Spring 2019

A Machine Learning Approach to Network Intrusion Detection System Using K Nearest Neighbor and Random Forest

Ilemona S. Atawodi
University of Southern Mississippi

A MACHINE LEARNING APPROACH TO NETWORK INTRUSION DETECTION SYSTEM USING K NEAREST NEIGHBOR AND RANDOM FOREST

by

Ilemona Solomon Atawodi

A Thesis
Submitted to the Graduate School,
the College of Arts and Sciences
and the School of Computing Sciences and Computer Engineering
at The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Master of Science

Approved by:

Dr. Zhaoxian Zhou, Committee Chair
Dr. Chaoyang Joe Zhang
Dr. Kuo Lane Chen

_____	_____	_____
Dr. Zhaoxian Zhou	Dr. Andrew H. Sung	Dr. Karen S. Coats
Committee Chair	Director of School	Dean of the Graduate School

May 2019

ABSTRACT

The evolving area of cybersecurity presents a dynamic battlefield for cyber criminals and security experts. Intrusions have now become a major concern in the cyberspace. Different methods are employed in tackling these threats, but there has been a need now more than ever to updating the traditional methods from rudimentary approaches such as manually updated blacklists and whitelists. Another method involves manually creating rules, this is usually one of the most common methods to date.

A lot of similar research that involves incorporating machine learning and artificial intelligence into both host and network-based intrusion systems recently. Doing this originally presented problems of low accuracy, but the growth in the area of machine learning over the last decade has led to vast improvements in machine learning algorithms and their requirements.

This research applies k nearest neighbours with 10-fold cross validation and random forest machine learning algorithms to a network-based intrusion detection system in order to improve the accuracy of the intrusion detection system. This project focused on specific feature selection improve the increase the detection accuracy using the K-fold cross validation algorithm on the random forest algorithm on approximately 126,000 samples of the NSL-KDD dataset.

ACKNOWLEDGMENTS

I would like to express my appreciation to my supervisor, Dr. Zhaoxian Zhou for everything I have learnt under his mentorship during the course of my master's degree especially for everything I have learnt on the research front.

I would also like to thank my committee members Dr. Chaoyang Joe Zhang whom I have taken a lot of classes under and Dr. Kuo Lane Chen who has given me the occasional career advice.

I am thankful to Mr. Gabriel Idakwo from the Bioinformatics lab for all the help I received on this research, and Ms. Ojonoka Atawodi for assisting me with proof reading while I wrote this thesis.

I also want to acknowledge my lab mates Sarbagya Shakya, Pawan Sharma, Bailey Lutrell , and Zhuo He

DEDICATION

I would like to dedicate this research to my parents Prof. and Dr. Atawodi, Mr. and Mrs. Idakwo, Mr. and Mrs. Atawodi-Alhassan, Ms. Ojonoka Atawodi, Ms. Chidinma Chikere, Yusuf Atawodi, Yahaya Atawodi, Abdulsalam Atawodi, Mrs. Aisha Alidu, Mr. and Mrs. Richard Alidu and Specially to Ms. Karen Eberechukwu.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ABBREVIATIONS

APWG	Anti-Phishing working group
URL	Universal Resource Locator
DOS	Denial of Service
R2L	Remote to User
U2R	User to Root
IDS	Intrusion Detection system
SVM	Support Vector Machine
KNN	K Nearest Neighbor
I/O	Input/Output
ROC	Reciever Operating Characteristic
CV	Cross Validation

CHAPTER I – INTRODUCTION

Cybersecurity is a growing problem in modern times because of the rapid growth and technological advancement. The internet provides all knowledge that has been accumulated by man and with the advent of mobile computing at every person's finger tips, cyberattacks and cyber crimes have become all too popular. A report from the antiphishing working group has shown that about 227,000 malware detections occur daily which is linked to over 20 million new malwares daily. Malwares can simply be defined as a computer program created to cause harm on a computer system (Kaspersky 2017).

There has been a straight forward method for dealing with malware in the past, but over the past two decades there has been an evolution in cyber attacks and how exploits are carried out, as such cybersecurity techniques are also undergoing an evolution into more intelligent approaches.

The main problem that has risen from the growth of technology and the internet is the amount of skill required to perform an attack on an unsuspecting target computer. Automated scripts and sophisticated programs that can bypass and evade security measures are readily available for anyone who wishes to perform an attack, and attacks from low skilled cyber criminals has been on the rise (Aliyev, 2010).

A 2016 report from the APWG showed that billions of US Dollars were lost due to phishing attacks, and 42.71% of these attacks were targeted at the retail industry. Such a large scale of attack towards business infrastructure in a country can greatly cripple the growth of businesses, in this report it was also shown that The United States hosted the largest number of phishing websites and china was the most affected by phishing.

This has motivated research into an improvement of the cybersecurity applications to mitigate loss of personal data and reduce the damages caused by cyber attacks because a properly coordinated cyber attack can cause extensive damage to a business. The traditional methods that exist in place cannot keep up with the rapid innovations that happing in the cyber crime space. An example of a traditional method for mitigating phishing attacks is the use of blacklists. A blacklist is a curated list of harmful URLs that are curatted and updated by a security company such as Avast. A network blocks all URLs that exist on the blacklist and allow all other URLs to flow through the network. By 2016, there were 300,000 unique phishing websites reported monthly, this presents a problem for the security company creating the blacklist in two phases, firstly; a phising website has to successfully attack a system before it is flagged as illegitimate and blacklisted because all phishing are created to mimic legitimate websites, secondly; the sheer number of new phishing websites are very difficult to keep up with.

The most efficient way to tackle this growing problem involves the use of machine learning algorithms to become able to detect attacks before they attack a legitimate system. A lot of network logs already exists from past attacks, and these can be fed into the algorithm to train it to be able recognize attacks and alert a prevention response in the security system.

This aim of this research is to improve the detection rate of a combination of cyberattacks by leveraging feature selection and evaluating what combination of features would provide the best classification accuracy in the K-fold cross validation method and random forest algorithm.

CHAPTER II – THEORETICAL BACKGROUND

In this chapter a background of definitions and explanations that are necessary to understand intrusion detection and related tools are laid out. Attack types and intrusion detection systems, and related works to the topic are discussed in this chapter.

2.1 Attack Types

In order to understand the reason for the multiple kinds of attacks, it is necessary to know that the target platform and the intention behind the attack are the main deciding factors behind the kind of attack a system would be exposed to at any given time, for example a phishing attack (attack type) would be carried out on the web or through email (attack platform) and is intended to steal personal information like passwords and credit card details. Some other attacks are intended to simply disrupt network flow and constitute nuisance or to advertise merchandise.

In this project 21 unique attacks were studied. They broadly fall under three classes, other classes beyond these three exist but the dataset used limited classes to the first three classes mentioned.

Some of these attack classes are as follows:

- **Denial of Service Attacks (DOS).** The attacker occupies the system resources (memory and processing components of the machine) and engages them continuously to prevent users with legitimate access from making use of the network or machine (Abliz, 2011).
    - **Attack types.** Back, Land, Neptune, Pod, Smurf
- **Remote to User Attacks (R2L).** The attacker sends packets over the web (remotely) to expose vulnerabilities in a network or machine and tries to

exploit the privileges of a system that he has no physical access to (S. Paliwal, 2012).

    - **Attack types.** FTP_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster

- **User to Root (U2R).** This is usually performed by a legitimate user or an attacker who has gained access to a network or system with low level or standard access and privileges. The attack here involves the attacker illegally trying to elevate their user privileges and gaining superuser privileges by seeking out vulnerabilities in the system. This is sometimes due to negligence on the part of the system administrator such as not changing the default admin login information on new equipment (S. Paliwal, 2012).
    - **Attack types.** Buffer_overload, Loadmodule, Perl, Rootkit
- **Probe.** A probe attack simply involves the attacker scanning the system for vulnerabilities to exploit. Such vulnerabilities include open ports, back doors and poor passwords (V. Shmatikov, 2007). This is an easy attack type for low level attackers who simply use hacking software created by others, these type of attackers are known as script kiddies.
    - **Attack types.** IPsweep, Nmap, Satan
- **Phishing Attacks.** Phishing attacks are performed by creating clones of legitimate websites in order to trick users of a legitimate website to enter their login, financial or private details such as their social security number. This is an attack type that exploits multiple vulnerabilities in a single

attack such as the mistakes in a URL, most users would not notice if the popular search engine ‘www.google.com’ was replaced with ‘www.goggle.com’. Similar URLs and appearance of the website are utilized in tandem to fool the targets. These URLs could be attached in an email and would be clicked innocently.

