99
MLP/K-NN	0.89	0.28	0.53	0.88	0.99
MLP/MLP	0.94	0.44	0.65	0.98	0.99
MLP/SVM	0.92	0.34	0.61	0.97	0.99
SVM/K-NN	0.93	0.29	0.56	0.88	0.99
SVM/MLP	0.98	0.51	0.57	0.98	0.99
SVM/SVM	0.93	0.36	0.6	0.97	0.99

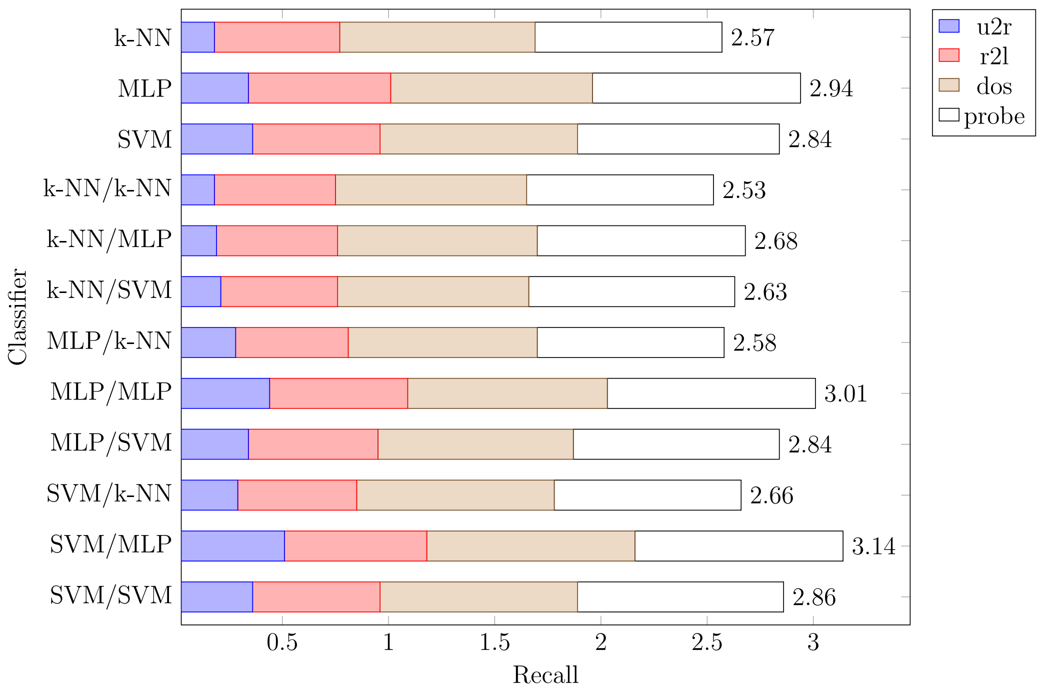

Figure 10: Classifiers Recall Graph

I NSL-KDD F1-Score

Table 8: NSL-KDD Classifiers f1-Score Table

	dos	u2r	r2l	probe	normal
K-NN	0.92	0.18	0.63	0.89	0.99
K-NN/K-NN	0.9	0.18	0.51	0.89	0.99
K-NN/MLP	0.94	0.23	0.54	0.98	0.99
K-NN/SVM	0.91	0.29	0.58	0.96	0.99
MLP	0.95	0.36	0.72	0.98	0.99
MLP/K-NN	0.89	0.24	0.51	0.89	0.99
MLP/MLP	0.94	0.46	0.67	0.98	0.99
MLP/SVM	0.93	0.45	0.68	0.97	0.99
SVM	0.93	0.45	0.69	0.97	0.99
SVM/K-NN	0.91	0.22	0.48	0.89	0.99
SVM/MLP	0.96	0.43	0.54	0.98	0.99
SVM/SVM	0.93	0.45	0.69	0.96	0.99

