T.C.
ISTANBUL AYDIN UNIVERSITY
INSTITUTE OF GRADUATE STUDIES

MACHINE LEARNING TECHNIQUES FOR CYBERSECURITY

MASTER'S THESIS

Fadi HAMMAD

Department of Engineering
Electrical and Electronics Engineering Program

JULY, 2022

T.C.
ISTANBUL AYDIN UNIVERSITY
INSTITUTE OF GRADUATE STUDIES

MACHINE LEARNING TECHNIQUES FOR CYBERSECURITY

MASTER'S THESIS

Fadi HAMMAD

(Y2013.300014)

Department of Engineering
Electrical and Electronics Engineering Program

Thesis Advisor: Assist. Prof. Dr. Necip Gökhan KASAPOĞLU

JULY, 2022

ONAY FORMU

DECLARATION

I hereby declare with respect that the study “Machine Learning Techniques for Cyber Security”, which I submitted as a Master / PhD thesis, is written without any assistance in violation of scientific ethics and traditions in all the processes from the Project phase to the conclusion of the thesis and that the works I have benefited are from those shown in the References. (15/07/2022)

Fadi HAMMAD

FOREWORD

This thesis was written for my Master degree in Electrical and Electronics Engineering with specialization in Cyber Security at Istanbul Aydin University, Turkey. The subject of this thesis is related to Machine Learning Techniques for Cyber Security.

After thanking Allah Almighty and my family for their endless support, I would like to thank my supervisor Dr. Necip Gökhan KASAPOĞLU for his guidance, reviews and recommendations.

July 2022 Fadi HAMMAD

SİBER GÜVENLİK İÇİN MAKİNE ÖĞRENİMİ TEKNİKLERİ

ÖZET

Ağ saldırıları ve siber tehditler her geçen gün daha karmaşık hale gelmektedir. Dünya çapında giderek artan sayıda günlük saldırılar gerçekleşiyor ve iş istasyonlara ve bireysel cihazlara sızmak için birçok farklı yöntem ve strateji vardır ve geliştirilmektedir. Siber güvenlik tehditleri, ağ ortamlarında büyüyen önemli bir endişe kaynağıdır. Saldırı Tespit Sistemlerinin (IDS) geliştirilmesi, ekstra güvenlik seviyesi sağlamak için esastır. Saldırı Tespit Sistemi, ağ bağlantılı sistemlerin güvenliğini sağlamak için önemli bir sorundur.

Bu tez, siber güvenlik zorluklarını çözmek için makine öğrenimi tekniklerinin nasıl kullanıldığına dair genel bir bakış sunmaktadır. Ek olarak, tez, mevcut Yapay Zeka keşiflerini ve bunların siber güvenliği nasıl geliştirdiğine vurgu yapmaktadır. Yapay Zeka kullanan Saldırı Tespit sistemleri, algılama ve tepki vermede daha önleyici ve duyarlı hale gelirken, otomatik olmayan geleneksel faliyetlerine kıyasla izleme güvenlik faaliyetlerinde zamandan tasarruf sağlar. Bu araştırma, Makine Öğrenimi algoritmalarını kullanarak saldırı tespit sistemlerinin analizini gerçekleştirmektedir.

Anahtar Kelimeler- Makine Öğrenimi, Saldırı Tespit Sistemi, Siber Güvenlik.

MACHINE LEARNING TECHNIQUES FOR CYBER SECURITY

ABSTRACT

Network intrusions and cyber threats are getting more complicated day after day. Worldwide, there are an increasing number of daily attacks, and there are many different methods and strategies for breaking into business systems and individual devices. Cyber-security threats are a growing concern in networked environments. The development of Intrusion Detection Systems (IDSs) is fundamental in order to provide extra level of security. Intrusion Detection System is a key problem for ensuring the security of networked systems.

This thesis shows an overview of how machine learning techniques are used to solve cybersecurity challenges. Additionally, this thesis highlights current Artificial Intelligence discoveries and how they enhance cyber security. Intrusion Detection systems that use Artificial Intelligence become more preventive and responsive in detection and reaction, as well as saving time on human monitoring security activities. This research intends to analyze and improve intrusion detection systems via the use of Machine Learning algorithms.

Keywords- Machine Learning, Intrusion Detection System, Cyber Security**.**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

A. Introduction:

In last few years, The Internet and computer technologies have witnessed tremendous growth and have become an essential element of today's generation of people. Security is becoming increasingly important as the use of computer applications and computer networks grows at an enormous speed. Attackers may exploit a variety of application mechanisms to cause destruction in yo ur business or organization. According to the National Institute of Standards and Technology (NIST), American businesses experienced losses of up to $65.6 billion in 2017 because of cyber-attacks. Because of the significant growth in the number of cyber-attacks, artificial intelligence and machine learning-based approaches have become critical in identifying cyber threats. From this perspective, they are crucial for various kinds of cyber security researches, like intrusion detection systems. Intrusion Detect Systems (IDS) are a group of cybersecurity-based technologies that were first created to detect intrusions and exploits in a target host. Therefore, it becomes clear that machine learning techniques and IDS may function at a superhuman level when united. In this thesis, deeper research concerning this combination will be discussed.

B. Cybersecurity:

This chapter covers fundamental security principles, such as cybersecurity needs and methods for achieving them. It also discusses the many types of intrusion detection techniques and their present condition in terms of fulfilling computer security goals.

1. Cybersecurity Principles:

A computer environment is made up of a variety of hardware and software components. Cybersecurity covers securing such assets. To achieve this security, the three computer security principles should be maintained. These principles are Confidentiality, Integrity, and Availability (CIA Triad).The next sections will give brief definitions of each one.

Fig 1. CIA Triad: The three principles of Cybersecurity.

a. Confidentiality:

The confidentiality definition states that computer data can only be accessed by authorized users. This access can include everything from reading to writing to printing to simply knowing that a given item exists. Some examples that attain this are military secrets.

b. Integrity:

The concept of integrity states that only authorized users and may modify computer data. Updating, removing, writing, and generating assets are all examples of asset modifications. Examples in accomplishing integrity includes incorrect data entered by the user into a database.

c. Availability:

The availability principle states that systems, functions, and data must be accessible on demand based on agreed-upon guidelines and service levels. An example that guarantees availability Network Load Balancing.