- **Attack types.** Spear phishing, Clone Phishing, Whaling

Table 2.1 Attack class and types observed

Attack Class	Attack Name
Denial of Service (DOS)	Back
	Land
	Neptune
	Pod
	Smurf
	Teardrop
Remote to User Attacks (R2L)	Ftp_write
	Guess_passwd
	Imap
	Multihop
	Phf
	Spy
	Warezclient
	Warezmaster
User to Root (U2R)	Buffer_overflow
	Loadmodule
	Perl
	Rootkit
Probe	Ipsweep
	Nmap
	Satan

2.2 Types of Intrusion Detection Systems

Intrusion detection systems are methods and applications created to assess and protect computers, networks, programs and data from attacks, unauthorized access, illegal read/write and deletion/corruption. These can be divided into host and network based depending on where and how it is deployed. Intrusion detection systems should not be mistaken for other security measures that may exist in a network or system such as firewalls and antiviruses. Intrusion detection systems works in combination to these as they are more closely related to detecting and raising alarms.

Intrusions can happen both internally or externally due to these some system setups operate a hybrid intrusion detection setup.

Both host-based and network-based intrusion systems detect intrusions using one of two methods or some times a combination of both, these are;

2.2.1 Host Based

This kind of intrusion detection system is primarily used to monitor the internal environment of a host; resources, filesystem and applications of the host. A host based system monitors both the state and behavior and state of its host. Due to this it can detect unusual behavior of programs and files (M. Gupta, 2015).

Even though host based intrusion detection systems were primarily created for internal monitoring there are some variations of the host based intrusion detection system that can also detect network intrusions (A. P. Singh, 2016).

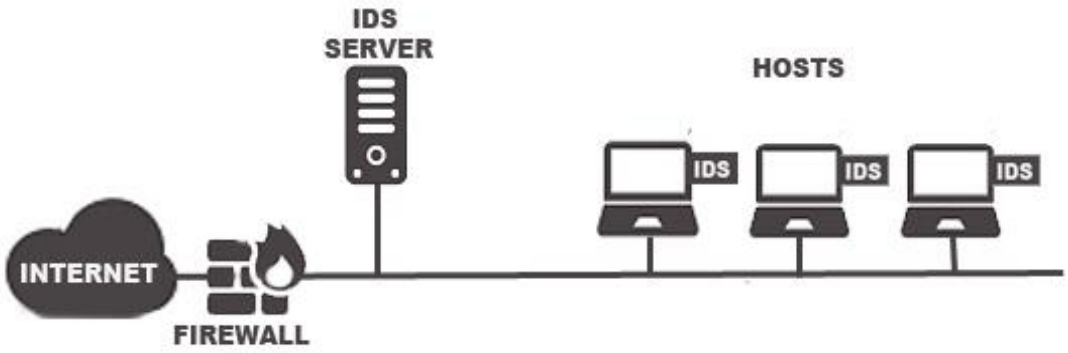


Figure 2.1 Setup of a Host Based intrusion detection system

Genetic algorithm performance with different selection strategiesin solving tsp by J. G. Noraini Mohd Razali (WCE, 2011)

.

2.2.2 **Network Based**

The main difference between the network based and the host based is how it is deployed. It exists as part of the network to detect intrusion attempts on the network. The network based intrusion detection system analyzes inbound and outbound traffic for patterns outside the expected behavior (J. G. Noraini, 2011).

It is good to note that a network based intrusion detection system does not replace a firewall in a system.

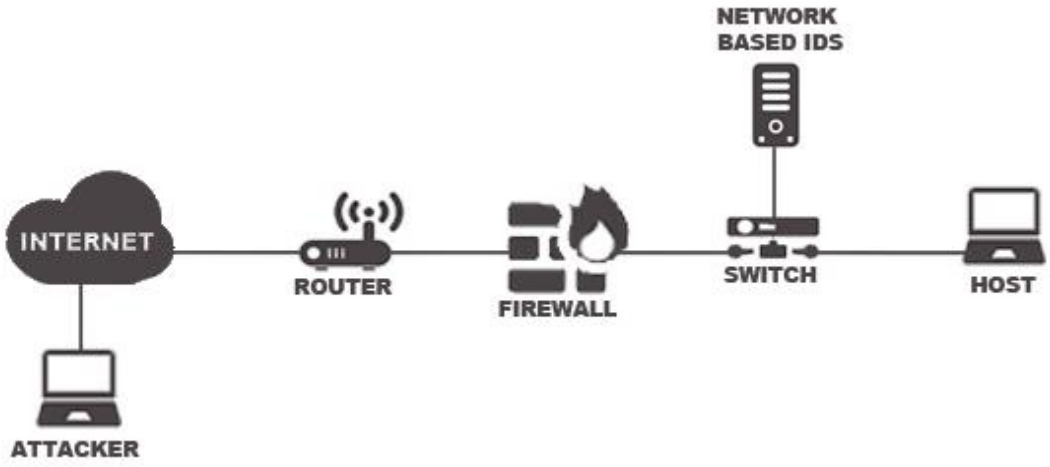


Figure 2.2 Setup of a Network based Intrusion detection system

Genetic algorithm performance with different selection strategiesin solving tsp by J. G. Noraini Mohd Razali (WCE, 2011)

2.3 Types of Intrusion Detection Approach

Intrusion detection systems approach the problem of detection in two basic ways regardless of the type, host based or network based. The approach is either signature based or anomaly based. In other cases its possible to find some systems operating a hybrid system.

2.3.1 **Misuse Based (Signature Based)**

Attacks generally have signatures such as metadata and file fingerprints (MD5 or SHA1 hash), these signatures can be used to determine whether an attack is taking place or the observed system behavior is normal. This is accomplished by comparing the signature to the signatures of previous attacks (A. P. Singh, 2016). This method of detection is very effective as the rate of false positives in detection is low. The caveat here is that the system needs to be updated frequently otherwise, attacks whose signatures don't exist in the catalog would likely not be detected. This means that signature based systems are not effective against zero day attacks (novel attacks that have not yet been documented)

2.3.2 **Anomaly Based**

Anomaly based detection searches through networks for abnormal behavior, it suspects any network behavior that deviates from the regular network operating baseline as an anomaly. This method is preferred under certain conditions to the signature based system because it has a much higher chance of detecting new (zero day) attacks. Every network has a unique baseline as such its difficult for attackers to successfully pull of an attack that goes undetected by the anomaly system. Despite the stated advantages of this

method, it has a high propensity for frequent false positives since some of the detected changes are legitimate changes to the system.

Anomaly based detection system can be divided into statistical based and knowledge based.

The statistical method views anomaly detection from randomness while the knowledge based method involves capturing claimed behavior from network traffic instances and other relevant system data.

2.3.3 **Hybrid Based**

This method combines the signature and anomaly methods in order to maximize the advantages and minimize the drawbacks of both methods meaning detection rate of zero day attacks and new/unknown attacks increases, while managing to reduce the amount of false alarms. A survey by (A. Buczak) found that no system was purely signature or anomaly, intrusion detection systems are usually a deployed as a hybrid setup.

2.4 Machine Learning

Traditionally intrusion detection systems posses a high reliance on a human intervention to keep the system up to date. This reliance includes adding newly discovered phishing URLs to blacklists, adding new rules to rule based systems, creating exceptions and curation of whitelists. The growth of the internet and the sheer size of networks that exist and the rapid creation rate of zero day attacks its easy to see the short coming of a human based intrusion detection system (T. M. I. White Paper, 2012).

One way to combat this demerit is machine learning. First of all machine learning is something that improves with scale, a machine learning system is simply system that

relies on learning from past occurrences of attacks and attempted intrusions and learning the patterns that come along with these and how it differs from normal behavior [15]. Once a machine learning algorithm has discovered these patterns recognition and classification happens at a faster rate on a larger scale that any human-centric system can operate.

Machine learning algorithms are broadly grouped into three, these groups are simply due to how the algorithm is trained. These groups are:

- **Supervised Learning.** This algorithm class requires training data that has already been labelled. Supervised learning algorithms are designed for prediction. An example of a supervised learning algorithm used in intrusion detection is the support vector machine (SVM), and random forest. SVM is particularly used in networks based intrusion detection systems because of its computational practicality (A. Elike Hodo, 2017)
- **Semi-supervised.** Semi-supervised machine learning algorithms differs from the supervised model in its ability to use unlabeled data to train, i.e it is able to see patterns unassisted and create classifications. This ability is useful when large amounts of labelled training data is scarce or unavailable. An example of a semi-supervised machine learning algorithm is the semi-supervised support vector machine. This is also used in network based intrusion detection machines (A. Elike Hodo, 2017).
- **Unsupervised learning.** This is also a classification algorithm, it is trained using unlabeled data

2.4.1 **Machine learning process**

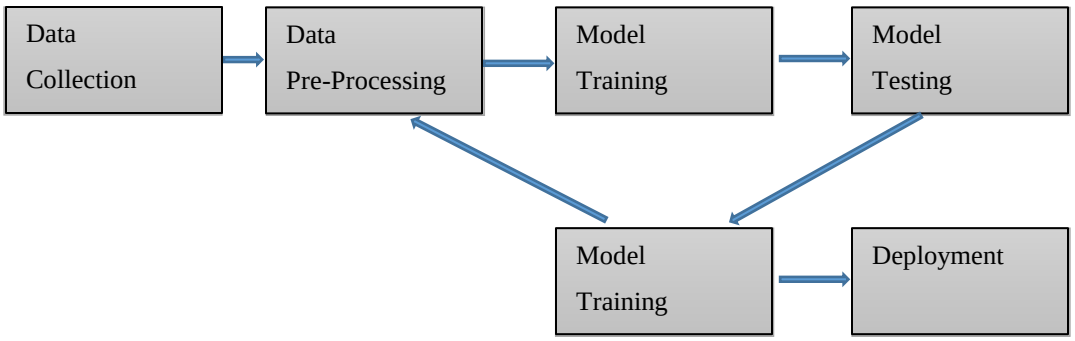


Figure 2.3 Basic workflow of a machine learning algorithm

Machine Learning Methods for Malware Detection and Classification by Catheryna Chumachenko (University of Applied Sciences, Bachelor Thesis 2017).