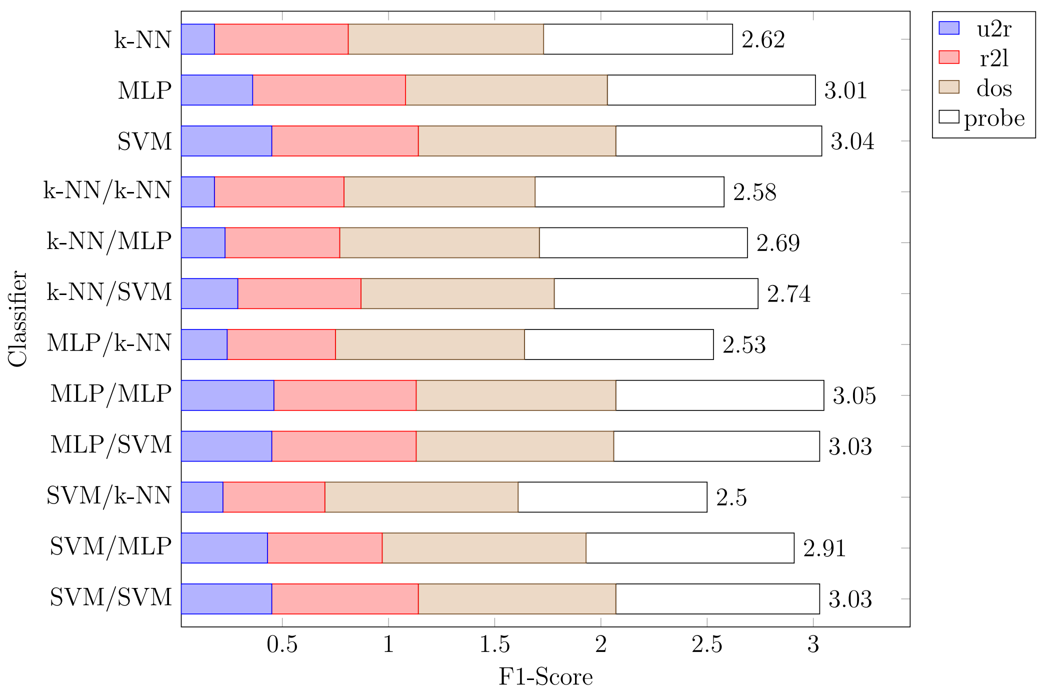

Figure 11: Classifiers f1-Score Graph

J UNSW-NB15 Precision

Table 9: UNSW-NB15 Classifiers Precision Table

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Recon	Shellcode	Worms	Normal
K-NN	0.1	0.08	0.22	0.23	0.19	0.95	0.7	0.18	0	0.73
MLP	0.13	0.13	0.38	0.62	0.57	0.98	0.71	0.3	0.49	0.93
SVM	0.06	0	0.23	0.59	0.6	0.99	0.69	0.44	0.1	0.9
K-NN/K-NN	0.1	0.06	0.19	0.24	0.19	0.95	0.62	0.14	0	0.73
K-NN/MLP	0.01	0.13	0.34	0.4	0.25	0.99	0.6	0.16	0.27	0.73
K-NN/SVM	0.01	0	0.17	0.39	0.24	0.99	0.64	0.31	0	0.73
MLP/K-NN	0.1	0.06	0.21	0.46	0.25	0.95	0.63	0.16	0	0.94
MLP/MLP	0.11	0.13	0.38	0.61	0.52	0.98	0.68	0.32	0.53	0.94
MLP/SVM	0.06	0	0.23	0.58	0.59	0.98	0.68	0.44	1	0.93
SVM/K-NN	0.1	0.06	0.21	0.45	0.22	0.95	0.63	0.15	0	0.92
SVM/MLP	0.11	0.14	0.38	0.61	0.47	0.98	0.65	0.28	0.53	0.92
SVM/SVM	0.06	0	0.23	0.57	0.53	0.98	0.66	0.39	1	0.92