These principles are the main goals in Cybersecurity that were set by *Donn Parke*r in 1998. There are more features that can be recognized part of computer security in addition to the ones listed above. Authentication, accountability, and reliability, are some of them.

2. Challenges in Cybersecurity:

If all of the previously stated conditions are met, a computer system may be made completely secure. In practice, however, it is difficult to build a system with perfect security and usability. A breach in anyone of these security principles (CIA Triad) will compromise the system, leaving it insecure and vulnerable to attacks from hackers. To handle this scenario, it is understood that a system may fail, hence detection and response methods, in addition to protective measures, must be implemented. Protecting the asset is the proactive aspect of security. To avoid any breaches in CIA Triad, the asset is kept secure. Detection methods are used to identify potential security breaches, and their effectiveness is determined by the time it takes to discover them. The responding procedure to security breaches complements the detection procedure. In different cases, the response type varies depending on the condition. Some response types include assessing the impact, recovering from the impact, learning from encounter, and so on.

C. Intrusion Detection System:

One of the main defense lines that keeps a system or data well secured is the Intrusion Detection System (IDS). Intrusion detection is a technique for detecting a breach in an organization's security policy. Intrusion detection is a technique for detecting a breach in an organization's security policy. External parties (intruders - hackers) or inside personnel may be responsible for these breaches (i.e. insiders). While progress has been made in detecting attacker breaches, internal incidents are

difficult to identify. The IDS is used to detect suspicious activity over the network and send an alarm to the user.

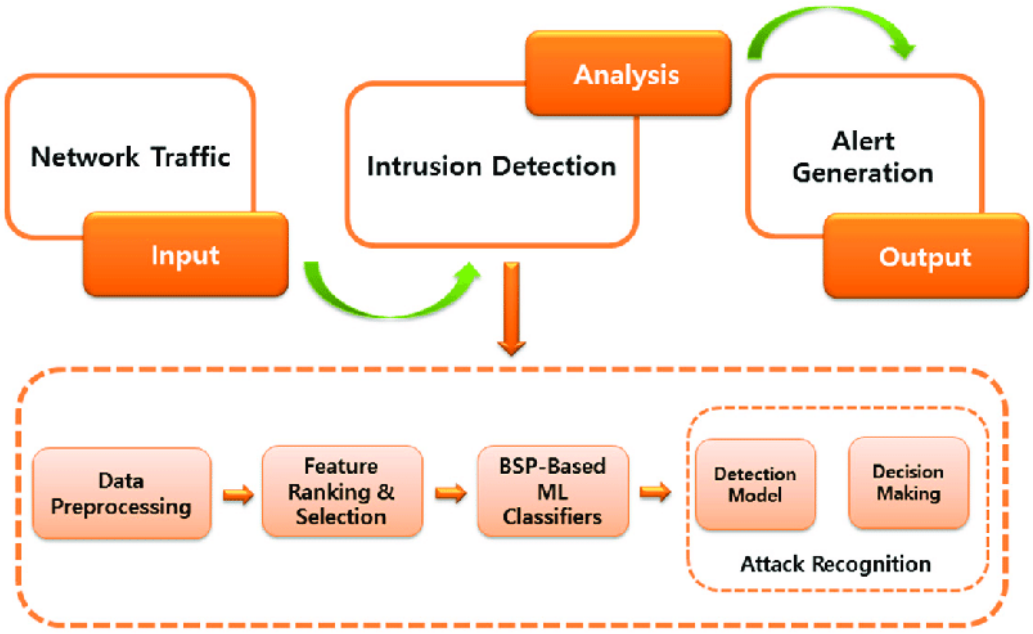

Fig 2. Intrusion Detection System Architecture.

1. Importance of Intrusion Detection System:

The following are some of the reasons why intrusion detection has earned a serious attention as noted by:

1. If an intrusion is noticed immediately, an attack or malicious behavior can be quickly identified and eliminated from the system. Even if the detection is not made early enough to stop the attacker, the sooner the intrusion is discovered, the less harm is done and the faster recovery may be performed.

2. A good intrusion detection system can operate as a deterrent, preventing intrusions into any asset.

3. Intrusion detection allows for the collection of data on intrusion strategies that may be utilized to improve the intrusion prevention system.

2. Intrusion Detection Types:

IDS are classed as follows based on their functionality:

1. Signature-based IDS (also known as Misuse detection): The Signature-based intrusion detection technology is created to monitor network traffic and detect attack patterns. If a match is discovered, an alarm is generated. Only known attacks may be detected by this type. The identified patterns are known as sequences, and each sequence is a specified number of bytes or a collection of 0's and 1's in the network. This technique works in pattern matching from database of known attacks using machine learning algorithms. It is simple to identify cyberattacks whose patterns are stored in the system as signatures. However, detecting new cyberattacks with no signature stored or modified patterns is problematic. Its only drawback is that it must be regularly updated.

2. Anomaly-based IDS: As mentioned before, the Signature-Based Detection approach makes it harder to identify unknown or new malware threats. As a result, organizations apply the anomaly-based intrusion detection approach to discover new and unknown suspicious activities and intrusions that the signature-based detection approach cannot reliably detect. An anomaly can be detected as an increase in packet (data) activity or multiple incorrect login .This type uses observable system behavior to create models for regular system functioning. An activity model is used to analyze and store normal behaviors of users in the system as a baseline. The disadvantage of this method is the defining of normal behavior. If this approach identifies any receiving patterns that are not in the model, it classifies them as malicious behavior.

3. Hybrid IDS: Both types of Signature-based intrusion detection and Anomaly-based intrusion detection are combined to build the best IDS.

The main focus in this thesis is on the Signature-based IDS and how machine learning techniques are used to build such a system and will it be able to detect new undefined attacks. Then next part will discuss about machine learning and its contribution to cybersecurity field in an IDS.