1. **Data collection.** Depending on the project relevant data specific to research is collected and stored in memory (Chumachenko, 2017).
2. **Data pre-processing.** At this point the data collected is arranged, and transformed into a format that can be fed into the machine learning algorithm. In most cases this is stored as a table or numpy arrays. Feature extraction also occurs at this point (not all information from the data collection stage is relevant to the experiment, as such some things are ignored entirely). The pre-processed data is then split into training and test datasets.
    a. **Feature extraction.** Feature extraction is a pre-processing task that involves selecting specific relevant features to create the training and testing datasets that would be fed into the algorithm. This achieves a couple of aims; it reduces the probability of overfitting, it makes interpretation easier and it improves the chance for generalization. Improper feature extraction can lead to the model running longer than

necessary as it has to go through more data than it needs to in order to train and test (Oscar Jimenez-del-Toro, 2017).

3. **Training.** At this stage the training dataset from the data pre-processing stage is fed into the chosen machine learning algorithm and a model is built. This stage may be done once or may be repeated, this depends on the algorithm.
4. **Testing.** Once the model has been built, the test dataset from the data pre-processing stage is fed into to model in exactly the same format as the training dataset. The classification or prediction accuracy is taken. The closer to a hundered percent the accuracy is, the better the model. Of course at this stage its statistically impossible to build a model with an accuracy of one hundred percent, as the size of the data grows and adjustments are made to the model, steps 2 through 4 are repeated.
5. **Deployment.** At this point, the model with the best prediction or classification, depending on need is chosen based of comparison of results. (Chumachenko, 2017)

2.4.2 **Machine learning: Classification and Regression**

Classification and regression fall under supervised learning where data is labelled and the outcomes are known and can be mapped accordingly. This means that the model is created from datasets where the outcome is known in both cases.

- **Classification.** A classification problem is one where the model tries to find a category for an element in the dataset. For example if certain characteristics (features) are presented based on the similarity to a class in the training dataset what category should an item fall into (google developers, 2019).

- **Regression.** Regression problems are those where the model attempts to predict what a missing in a dataset. Regression problems tend to be used more on continuous data as opposed to classification tasks (google developers, 2019).

2.4.3 **Machine Learning Algorithms**

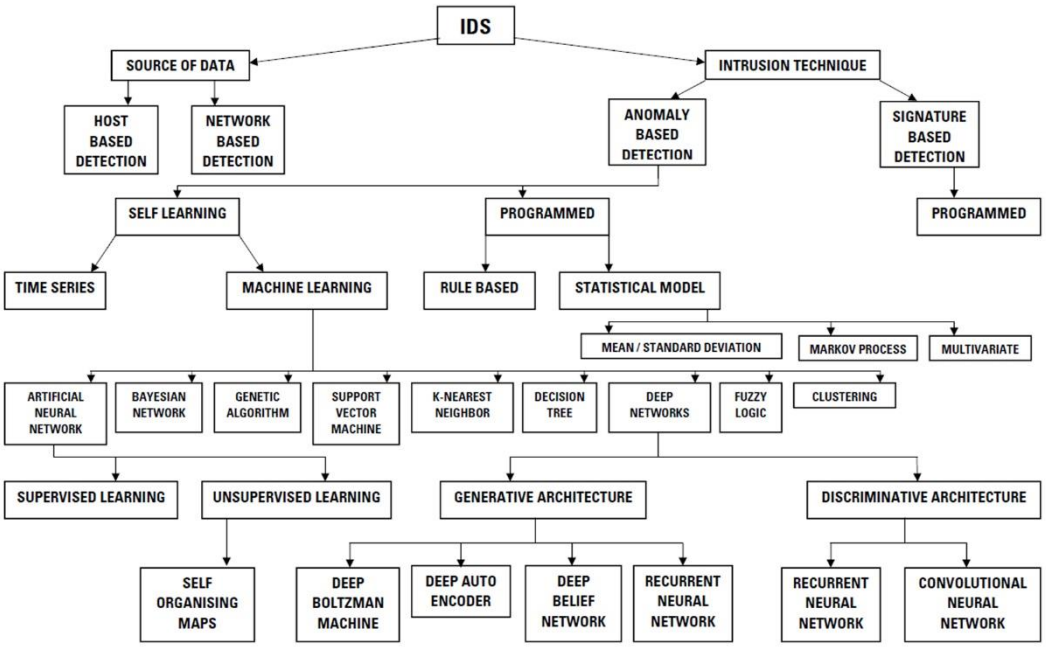


Figure 2.4 Full chart showing categories of intrusion detection

Shallow and Deep Networks Intrusion Detection System: A Taxonomyand Survey by Xavier J. A. Bellekens and Andrew Hamilton and Christos Tachtatzis and Robert C. Atkinson (ArXiv, 2017).

CHAPTER III – METHODS

The main aim of this project involved selecting a good combination of features to increase the classification accuracy of the K-Nearest neighbor algorithm on the NSL-KDD+ dataset. A couple of combinations were iterated through until the combination with the highest classification accuracy was found (Özgür A).

3.1 Dataset

The NSL-KDD data set was used for this project. It is a lighter version of the DARPA 99 dataset and it has 42 distinct features; the target classes of this dataset can be viewed in two ways.

- **As Binary class**. This is ideal for a dataset like the NSL-KDD dataset with multiple features, in order to keep the complexity in check. Its binary as the target is simply ‘1 or 0’. ‘0’ for normal network activity, ‘1’ for attacks. This generalizes all the different kinds of attack to the target value of ‘1’.

  The targets for this projects follows this scheme.

  - **As Multi class.** For multi class targets, rather than have a target response of binary values, you would have multiple target responses. In this dataset there are 22 distinct classes of attacks in the dataset, as such multiclass classification would mean the model would attempt to classify out of 22 possible target responses. During the experiment it was observed quickly from the results of the initial train-test process on the algorithm used that its not ideal to use the dataset for multiclass classification.

3.1.1 **Features of the NSL-KDD dataset**

Table 3.1 Feature names and data types

Feature name	**Data type**
duration	continuous
src_bytes	continuous
dst_bytes	continuous
land	continuous
wrong_fragment	continuous
urgent	continuous
hot	continuous
num_failed_logins	continuous
logged_in	continuous
num_compromised	continuous
root_shell	continuous
su_attempted	continuous
num_root	continuous
num_file_creations	continuous
num_shells	continuous
num_access_files	continuous
num_outbound_cmds	continuous
is_host_login	continuous
is_guest_login	continuous
count	continuous
srv_count	continuous
serror_rate	continuous
srv_serror_rate	continuous
rerror_rate	continuous
srv_rerror_rate	continuous
same_srv_rate	continuous
diff_srv_rate	continuous
srv_diff_host_rate	continuous
dst_host_count	continuous
dst_host_srv_count	continuous
dst_host_same_srv_rate	continuous
dst_host_diff_srv_rate	continuous
dst_host_same_src_port_rate	continuous
dst_host_srv_diff_host_rate	continuous
dst_host_serror_rate	continuous
dst_host_srv_serror_rate	continuous
dst_host_rerror_rate	continuous
dst_host_srv_rerror_rate	continuous

3.1.2 **K-nearest neighbors**

The K-Nearest (KNN) is a simple and accurate machine learning algorithm that that classifies instances based on the nearest training instance in a feature space. It can be used for both regression and classification. KNN is an algortithm that makes no assumptions about the structure of data and distribution which means it is a non-parametric algorithm. This is an advantage because real life data scarcely obeys theoretical rules. Technically since the training data set is stored by the KNN, learning is no longer a requirement (Chumachenko, 2017).