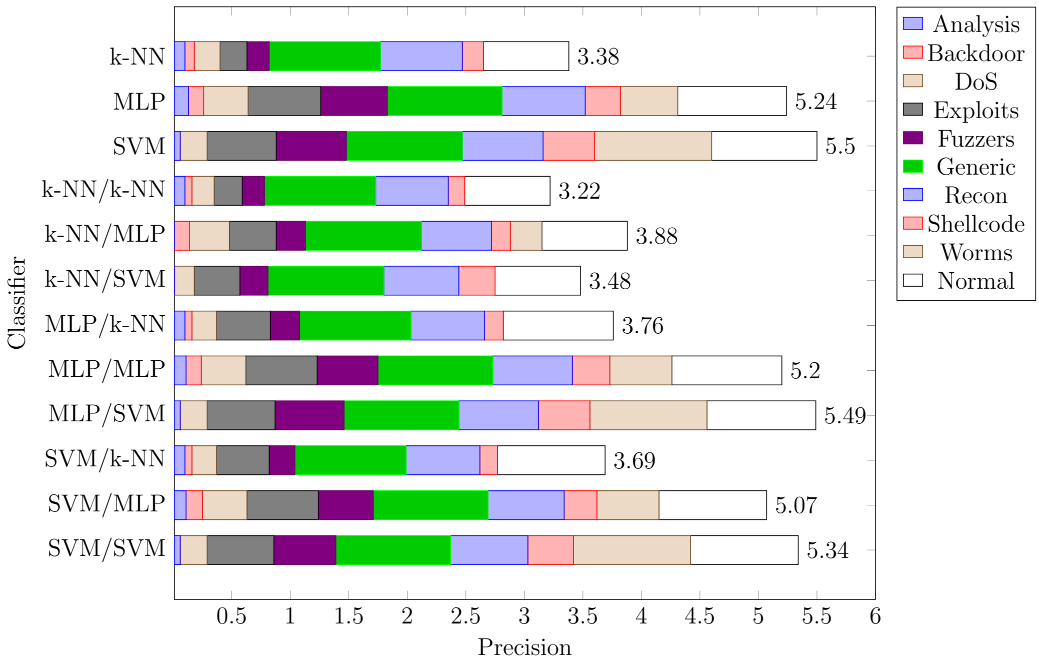

Figure 12: Classifiers Precision Graph

K UNSW-NB15 Recall

Table 10: UNSW-NB15 Classifier Recall Table

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Recon	Shellcode	Worms	Normal
k-NN	0.33	0.08	0.24	0.26	0.14	0.96	0.36	0.03	0	0.72
MLP	0.01	0.02	0.15	0.78	0.6	0.97	0.69	0.21	0.09	0.95
SVM	0	0	0.13	0.82	0.43	0.96	0.6	0.23	0.05	0.92
k-NN/k-NN	0.35	0.08	0.19	0.26	0.14	0.96	0.37	0.03	0	0.72
k-NN/MLP	0.01	0.03	0.13	0.43	0.4	0.96	0.52	0.14	0.03	0.72
k-NN/SVM	0	0	0.11	0.48	0.3	0.96	0.51	0.13	0	0.72
MLP/k-NN	0.35	0.08	0.21	0.51	0.2	0.96	0.36	0.05	0	0.94
MLP/MLP	0.01	0.02	0.16	0.79	0.61	0.97	0.65	0.26	0.1	0.93
MLP/SVM	0	0	0.13	0.83	0.5	0.97	0.62	0.24	0.05	0.93
SVM/k-NN	0.35	0.08	0.21	0.5	0.17	0.96	0.36	0.05	0	0.91
SVM/MLP	0.01	0.02	0.15	0.78	0.56	0.97	0.64	0.26	0.1	0.91
SVM/SVM	0	0	0.13	0.83	0.45	0.97	0.6	0.24	0.05	0.91