D. Machine Learning:

Machine Learning (ML) is a subfield of artificial intelligence that focuses on the development of technology that allow a computer to learn on its own from data and previous experiences. These data are referred to as **training data.** These training data are applied to machine learning algorithms to build a mathematical model that will give decisions and classifications to some new data. These new data are referred to as **testing data.** The accuracy and performance these models are strongly dependent to the training data that is the more data we feed the model the better it will correctly classify the testing data.

1. Machine Learning Workflow:

The workflow of building a machine learning model as applied as show in Fig.2 below:

1. Preparing training (input) data.
2. Choosing machine learning algorithm.
3. Building a model by applying the training data to the chosen algorithm.
4. Feeding the model with testing data for classification and decision making (output).
5. Performance and accuracy Evaluation.

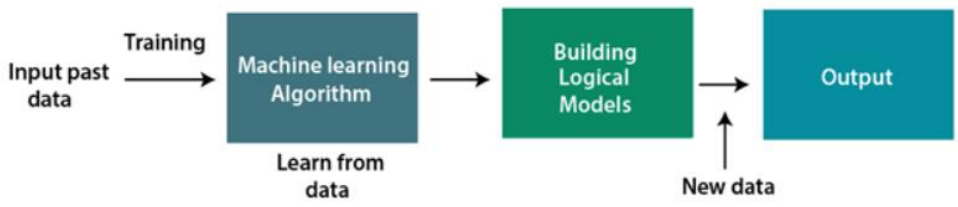

Fig 3. ML Workflow Diagram.

2. Machine Learning Types:

Machine learning algorithms are categorized into three types:

1. Supervised Learning**:** A model using this algorithm is built by using labeled (classified) training data and gives decisions to testing data based on these trained classifications. The system constructs a model using labeled data to analyze the datasets and learn about each data, after completion, the model is tested feeding sample data to see if it accurately predicts the output. The idea behind supervised learning is matching the train data to the test data. Regression and classification are two algorithms that support the supervised machine learning type:

a. Classification: this algorithm deals with discrete data like male and female, spam and not spam, or attack and normal. This algorithm holds many types in machine learning like K-Nearest Neighbors, Naïve Bayes, Decision Tree and support vector machines which will be discussed later on in this thesis.

b. Regression: this algorithm deals with continuous data like price, salary or age. It works by building a function that links between dependent and independent data. Simple linear, multiple linear, and polynomial regression are some types that use the regression algorithm.

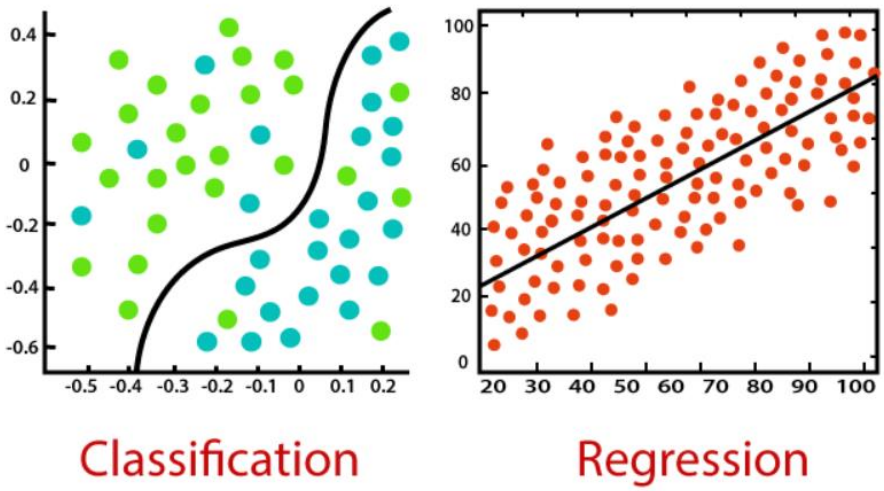

Fig 4. Classification vs. Regression.

2. Unsupervised Learning: A model using this algorithm is built by using unlabeled (unclassified) training data. Unsupervised learning aims to reorganize incoming data into new features or a collection of objects with similar patterns.

3. Reinforcement Learning: Reinforcement learning is a feedback-based learning strategy in which a learning agent is rewarded for correct categorizing and fined for incorrect ones. With these feedbacks, the agent learns automatically and improves its performance. The agent interacts with and inspects the environment in reinforcement learning.

3. Machine Learning applied to Intrusion Detection Systems:

The Intrusion Detection System (IDS) is a piece of software that uses machine learning methods to identify network intrusion. IDS monitors a network or system for malicious activity and prevents unauthorized access from attackers, including insiders. The learning aim for the intrusion detector is to create a prediction model (i.e. a classifier) that can discriminate between 'bad connections' (intrusion/attacks) and 'good (normal) connections'. As a result, machine learning can assist cybersecurity field in taking a more effective and efficient approach to risk reduction and real-time cyberattack response.

E. Aims and Goals of Thesis:

This thesis aims to demonstrate how machine learning methods and algorithms contribute to the cybersecurity field when applied in intrusion detection systems by building a more robust and automated systems. Not all machine learning methods are applicable to be used for IDS since each method works with different kind of data, so only best methods will be illustrated. The goals of this thesis are:

1. Review Recent systems and researches about using machine learning in IDS

2. Build a machine learning models for an intrusion detection system using MATLAB

3. Finding the best machine learning model by comparing the results, performances and time consumed.

F. Literature Review:

1- A research was carried by *Cuelogic Technologies* on how machine learning algorithms will perform on *KDD Cup 1999 dataset* which the considered the most used and relied on in building IDS since it contains a variety of malicious and normal data that were grouped into this dataset. Their goal was to find how machine learning algorithms can be useful in IDS and the most accurate ML method that can minimize the number of false positive and false negative issued by an IDS for best classification. False positives are attacks or malicious data that were classified as normal and harmless data. False Negative is the opposite where normal data were misclassified as a threat or malicious data. The research was using Python and its corresponding *pandas, numpy, scipy* libraries as a testing lab environment. Various machine learning algorithms were applied to the dataset and then accuracy, precision, false positive and false negative rates were calculated for comparison purposes.