KNN works by classifying or predicting based on a fixed number (K) of training instances closest to the input instance. This means that for a chosen value of K, an input instance would be classified or predicted to belong to the same class as the closest number of K instances nearest to it

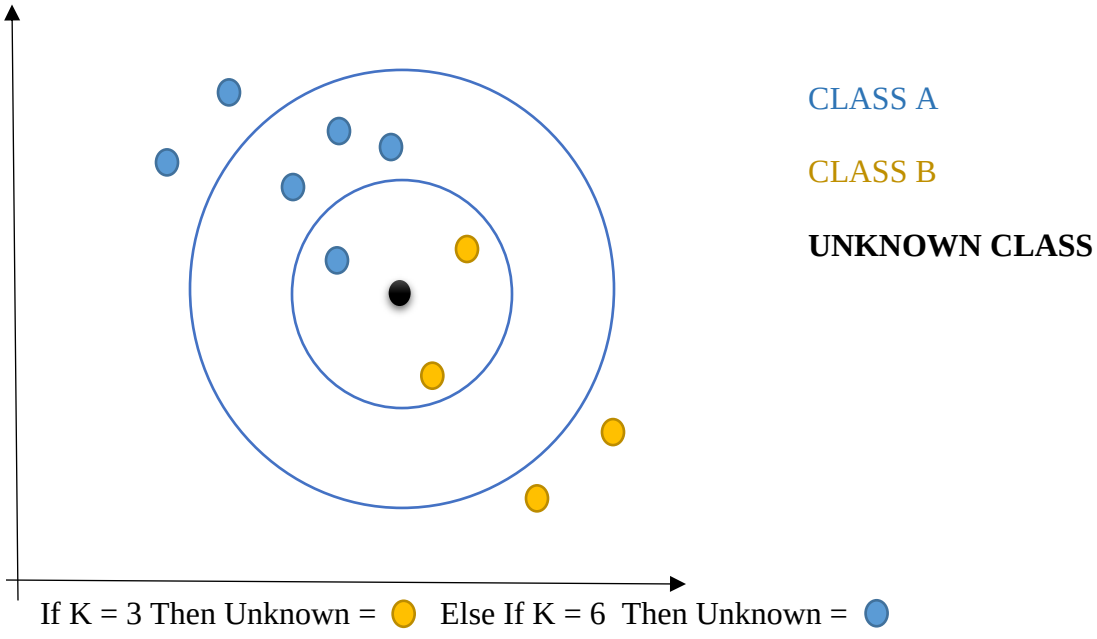


Figure 3.1 Basic example of KNN

Towardsdatascience.com (Italo Jose, 2018).

The distance from the unknown to the nearest neighbors is measured using different methods. The most popular one is Euclidean distance, other popular methods include Manhattan distance, Hamming distance, Minkowski distance (Chumachenko, 2017). Mathematically they can be represented by:

Euclidean distance = $d(\boldsymbol{p},\boldsymbol{q}) = d(\boldsymbol{q},\boldsymbol{p}) = \sqrt{q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$

$$= \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \quad [1]$$

Manhattan distance = $d_1(p,q) = ||p - q||_1 = \sum_{i=1}^{n} |p_i - q_i|$ [2]

Hamming distance = $d_{ij} \sum_{k=1}^{p} |x_{ik} - x_{jk}|$ [3]

Minkowski distance = $\sum_{i=1}^{n}(|x_i - y_i|^p)^{\frac{1}{p}}$ [4]

Each distance type is ideal under certain use cases. The Manhattan distance is best suited for KNN problems where the features are of different types. The Euclidean distance is suited to the opposite case where features are of the same type.

The value of 'k' is a very important part of the KNN algorithm because of prediction accuracy. Selecting a value for 'k' is a task that should not be taken for granted. Selecting small values for k will more than likely result in lower accuracy if the algorithm is training from a noisy dataset, and if the value for 'k' is too high the model may overfit resulting in low accuracy. An odd number 'k' should also be chosen if the number of classes are even to prevent ties in the majority vote.

3.1.3 **Cross Validation**

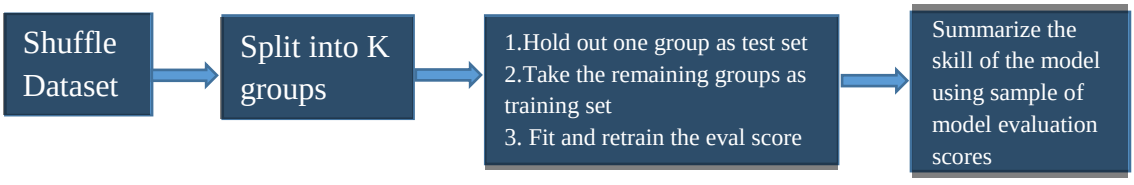


Figure 3.2 Cross Validation Flowchart

Most methods of evaluating accuracy suffer from the inability to predict the way they will perform on out of sample of new data. The K nearest neighbor algorithm also suffers from this drawback. Cross validation is a method of overcoming this. Based on the chart in figure 3.2, the dataset is first shuffled, then it is split into k groups (into a large group for training, a smaller group for testing) this is repeated k times until each part section of the split data has been used as training data and testing data (José, 2018).

There are three methods of cross validation:

1. **K-fold method.** The original dataset is shuffled into k samples of equal sizes. One of the k samples is held back to be used as a testing sample for the model. The remaining samples are used for training the model. This process is then repeated k times with each of the other k – 1 samples. Each run of the cross validation process has an error value. The average error is then calculated for all the k models. This reduces the variance and maintains accuracy amongst the models. The drawback of this method is the running time of this method (Saxena, 2016).
2. **Holdout method.** For the holdout method the sample is split in two, one part for training and the other for testing. The size of this split is usually such that the training set is larger than the testing set. The larger training sample is used to fit the model then the second split is then used to test model. This is the simplest cross validation method because its basically a single cross validation (Saxena, 2016). The advantage of the holdout method over the K-fold method is the running time.

3. **Leave one out method.** This method is similar to the k-fold method, with a very large value for k, generally as large as the sample universe. The model is trained on every value of the dataset except one, which is left out for testing. This reduces variance but is resource and time intensive (Saxena, 2016).

3.1.4 **Random Forest**

Random forest is a very popular machine learning algorithm. Its an ensemble learning algorithm, meaning that it creates many decision trees (this is the reason it is called a forrest) during the training phase of the algorithm and then produces the most popular output as it's classification. Its also used for regression or prediction problems.

One of the main advantages of the random forest is speed, classification happens really quickly (Ho, 1995).

Random forest creates a forest of independent subsets of the dataset. The best split is found by randomly selecting n variables at every individual node.

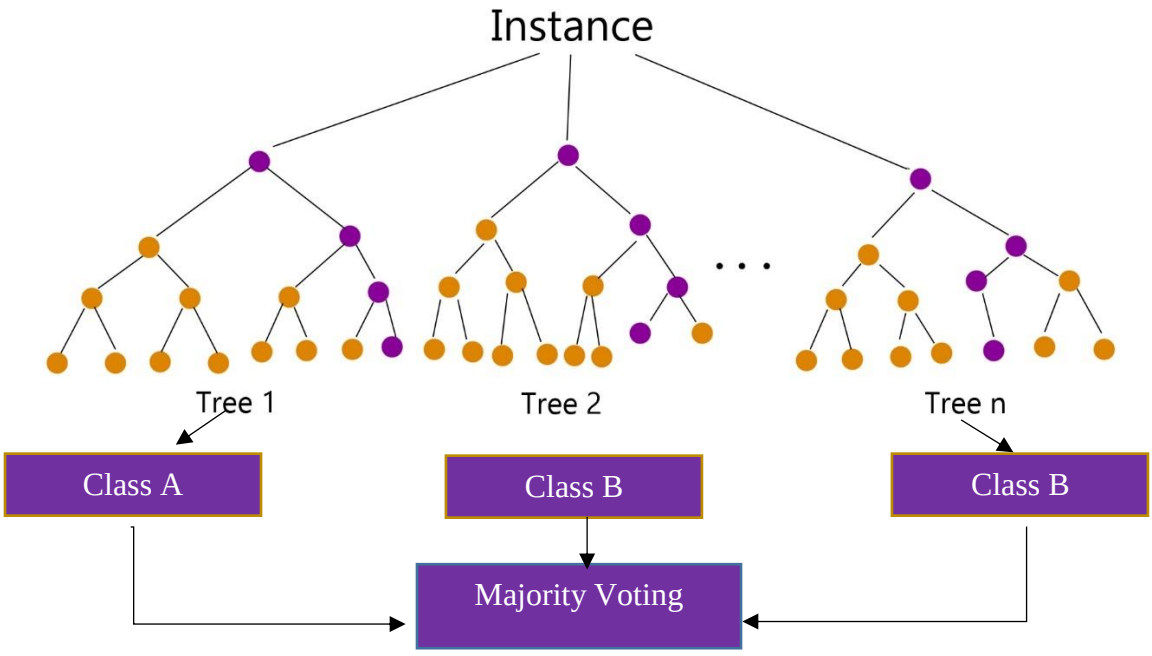


Figure 3.3 Flowchart Showing random forest operation (towards datascience)