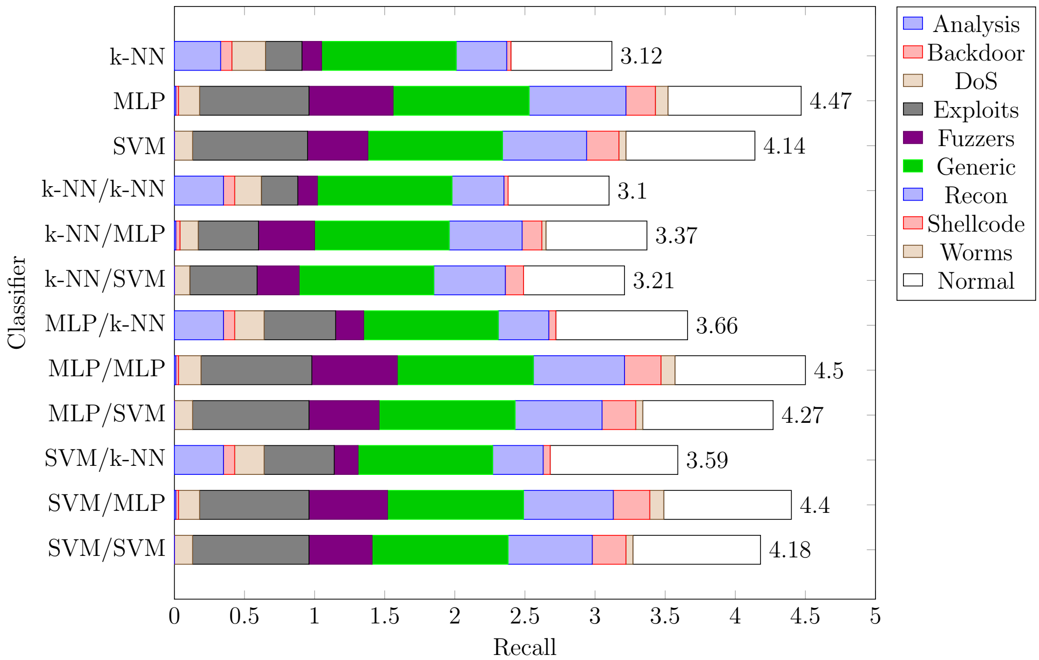

Figure 13: UNSW-NB15 Classifiers Recall Graph

Table 11: UNSW-NB15 Classifiers f1-Score Table

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Recon	Shellcode	Worms	Normal
k-NN	0.16	0.08	0.23	0.25	0.16	0.95	0.48	0.05	0	0.73
MLP	0.02	0.03	0.21	0.69	0.58	0.97	0.7	0.25	0.15	0.94
SVM	0	0	0.17	0.69	0.5	0.97	0.64	0.3	0.09	0.92
k-NN/k-NN	0.15	0.07	0.19	0.25	0.16	0.96	0.46	0.05	0	0.73
k-NN/MLP	0.01	0.05	0.19	0.42	0.31	0.98	0.56	0.15	0.05	0.73
k-NN/SVM	0	0	0.13	0.43	0.26	0.98	0.57	0.18	0	0.73
MLP/k-NN	0.15	0.07	0.21	0.48	0.22	0.96	0.46	0.08	0	0.93
MLP/MLP	0.01	0.03	0.23	0.69	0.56	0.98	0.66	0.29	0.16	0.93
MLP/SVM	0	0	0.17	0.68	0.54	0.97	0.65	0.31	0.09	0.93
SVM/k-NN	0.15	0.07	0.21	0.48	0.19	0.96	0.45	0.08	0	0.91
SVM/MLP	0.01	0.04	0.22	0.68	0.51	0.98	0.64	0.27	0.17	0.91
SVM/SVM	0	0	0.17	0.68	0.49	0.97	0.63	0.3	0.09	0.91

L UNSW-NB15 f1-Score

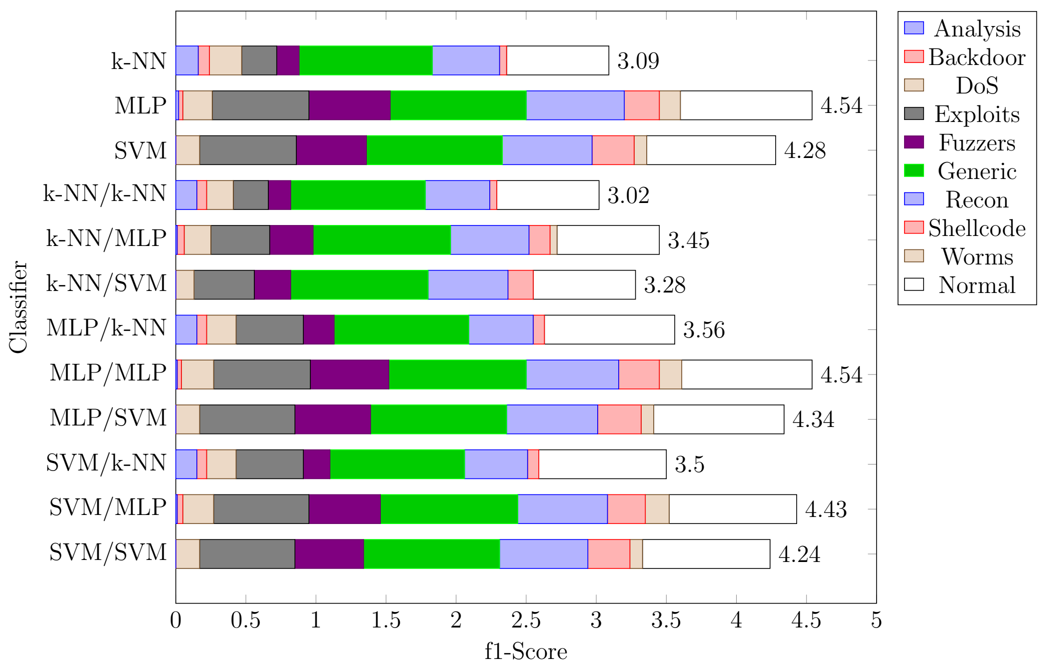

Figure 14: UNSW-NB15 Classifiers f1-Score Graph