Machine Learning Classifiers	Correctly classified Instances	Incorrectly classified Instances	Accuracy Rate
j48	55865	4135	93.10%
Random Forest	56265	3735	93.77%
Random tree	55345	5655	90.57%
Decision table	55464	4536	92.44%
MLP	55141	4859	91.90%
Naive Bayes	54741	5259	91.23%
Bayes Network	54439	5561	90.73%

Table 1. Accuracy Rates of applied machine learning classifiers.

Machine Learning Classifiers	TP Rate	Precision
j48	0.931	0.989
Random Forest	0.938	0.991
Random tree	0.906	0.992
Decision table	0.924	0.944
MLP	0.919	0.978
Naive Bayes	0.912	0.988
Bayes Network	0.907	0.992

Table 2. True Positive and Precision Rates.

All of the algorithms were used on the same dataset using python libraries. As the results presents, the applied machine algorithms produced high accuracy and precision rates. It was concluded that in order to build the best performing IDS, companies should consider one or more of these machine learning algorithms. Failure in any IDS would be to the lack of using ML as a base for the intrusion detection system.

2- In deeper research was carried out to check which machine learning algorithm can detect type of attack by classification in the '*KDD Cup 1999 Dataset*'. This dataset holds normal connections and bad connections. These bad connection can be classified into four different groups: Probing attacks (information gathering attacks), Denial-of-Service (DoS) attacks (deny legitimate requests to a system), user-to-root (U2R) attacks (unauthorized access to local super-user or root), and remote-to-local (R2L) attacks (unauthorized local access from a remote machine). Certain attack types have been demonstrated to be more detectable by specific algorithms. To identify cyberattacks in the KDD 1999 Cup Intrusion Detection dataset, a multi-classifier machine learning model was developed combining these different methods of classification. For this examination, nine distinct algorithms from multiple areas were chosen. The performance measures were the probability of detection (PD) and the false

alarm rate (FAR). Theoretical results showed a significant boost in performance for detecting probing, DoS, and user-to-root attacks. The machine algorithms used were:

1-Multilayer perceptron (MLP).
2-Gaussian classifier (GAU).
3-K-means clustering (K-M).
4-Nearest cluster algorithm (NEA).
5-Incremental radial basis function (IRBF).
6-Leader algorithm (LEA).
7-Hypersphere algorithm (HYP).
8-Fuzzy ARTMAP (ART).
9-C4.5 decision tree (C4.5).

		Probe	DoS	U2R	R2L
MLP	*PD*	0.887	0.972	0.132	0.056
	FAR	0.004	0.003	5E-4	1E-4
GAU	*PD*	0.902	0.824	0.228	0.096
	FAR	0.113	0.009	0.005	0.001
K-M	*PD*	0.876	0.973	0.298	0.064
	FAR	0.026	0.004	0.004	0.001
NEA	*PD*	0.888	0.971	0.022	0.034
	FAR	0.005	0.003	6E-6	1E-4
RBF	*PD*	0.932	0.730	0.061	0.059
	FAR	0.188	0.002	4E-4	0.003
LEA	*PD*	0.838	0.972	0.066	0.001
	FAR	0.003	0.003	3E-4	3E-5
HYP	*PD*	0.848	0.972	0.083	0.010
	FAR	0.004	0.003	9E-5	5E-5
ART	*PD*	0.772	0.970	0.061	0.037
	FAR	0.002	0.003	1E-5	4E-5
C4.5	*PD*	0.808	0.970	0.018	0.046
	FAR	0.007	0.003	2E-5	5E-5

Table 3. Machine learning algorithms results for DP and FAR.

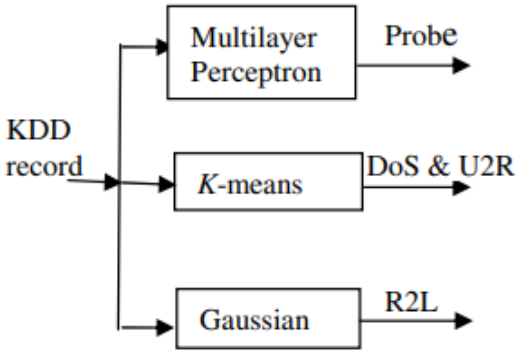

Fig 5. Multi-Classifier Model.

		Probe	DoS	U2R	R2L
KDD Cup Winner	*PD*	0.833	0.971	0.132	0.084
	FAR	0.006	0.003	3E-5	5E-5
KDD Cup RunnerUp	*PD*	0.833	0.971	0.132	0.084
	FAR	0.006	0.003	3E-5	5E-5
Agarwal and Joshi	*PD*	0.73	0.969	0.066	0.107
	FAR	8E-5	0.001	4E-5	8E-4
Multi-Classifier	*PD*	0.887	0.973	0.298	0.096
	FAR	0.004	0.004	0.004	0.001

Table 4. Comparing Proposed with Old Models.

3- In an experiment was set to compare two different machine learning algorithms on KDD Cup dataset. The Information Gain (G) method is used to measure which features are more effective in accurately categorizing the training data in decreasing order and the first 6 features showed up were used in this experiment. The experiment was set using 'Weka', a software that can apply machine learning algorithms on different datasets. The algorithms used were:

- NBTree that is a hybrid machine learning algorithm containing Decision-Tree classifiers and Naive Bayes classifiers.
- VFI algorithm, which is method that classifies data, based on voting frequency intervals.

Metric	*Value*
Time taken to build the model	1115.05s
Accuracy	99.94 %
Average Precision	90.33 %
Average Recall	92.72 %
Average F-Measure	91.14 %
Kappa Statistic	99.99 %

(a)

Metric	*Value*
Time taken to build the model	38.97s
Accuracy	99.89 %
Average Precision	94.54 %
Average Recall	90.84 %
Average F-Measure	92.28 %
Kappa Statistic	99.82 %

(b)

Metric	*Value*
Time taken to build the model	0.92s
Accuracy	86.58 %
Average Precision	41.27 %
Average Recall	80.54 %
Average F-Measure	44.05 %
Kappa Statistic	79.50 %

(c)

Metric	*Value*
Time taken to build the model	0.2s
Accuracy	75.81 %
Average Precision	35.71 %
Average Recall	75.82 %
Average F-Measure	37.43 %
Kappa Statistic	66.21 %

(d)

Table 5. Comparing Results Between NBTree and VFI methods.