3.2 Tools and Environment

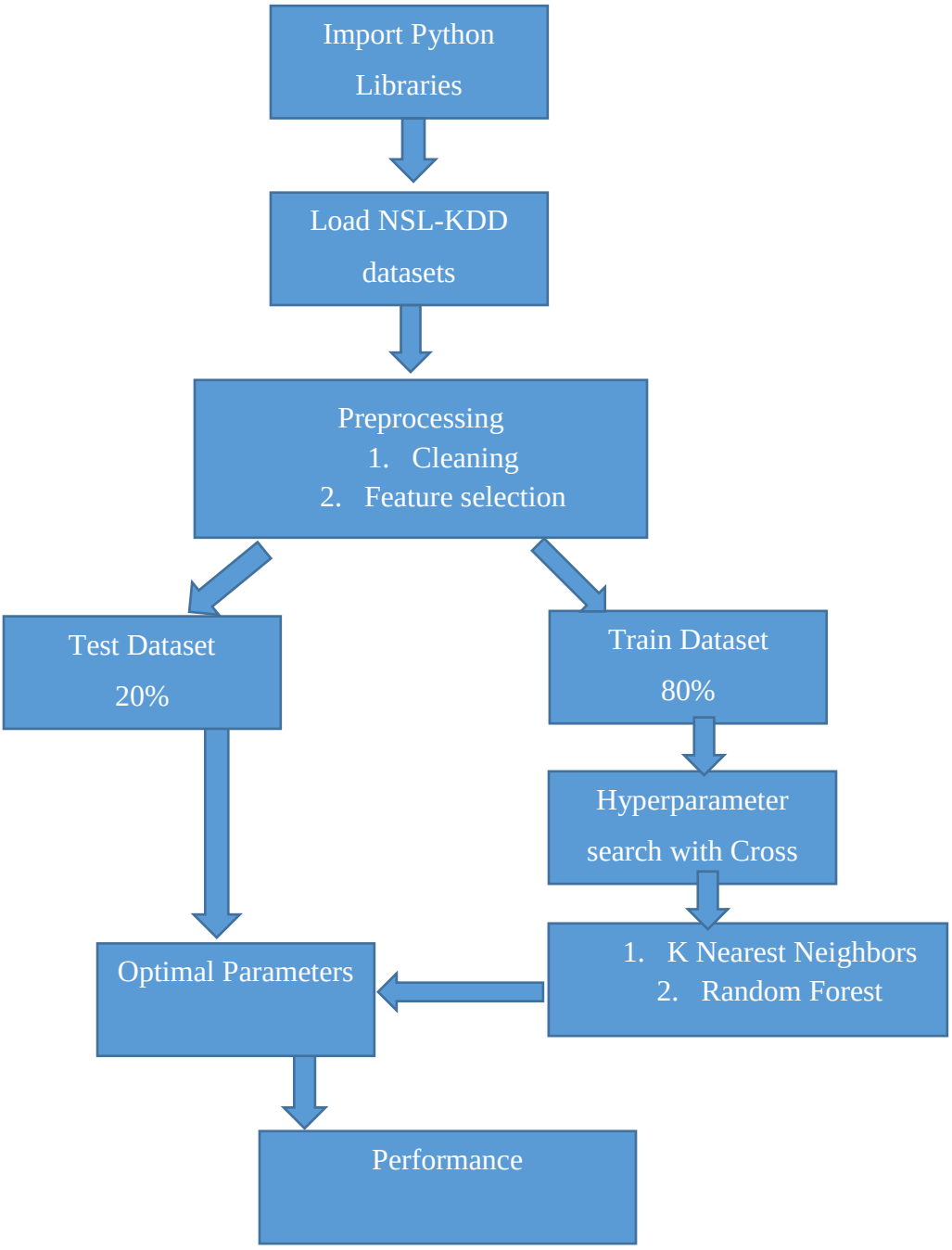


Figure 3.4 Flowchart showing method in the research

3.2.1 Anaconda

Anaconda is a python and R distribution for data science and machine learning applications. It's a free and open source software and contains over 1400 packages. The main advantage behind using anaconda is that, anaconda is like a central point for the libraries on would need for scientific computing. Any required library can be installed via the terminal.

3.2.2 Jupyter Notebook

The jupyter notebook is an environment that allows computations within the browser of the user's system. The jupyter notebook is a simply a JSON document that contains an ordered list of I/O cells which could contain code, mathematics, text, graphs/plots and other types of media.

The jupyter notebook allows the user to experiment with code in browser, without having to rerun the entire code each time the researcher wants to experiment on just a small block of the code. This allows for a lot of flexibility.

3.2.3 Scikit-learn

Scikit-learn is a free library for the scientific computation using python language. It contains various algorithms for regressiong, classification and clustering.

3.2.4 NumPy

Numpy is a python library that adds support for large multi-dimensional arrays and matrices. It also contains collections of high level mathematic functions to operates on arrays.

3.2.5 **Matplotlib**

This is a python and numpy plotting library. This allows for plotting various kinds of graphical representation of data.

3.2.6 **Pandas**

Pandas is another python library that is used for data manipulation and analysis. Pandas works with dataframes. Pandas has a lot of inbuilt tools that could be used for a host of things. One important function is data cleaning. According to IBM analytics about 80% of the time spent on a machine learning project is spent on data cleaning. A lot of datasets available have blank fields and this can greatly affect a model negatively. An example of a function in pandas for dealing with this is the '.isnull()' function.

3.2.7 **System Configuration**

This research was implemented on a system with the following configuration:

- 4 CPU core 2.5GHz
- 6 GB RAM
- Linux Mint 18 Operating System
- Python 2.7

CHAPTER IV – RESULTS

This chapter outlines the the results of the models and the metrics utilized to assess the implemented algorithms. In order to understand this section clearer, the following terms would be defined.

- **True Positive(TP).** A true positive outcome is one where the model predicts a positive outcome correctly.
- **False Positive(FP).** A false positive outcome is one where the model predicts a positive outcome incorrectly.
- **True Negative(TN).** A true negative outcome is one where the model predicts a negative outcome correctly.
- **False Negative(FN).** A false negative outcome is one where the model predicts a negative outcome incorrectly. (google developers, 2019)
- **Accuracy.** Accuracy is simply the measure of how correctly the model predicts a data given (Shung, 2018).

$$Accuracy = \frac{Total\ Number\ of\ correct\ predictions}{Total\ Number\ of\ predictions}$$

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

- **Precision.** Precision is the proportion of true positives out of the total number of positives. (Shung, 2018)

$$Precision = \frac{Total\ Number\ of\ correct\ predictions}{Total\ Number\ of\ positive\ predictions}$$

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

- **Recall.** Recall is the propotion of positives that was identified correctly (Shung, 2018)

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

- **F1-Score.** F1 Score is similar to accuracy but is a better metric because it seeks to create a balance between precision and recall especially when there is an un even class. F1 Score is given by (Shung, 2018):

  - $F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

4.1 K Nearest Neighbor and Random Forest

Table 4.1 KNN accuracy (k = 1 to 15)

Value of K	Accuracy(/1)
1	0.9952
2	0.9951
3	0.9948
4	0.9944
5	0.9939
6	0.9939
7	0.9932
8	0.9930
9	0.9924
10	0.9922
11	0.9917
12	0.9914
13	0.9905
14	0.9899
15	0.9893

As discussed in chapter 3.1.2 hyper parameter searching is a non-trivial task, as such there are different ways to go about it. For the sake of this research parameter search was done following two different methods. An informal rule states that the hyperparameter K should be chosen by finding the square root of the total number of rows in the training

set, if the number or estimate turns out to be an even number, add one to make it an odd number in order to prevent ties while voting. Following this grid search was used first time around for K = 1 to K = 15 which produced a mean accuracy value of 99.39%, the individual accuracy values for the individual K parameters are displayed in the table above.

Table 4.2 K Nearest Neighbor (80-20)

K	Accuracy (%)
1	99.48
2	99.54
3	99.49
4	99.44
5	99.37

Mean Accuracy = 95.375%

4.1.2 **The 60-40 Split Model**

Table 4.3 10 fold Cross validation K = 317

Index	Accuracy (%)
1	95.98
2	95.89
3	96.06
4	96.02
5	95.94
6	95.85
7	95.85
8	96.05
9	95.62
10	95.83

Mean Accuracy = 95.91%

The second method which stated the use of the square root to estimate an optimal K value was combined with grid search. Therefore since the square root was give as 317 approximately, grid search was performed from K = 300 to K= 322 using an 80-20 training-testing split, and repeated for K = 315 to K = 318 with a 60-40 training-testing

split. After both of these 10-fold cross validation was carried out on both the results and mean accuracy was recorded.