(a) NBTree Results with all features included.

(b) NBTree Results with features selected using G.

(c) VFI Results with all features included.

(d) VFI Results with features selected using G.

The information gain (G) utilized in decreasing the time needed to build the models. G didn't affect the metrics results in NBTree method whereas it affected negatively on results in VFI method. Despite VFI method being faster than NBTree, the overall best performing experiment was (b). A good IDS needs to have highest values of accuracy, precision and recall.

4- An assessment was conducted in to verify if using support vector machine (SVM) learning algorithms can increase the accuracy in an IDS or not. The assessment was divided into two parts: In order to detect some network assaults and intrusions, it is first necessary to assess which SVM technique yields the best detection results. One-class and two-class SVMs are evaluated using linear and non-linear versions with Radial Basis Function (RBF) functions. Second, the best results from steps one were

compared to an unsupervised anomaly-based IDS. Six different datasets were used in this research which are:

- Normal: dataset that contains only normal data.
- *Attack01*: include malicious content that aims to replace the requesting website's HTTP header fields.
- *Attack02*: contain a malicious data that focus on replacing the images in the website.
- *Attack03*: contain a malicious data that comprises both *Attack01* and *Attack02*.
- *DeAuth*: contain a malicious data that deals with sending spoofed deauthentication frames to force the target to refresh the connection with the access point.
- *Probing*: include harmful content related to port scanning, a method for identifying network vulnerabilities via inspecting open ports.

The five datasets which contain malicious data contained also normal data. The table below illustrates the description of each dataset:

Dataset	Total Instances	Normal Instances	Normal Instances (%)	Malicious Instances	Malicious Instances (%)
Normal	3631	3631	100	n/a	n/a
Attack01	1361	1350	99.2	11	0.8
Attack02	14493	13498	93.1	995	6.9
Attack03	12130	12016	99.1	114	0.9
DeAuth	228	164	71.93	64	28.07
Probing	700484	696638	99.4	4220	0.6

Table 6. General Description of the Evaluated Datasets.

In the first part of the assessment, four different types of SVM algorithms were used on these datasets and results of Detection Rate (DR), False Positive Rate (FPr), False Negative Rate (FNr), and Overall, Success Rate (OSR) were collected in the tables below for comparison:

Linear Two-Class SVM				
Dataset	**DR (%)**	**FPr (%)**	**FNr (%)**	**OSR (%)**
Attack01	100	0	0	100
Attack02	100	0	0	100
Attack03	100	0	0	100
DeAuth	100	0	0	100
Probing	98.78	16.6	1.22	83.4

Non-Linear Two-Class SVM with Gaussian Radial Basis				
Dataset	**DR (%)**	**FPr (%)**	**FNr (%)**	**OSR (%)**
Attack01	62.5	0	37.5	99.66
Attack02	99.52	0	0.48	99.97
Attack03	93.51	0	6.49	99.94
DeAuth	97.78	0	2.22	99.25
Probing	97.85	18.32	2.15	81.67

Linear One-Class SVM				
Dataset	**DR (%)**	**FPr (%)**	**FNr (%)**	**OSR (%)**
Attack01	100	0	0	99.93
Attack02	89.25	0.41	10.75	98.85
Attack03	99.12	2.46	0.88	97.53
DeAuth	100	1.75	0	98.25
Probing	93.78	11.05	6.22	88.91

Non-Linear One-Class SVM with Gaussian Radial Basis				
Dataset	**DR (%)**	**FPr (%)**	**FNr (%)**	**OSR (%)**
Attack01	100	14.62	0	85.38
Attack02	100	6.84	0	93.16
Attack03	100	5.82	0	94.18
DeAuth	95.38	3.07	4.62	95.61
Probing	61.37	1.15	38.63	98.61

Table 7. Detection Results Using Different SVM Algorithms.

Obviously, the outcomes produced by the linear two class SVM approach are the most accurate. In the second part of the assessment, an anomaly-based IDS built by the same team was used for detection for the same datasets, the results are shown in the table below:

Unsupervised Anomaly-based IDS				
Dataset	**DR (%)**	**FPr (%)**	**FNr (%)**	**OSR (%)**
Attack01	100	0	0	100
Attack02	100	0.03	0	99.97
Attack03	100	0.06	0	99.94
DeAuth	100	2.19	0	97.81
Probing	18.82	15.99	81.18	83.52

Table 8. Detection Results of Anomaly-Based IDS.

After comparing the results between Linear Two-Class SVM and Anomaly-Based IDS, It is noticeable in Probing Attack DR results are most accurate in Linear Two-Class SVM. With that, it is found that the linear SVM was the winner having the best outcomes. As a result of these findings, it appears that anomaly-based intrusion detection systems might benefit from the application of Linear SVM to improve detection accuracy.

5- Another research, a lot of machine learning algorithms were implemented to figure out which is the best machine learning model that can be used on KDDCup-'99' dataset and give best results to be used in building an IDS. The experiment was focused on applying deep neural networks (DNN) by changing number of hidden layers used from one to five and comparing the best with other machine learning

algorithms. For that ‘Keras’ and ‘Tensorflow’ were used. All other conventional machine learning techniques were outperformed by the DNN 3-layer network. This is due to DNNs' capacity to extract important data and features, as well as their non-linearity, which holds power over other techniques. And after comparing it with other different methods, this technique was the winner. It was deduced that DNN can play an important role in facilitating the performance of an IDS. The results of accuracy, precision, recall, and f1-score were collected from each method for comparison as illustrated in the table below:

Algorithm	Accuracy	Precision	Recall	f1-score
DNN-1	0.929	0.998	0.915	0.954
DNN-2	0.929	0.998	0.914	0.954
DNN-3	0.930	0.997	0.915	0.955
DNN-4	0.929	0.999	0.913	0.954
DNN-5	0.927	0.998	0.911	0.953
Ada Boost	0.925	0.995	0.911	0.951
Decision Tree	0.928	0.999	0.912	0.953
K-Nearest Neighbour	0.929	0.998	0.913	0.954
Linear Regression	0.848	0.989	0.821	0.897
Navie Bayes	0.929	0.988	0.923	0.955
Random Forest	0.927	0.999	0.910	0.953
SVM*-Linear	0.811	0.994	0.770	0.868
SVM*-rbf	0.811	0.992	0.772	0.868

Table 9. Results of DNN and other ML algorithms.

II. BASELINE ALGORITHMS

A. Baseline Algorithms:

Three baseline algorithms that I used for writing my thesis are Linear Support Vector Machine, Radial Basis Function, Logistic Regression, and Deep Learning. As mentioned in literature review, a lot of researches and experiments were done on IDS to build a robust system with high performance using different datasets and techniques in machine learning and artificial intelligence. Recent experiments proved that one the best models that can be reliable is SVM model. Using SVM algorithms on datasets that are examined by IDS in real life scenarios resulted in high accuracies and low false negatives. Linear Support Vector Machine (Linear SVM) and Radial Basis Function (Non-linear SVM) are examples of these algorithms that I used in my thesis as baseline algorithms and in this part, I will explain the mechanism of these algorithms and their corresponding experiments in IDS.

B. Support Vector Machine:

A support vector machine (SVM) is a machine learning algorithm that analyzes the structure of input data, and outputs an optimal classifier for it. It is a type of non-linear classifier that takes a set of initially overlapping decision regions, or decision boundaries, divides them into two sets of non-overlapping subsets, then projects each training example into one of the new regions to decide which class with the highest probability it belongs to. Support vector machines were first introduced in 1999 by Vladimir Vapnik and Alexey Chervonenkis (Chervonenkis' main contribution was the introduction of kernel methods, one of the formulations of SVM). A support vector machine (SVM) is a supervised learning model that analyzes data with the goal of making predictions. A supervised learning model that takes labeled data for training and build a model that will give decision to tested data based on input data. SVM is a classification problem where the output data will be discrete labels.

Support vector machines work in two stages. The first stage determines the decision boundary, which separates classes, based on training instances. The second stage fits within class supports (i.e., cases belonging to one class). Support for one class is assigned a positive weight, while another class gets negative weight, producing an asymmetrical classification scheme also known as margins; this makes SVMers different from other classification schemes based on probabilities-only scoring systems such as logistic regression or naive Bayes where all classes are weighted equally. Margins are the distances that separates classes from each other. Hyperplane is the line that splits data into two classes, and also called the decision boundary. Support vectors are nearest points to the hyperplane. The optimal hyperplane is the best boundary splits the data because it is as far as possible from these support vectors which is another way of saying maximized margin. The goal of SVM is to find this hyperplane from train data and give decisions to test data based on this decision boundary. Fig 5. below illustrates the explanation:

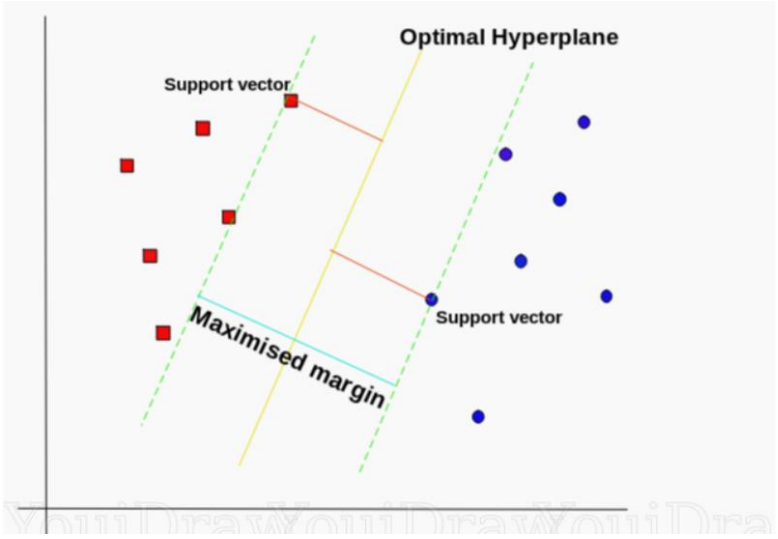

Fig 6. Optimal Hyperplane using the SVM algorithm.

1. Applications of SVM:

The main goal of SVM is to correctly classify new input data. With its good performance and high accuracy, SVM applications are widely used, seen and implemented in new technologies and appliances. Here I will give brief examples of these applications:

- **Face Detection:** SVM classifies parts of the image as face or non-face and draws a square boundary around the face.

- **Text and hypertext categorization:** SVM classifies documents into distinct groups based on training data like articles, e-mails, and web pages.

- **Classification of Images:** SVMs have a higher search accuracy when it comes to classifying images. It outperforms typical query-based refining approaches in terms of accuracy.

- **Bioinformatics:** Protein classification and cancer classification are included. SVM is used for gene classification, patient classification based on genes, and other medical difficulties.

- **Handwriting Recognition:** SVMs can also be used to recognize handwritten characters for data input and document signature checking.

The main focus in my thesis lies in its applications used in signature-based Intrusion Detection Systems. Here we use input data containing normal and malicious traffic to build a model that can detect when a malicious traffic has been recognized by SVM, an alarm or warning sign will be sent to the user.

2. SVM Parameters Tuning:

There are three parameters that should be considered and tuned when using SVM techniques. These are the kernel, regularization (complexity), and gamma that are adjusted to produce the best hyperplane. Each of these affect the model differently and not choosing the best parameters may cause the model to fail and not produce that high performance and accurate results:

- Kernel: The SVM's kernel is in charge of converting the input data into the required format. SVM kernels include linear, polynomial, and radial basis functions (RBF). We use RBF and the Polynomial function to create a non-linear hyperplane. To separate nonlinear classes in complex applications, more advanced kernels should be used. This transformation can produce accurate classifiers.