Table 4.4 Classification Report

Class	Precision	Recall	F1-Score
Positive	0.96	0.95	0.96
Negative	0.95	0.95	0.95

Table 4.5 Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	TN = 26524	FP = 411
Actual 1	FN = 300	TP = 23155

Table 4.6 Random Forest Classifier

N Estimators (number of trees)	Accuracy (%)
10	99.75
25	99.77
50	99.78
100	99.77

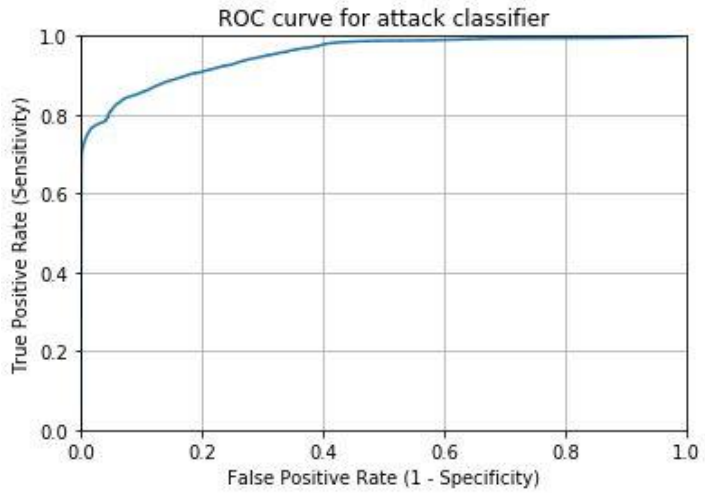


Figure 4.1 ROC Curve for the 60-40 Model

4.1.3 **The 80-20 Split Model**

Table 4.7 K Nearest Neighbors

K	Accuracy (%)
300	95.96
301	95.95
302	95.95
303	95.95
304	95.95
305	95.95
306	95.94
307	95.94
308	95.94
309	95.93
310	95.92
311	95.93
312	95.92
313	95.92
314	95.92
315	95.93
316	95.91
317	95.92
318	95.91
319	95.91
320	95.91
321	95.90
322	95.91

Table 4.8 10 fold Cross validation K = 5

Index	Accuracy (%)
1	99.4
2	99.35
3	99.4
4	99.38
5	99.38
6	99.35
7	99.33
8	99.32
9	99.46
10	99.98

Mean Accuracy = 99.39%, Standard Deviation = 000.45

Table 4.9 Classification Report

Class	Precision	Recall	F1-Score
Positive	1.00	0.99	0.99
Negative	0.99	0.99	0.99

Table 4.10 Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	TN = 26524	FP = 411
Actual 1	FN = 300	TP =23155