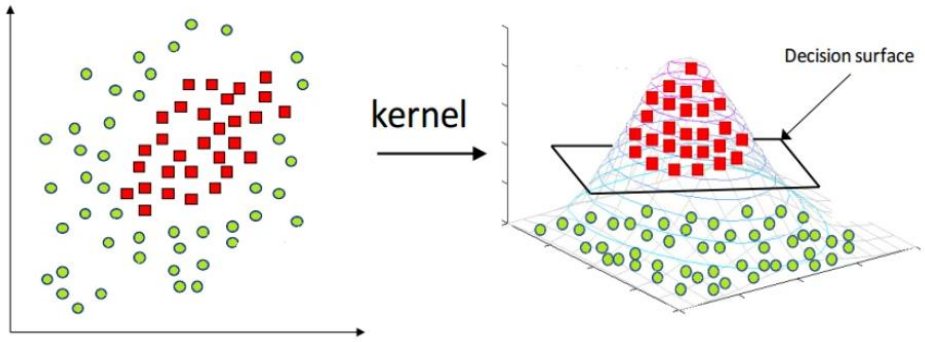

Fig 7. Example of how kernel can classify non-linear data.

- Regularization: It is a method of preventing the model from overfitting by providing more information to it. **Overfitting** problem is said to algorithm that produces high accuracy in classification of training dataset and low accuracy in classification of testing dataset. If algorithm produce low accuracy in both datasets, then it is said to be **Underfitting**. Overfitting model can described in having high training accuracy (low bias) and low testing accuracy (high variance). Underfitting model can described in having low training accuracy (high bias) and low testing accuracy (high variance).

We can keep regularization by adjusting the C parameters. C is a penalty parameter that represents an error or any other type of misclassification. With this misclassification, it is possible to determine how much of the error is actually tolerable. You can do this by nullifying the compensation between the misclassified term and the decision boundary. The parameter C, which is shared by all SVM kernels, deals with misclassification of training examples versus decision surface simplicity. A low C soothes the decision surface, whereas a high C attempts to correctly classify all training examples. This means that a low value of C makes hyperplane underfitting while a high value of C makes hyperplane overfitting. Increasing C will reduce misclassification and margins of hyperplane while decreasing C will have misclassification and more regularization but margins of hyperplane will be maximized. In Fig 3 below an example is shown on some data how C value can affect the hyperplane and thus the model.

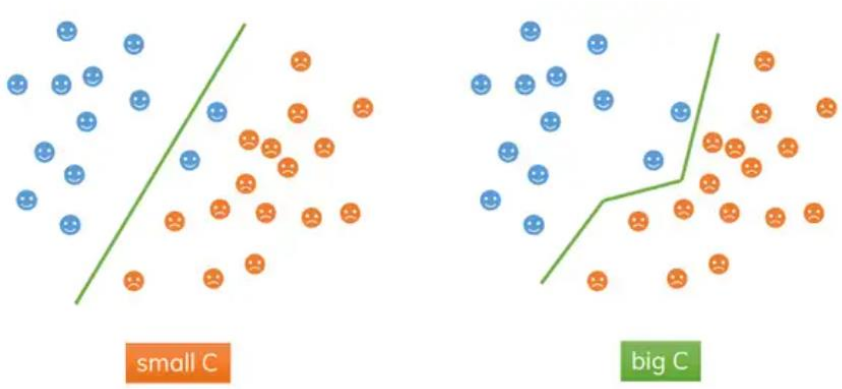

Fig 8. Example of using C.

Regularization can be achieved also by adding a loss function to minimize the cost function of a specified model (ex: linear) in order to optimize the model parameters leading to better results. The Loss function in Logistic Regression can be defined as:

$$L = y \log (wx + b) + (1 - y)\log(1 - (wx + b))$$

These loss functions are classified into two techniques:

1. Ridge Regression (L2 Regularization): It is used to minimize the model's complexity. We compute it by multiplying the squared weight of each individual feature by the lambda. This is called the penalty term. As the value of lambda decrease (tend to zero), the equation will return to its original form. L2 will never reduce any weight to zero and thus no feature elimination will occur. L2 perform better in correlated inputs and will create a robust model. The Loss function equation will be:

$$L = y \log (wx + b) + (1 - y)\log(1 - (wx + b)) + \text{lambda} * ||w||^2_2$$

2. Lasso Regression (L1 Regularization): It is also used to minimize the model's complexity using different penalty term. It is referred as Least Absolute and Selection Operator. The squared weights in L2 are replaced by absolute of the weights for L1. L1 penalty term can reduce some weights into zero and thus eliminating its corresponding feature during model evaluation. Therefore, L1 Regularization can

support us in both feature selection and minimizing overfitting in the model. L1 deals with sparsity and will create a sparser model. The loss function equation will be here:

$$L = y \log (wx + b) + (1 - y)\log(1 - (wx + b)) + \text{lambda} * ||w||_1$$

lambda is a kind of hyperparameter. It is bigger than zero and is known as the regularization constant.

- Gamma: A lower Gamma value will result in a loose fit of the training dataset. A high gamma value, on the other hand, will allow the model to be fit more accurately. A high gamma value only considers nearby points when calculating the hyperplane, whereas a low gamma value considers all data-points when calculating the hyperplane. Fig 4 below an example is shown on some data how Gamma value can affect the hyperplane and thus the model.

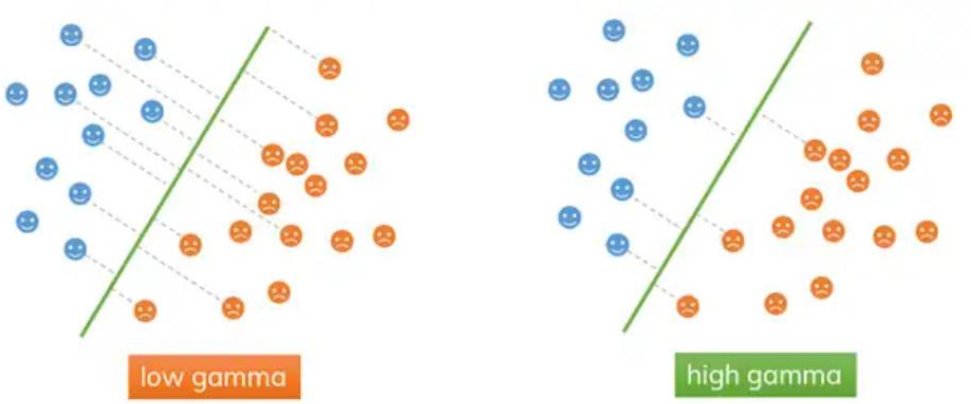

Fig 9. Example of using Gamma.