Table 4.11 Random Forest Classifier

N Estimators (number of trees)	Accuracy (%)
10	99.72
25	99.78
50	99.81
100	99.79

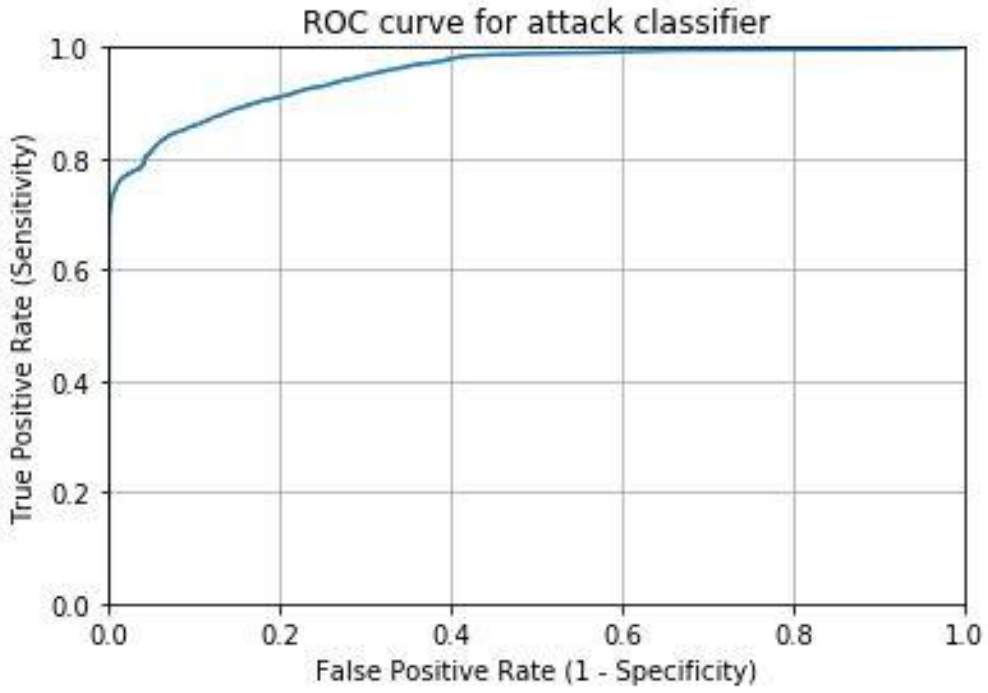


Figure 4.2 ROC Curve for the 80-20 Model

4.2 Model Comparison

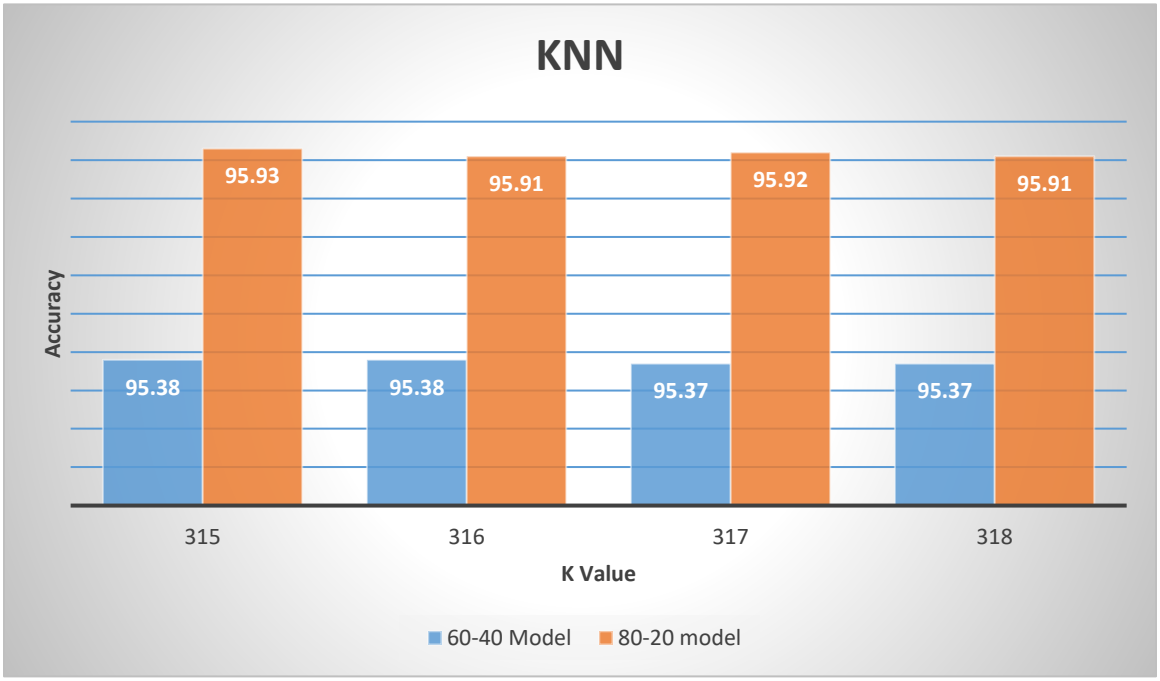


Figure 4.3 Model Accuracy Comparison on K Nearest Neighbor

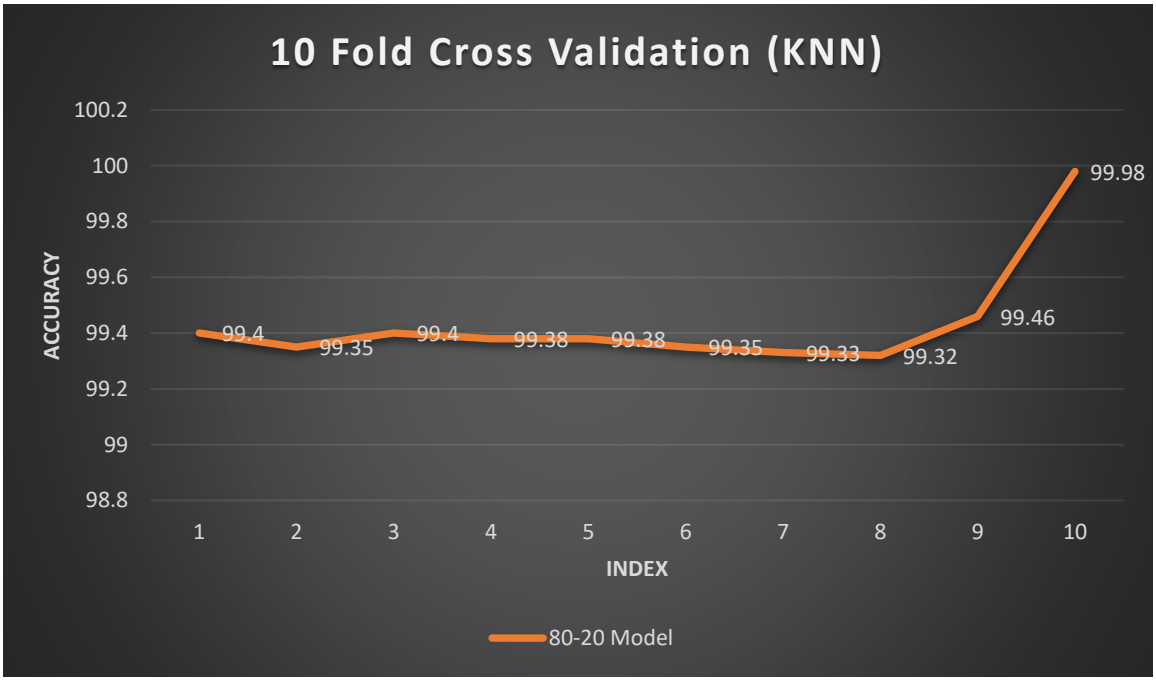


Figure 4.4 Model Accuracy Comparison on KNN with Cross Validation

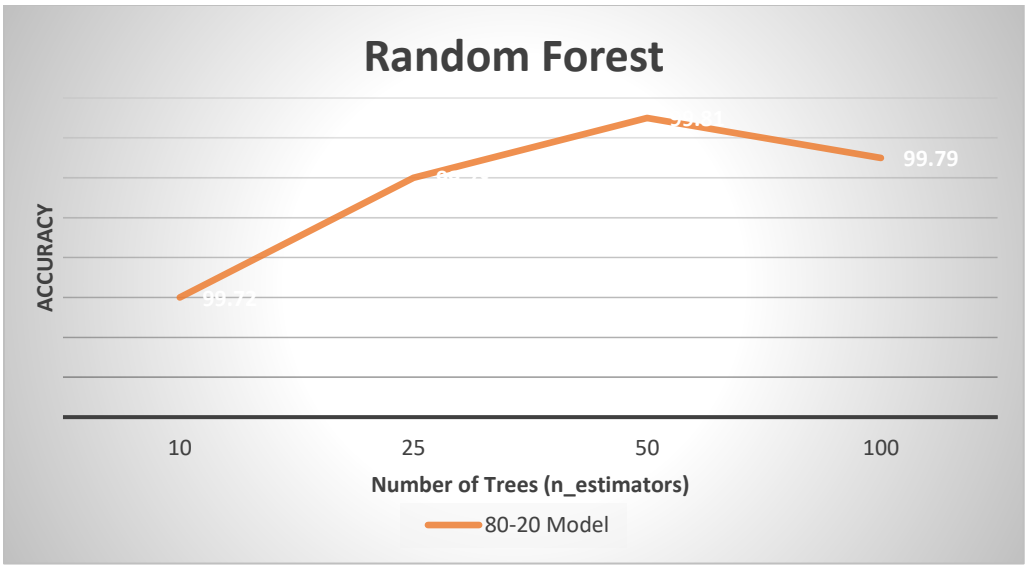


Figure 4.5 Model Accuracy Comparison on Random Forest

Table 4.12 KNN Comparison (K = 1, 5)

K	60-40 Model	80-20 model
1	99.52	99.48
2	99.51	99.54
3	99.48	99.49
4	99.44	99.44
5	99.39	99.37

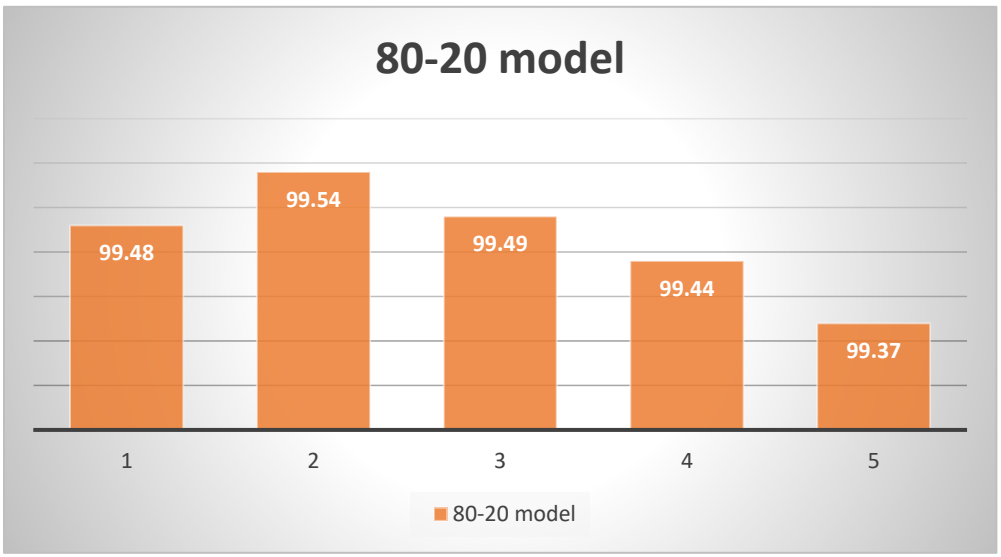


Figure 4.6 Model Accuracy Comparison on KNN (1, 5)

4.3 Discussion of Results

As discussed in section 4.2 above two models were created: a 60-40 train-test model and an 80-20 train-test. Each algorithm was run in both cases, KNN, KNN with K fold cross validation and random forest.

In the first KNN model the model trained using the 80-20 split had a higher accuracy than the 60-40 model, this is expected because more training data is always better for a model. In the second case the 10 fold cross validation was applied to both models and they had the same accuracy across board for each of the 10 runs.

Finally the models were both trained using four different instances of random forest algorithm. The n_estimators were set at 10, 25, 50 and 100. At n_est = 10 the 60-40 model performed better than the 80-20 model, but in the remaining three instances the 80-20 model outperformed the 60-40 model. For n_estimator values greater than 100 on the random forest model generated the same output which is the reason 100 was the largest value used in the model.

CHAPTER V –CONCLUSION

During the course of this research work, an intrusion detection machine learning model was developed using the K nearest neighbors and the random forest ensemble algorithm. The model was run multiple times with different variables and conditions to study how each parameter affects the model. In some cases, it was observed that even though there was no visible change in the detection accuracy, there was a significant delay in the time it took to make a classification especially in the k nearest neighbors classifier. Feature selection and cleaning make all the difference for models that utilize the k nearest neighbors' model.

In the random forest it was the effects of the variables like the n_estimators were also outlined. One recurring drawback was that the model started memorizing the dataset and giving the same exact classification accuracy as the number of estimators rose, for example during one of the runs of the model, the random forest n_estimators at 100, 500 and 100 produced the exact same classification to the tenth decimal. This is a signal that the model has memorized the data. This could be solved a couple of ways: the testing set could be split so that different testing data is used each time the model is used. In a real-life scenario this wouldn't be an issue as the data is changing with time.

The main aim of this research which was to evaluate the conditions under which the classification accuracy for the models on the NSL-KDD dataset was achieved.

CHAPTER VI – FUTURE WORK

Future work recommendations include the following:

Train the model to enable multiclass classification

This would enable the model to be more robust as the current model in this research makes use of binary class classification. Even though one of the objectives of this research was to create an intrusion detection model that was good at generalizing attacks to make it more impervious to zero-day attacks. Having a model that can identify the specific attack would also be useful in terms of letting the security system or system administrator know the specific action to take regarding the detected attack.

Use a newer dataset

Majority of the cybersecurity research published makes use of one form of the KDD99 dataset (Özgür A). This research utilizes the NSL-KDD+ dataset which is a fork of the KDD99 dataset. The University of New South Wales NB15 dataset is a newer dataset that could be considered.

APPENDIX A – Code Used (Python)

A.1 Imports

```
#Imports
# Import required libraries
# Python 2.7 environment
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Import ML algorithms
from sklearn.kernel_approximation import RBFSampler
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
```

A.2 Feature Extracton and Selection

```
#feature selection and extraction

# import datasets
# Dataset for all 39 features and Multiclass Target
data = pd.read_csv("/home/ilemona/Downloads/proposed_projects/Network-
Intrusion-Detection-master/UNSW-
NB15/classical/binary/KDDTrain_with_titles_num_types.csv")
# Dataset for Binary class Target
data_bin = pd.read_csv("/home/ilemona/Downloads/proposed_projects/Network-
Intrusion-Detection-master/UNSW-
NB15/classical/binary/dataset_with_conattack_numbers/KDDTrain_binary_targe
ts.csv")
#import seaborn
import seaborn as sns
# allow matplotlib display graphs in the notebook
%matplotlib inline
# input features
data_features = data[['duration', 'src_bytes', 'land','wrong_fragment',
'urgent', 'hot','num_failed_logins', 'logged_in', 'num_compromised',
 'root_shell', 'su_attempted', 'num_root','num_file_creations',
'num_shells',
 'num_access_files', 'num_outbound_cmds', 'is_host_login',
'is_guest_login',
 'count', 'srv_count', 'serror_rate','srv_serror_rate','rerror_rate',
'srv_rerror_rate',
```

```
 'same_srv_rate', 'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
'dst_host_srv_count',
 'srv_diff_host_rate','dst_host_count', 'dst_host_srv_count',
'dst_host_same_srv_rate','dst_host_same_src_port_rate',
 'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
'dst_host_srv_rerror_rate'  ]]
#Binary class
data_target_bin = data_bin.attack_name

#storing features in X and response vectors in y
X = data_features
#y = data_target
y_bin = data_target_bin
# print shapes
print X.shape
#print y.shape
print y_bin.shape
#start KNN classification
#import the class
from sklearn.neighbors import KNeighborsClassifier
#instantiate the estimator
knn = KNeighborsClassifier(n_neighbors = 1)
#fitting the training model
#Multiclass
#knn.fit(X, y)

#Binary
knn.fit(X, y_bin)
# import metrics and logistic regression
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
# test_size 0.4 means 40% split for testing, this was changed to 0.2 in
the 80-20 split
X_train, X_test, y_train, y_test = train_test_split(X, y_bin, test_size =
0.4, random_state = 4)
```

A.3 KNN and K-Fold Cross Validation

```
#classification K Nearest Nieghbors and Knn cross validation

# classifying with K-Nearest-Neighbors
# At K = 1
knn = KNeighborsClassifier(n_neighbors = 1)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print metrics.accuracy_score(y_test, y_pred)

# At K = 5
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print metrics.accuracy_score(y_test, y_pred)

#Trying values of K from 315 to 318 to get the best value of K for
Accuracy

#Hyperparameter selection for Knn in range of sqrt(k)
k_range = range(1, 15)
scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    scores.append(metrics.accuracy_score(y_test, y_pred))

#import Matplotlib (scientific graph plotting library)
import matplotlib.pyplot as plt

# allow plots to appear within the notebook
%matplotlib inline

# plot the relationship between K and testing accuracy
plt.plot(k_range, scores)
plt.xlabel('Value of k for KNN')
plt.ylabel('Testing Accuracy')
#sklearn.model_selection replaces sklearn.cross_validation
from sklearn.model_selection import cross_val_score
```

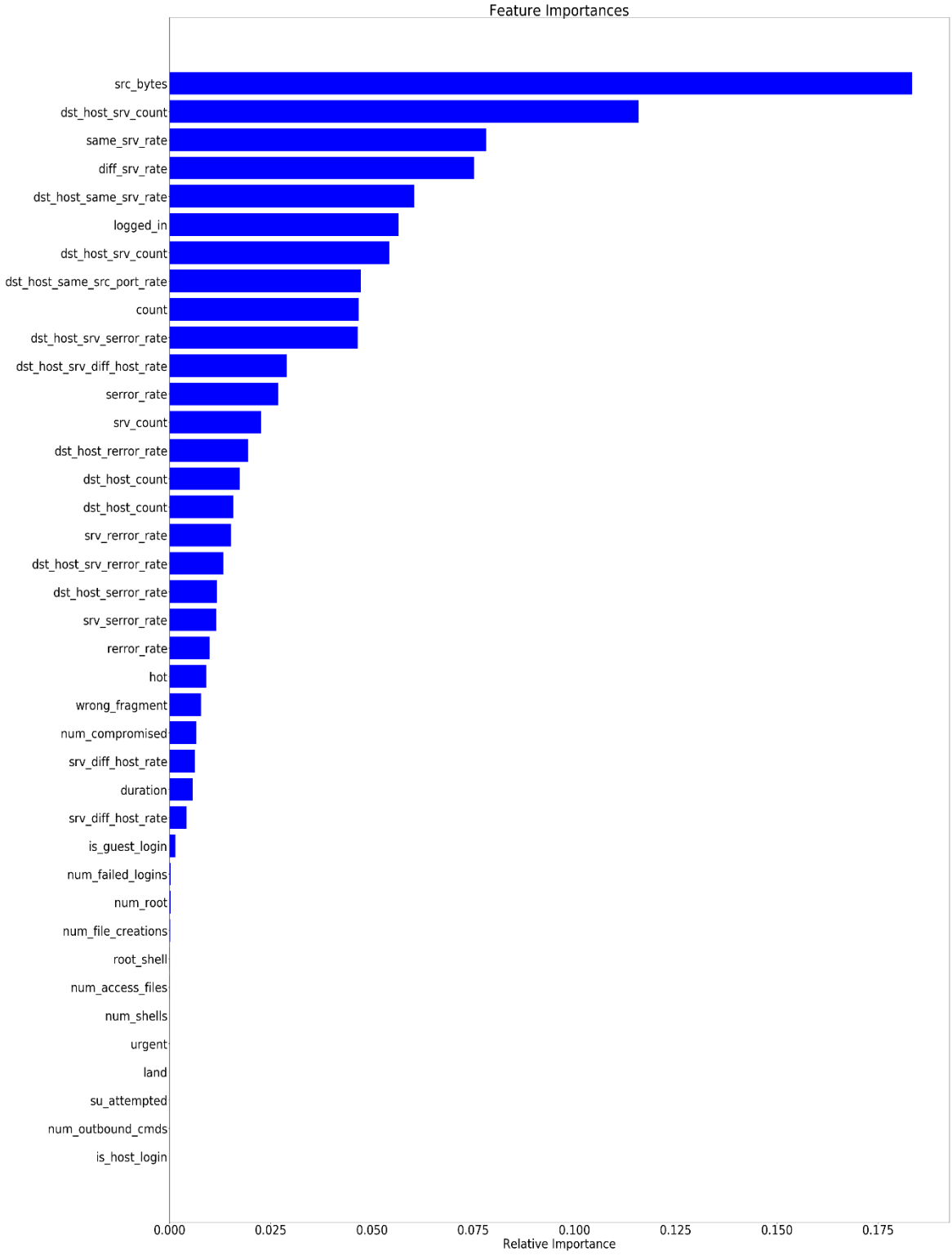
A.4 Random Forest

```
#Random Forest classificaton

#random forest classifier n_estimators = 10
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 10)
model.fit(X_train, y_train)
#Print the accuracy
model.score(X_test, y_test)

#random forest classifier n_estimators = 25
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 25)
model.fit(X_train, y_train)
model.score(X_test, y_test)
#random forest classifier n_estimators = 25
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 50)
model.fit(X_train, y_train)
model.score(X_test, y_test)
#random forest classifier n_estimators = 25
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 100)
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

APPENDIX B – Feature Importance and feature Description



REFERENCES

A. Buczak, E. G. (n.d.). survey of data mining and machine learning methods for cybersecurity intrusion detection. *IEEE Communications Surveys & Tutorials, ,* 1153–1176.

A. Elike Hodo, X. J. (2017). *Shallow and deep networks intrusion detection system: A taxonomy and survey,.* Retrieved from CoRR: https://arxiv.org/abs/1708.07174

A. P. Singh. (2016). nalysis of host-based and network-based intrusion detection system.

Abliz, M. (2011). *Internet denial of service attacks and defense mechanisms.* University of Pittsburgh,Department of Computer Science.

Aliyev, V. (2010). Using honeypots to study skill level of attackers based on. Chalmers University of Technology.

Chumachenko, K. (2017). Machine learning methods for malware detection and classification.

google developers. (2019). *Machine learning crash course*. Retrieved from google developers: https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative

Ho, T. K. (1995). trees, Random Decision. *Bell Labs*.

J. G. Noraini, M. R. (2011). Genetic algorithm performance with different selection strategiesin solving tsp. *World Congress on Engineering.* World Congress on Engineering.

José, I. (2018). *KNN (K-Nearest Neighbors) #1*. Retrieved from Towards Data Science : https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d

M. Gupta. (2015). Hybrid intrusion detection system: Technology and development,. *InternationalJournal of Computer Applications*.

Oscar Jimenez-del-Toro, S. O. (2017). Analysis of Histopathology Images: From Traditional Machine Learning to Deep Learning. In O. S.-K. Adrien Depeursinge, *Biomedical Texture Analysis* (pp. 281-314). Academic Press.

Özgür A, E. H. (n.d.). . A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. *PeerJ Preprints*. PeerJ Preprints.

S. Paliwal, R. G. (2012). Denial-of-service, probing & remote to user (r2l) attackdetection using genetic algorithm. *International Journal of Computer Applications,*, 57-62.

Saxena, R. (2016). *Knn sklearn, k-nearest neighbor implementation with scikit learn*. Retrieved from Dataaspirant: http://dataaspirant.com/2016/12/30/k-nearest-neighbor-implementation-scikit-learn/

Shung, K. P. (2018). *Accuracy, Precision, Recall or F1?* Retrieved from Towards Data Science: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

T. M. I. White Paper. (2012). *Addressing big data security challenges: The righttools for smart protection*. Retrieved from Trend Micro: https://www.trendmicro.com/en_us/business.html

V. Shmatikov, M.-H. W. (2007). ecurity against probe-response attacks in collaborativeintrusion detection. *2007 Workshop on Large Scale Attack Defense,ser. LSAD '07* (pp. 129-136). NY: ACM.