3. Hyper parameter Optimization using K-fold Cross validation:

It is the process of determining the best hyper parameters for an algorithm. The hyper parameter is used to adjust the algorithm's learning process. The model cannot predict these parameters. They have an impact on the model's performance, speed, and overall quality. In order to examine machine learning models on a small sample of data and identify the best hyper parameters, cross-validation is used as a resampling technique. The word "K-fold" describes how many groups a particular data sample

should be split into. This process is applied to the training datasets. For example if K is set to 5 this means that for 100% of training data will split in 5 groups equally each containing 25% of the train dataset. Each time, one of the k subsets is selected as the test/validation set, and the remaining k-1 subsets are combined to form the training set. The total accuracy (performance) of our algorithm is calculated by averaging the error estimation over all k trials. This minimizes bias (error of training data) since we are utilizing the majority of the data for fitting, and it minimizes variance because the majority of the data is also used in the validation set. The algorithm that holds highest cross validation accuracy and its corresponding hyper parameter can be set as the preferred algorithm.

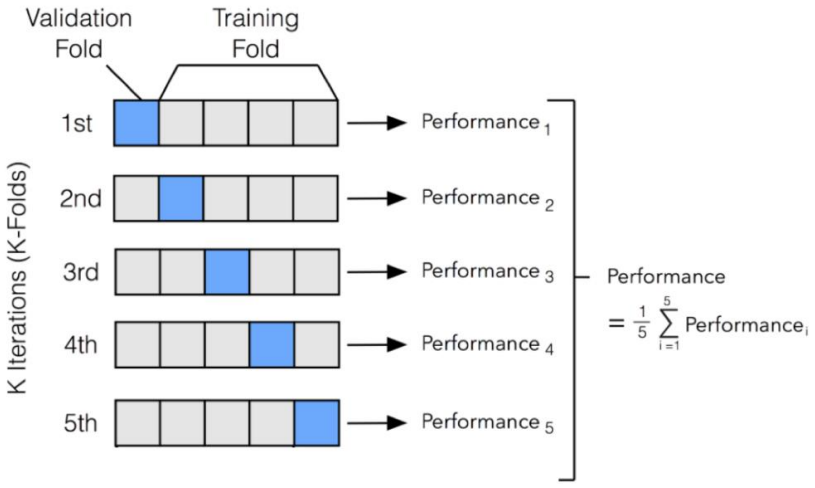

Fig 10. K-fold Cross Validation with K=5.

C. Linear Support Vector Machine (Linear SVM):

The term "linearly separable data" refers to data that can be divided into two groups by a single straight line. Linear SVM is the classifier employed on this type of data. Figure 5 above gives an example of how linearly separable data would look like.

For example, linear SVM can be used to learn a binary classifier that predicts whether an image contains a dog or not. Linear SVMs are very flexible and can be designed with many different loss functions and non-linearities. They also provide

principled estimates of generalization error (generalization error = twice training classification errors). In other words, given some training data c and its corresponding label y, linear SVM identifies two points (c1 , y1) in the feature space for the classifier: one for each possible label. If the classifier cannot distinguish between these two points, this is considered a training error e.

As mentioned in literature review part, SVM has been widely used in cybersecurity studies on different datasets. We can see that linear function has shown that it was efficient and resulted in good performance on some datasets that can classified as a linearly separable data. One of the experiments conducted in research done by was using linear two-class SVM on 6 datasets where one corresponded to normal data frames while the other five comprises both malicious and normal data frames to build a robust anomaly-based IDS. The two-class SVM was trained using 35% of each dataset. The training dataset was chosen at random in order to enhance the likelihood of having a realistic distribution of malicious and regular traffic. Following the training procedure, one model is created for each dataset. The classification is then performed using the remaining 65 percent of the dataset. The table below illustrates the results of these models:

Linear Two-Class SVM				
Dataset	**DR (%)**	**FPr (%)**	**FNr (%)**	**OSR (%)**
Attack01	100	0	0	100
Attack02	100	0	0	100
Attack03	100	0	0	100
DeAuth	100	0	0	100
Probing	98.78	16.6	1.22	83.4

Table 10. Detection Results: LINEAR TWO-CLASS SVM.

Each dataset was explained in literature review part under ‘‘**Support Vector Machine for Network Intrusion and Cyber-Attack Detection**’’[8]. The performance measurements used in the detection analysis are: Detection Rate (DR), False Positive Rate (FPr), False Negative Rate (FNr), and Overall, Success Rate (OSR). The results clearly showed how much efficient can linear SVM be used in IDS for the first 4 datasets where the OSR and DR showed 100% but not for the Probing dataset where the results for OSR and DR showed 83.4% and 98.78%. Another experiment using linear SVM can be found in research done by. Despite that Deep Neural Network algorithm outperformed all other classification methods, Linear SVM

was one of the tested methods that gave quite good scores of 0.811 for accuracy and 0.994 for precision. The detailed research of these experiments can be found in literature review part 5. Linear SVM is quite powerful method and fast to be used in an IDS for detecting abnormal traffic and cyber-attacks.

D. Gaussian Radial Basis Function (RBF):

Non-Linear SVM is used for non-linearly separated data, meaning that if a dataset cannot be classified using a straight line, it is non-linear data and the Non-Linear SVM classifier can be applied. This classifier is called a kernel function and kernels can give more accuracy and better performance than using standard linear SVM function. RBF is the one of the kernels that were used in experiments for many researches in making a robust IDS. Nevertheless, RBF showed very good results and proved to be efficient on some datasets but not on all of datasets. Unlike linear SVM, RBF kernel depends on two parameters which are 'C' and 'Gamma'. These parameters should be tuned together in order to have best results for the model or else the model would fail and the accuracy will be very bad if of one these parameters have been assigned a wrong value. The gamma parameter has a significant impact on the model's behavior. If gamma is too big, the radius of the support vectors' zone of effect only covers the support vector itself, and no amount of regularization using C can avoid overfitting. When gamma is very tiny, the model is too limited and unable to describe the data's complexity or shape and will behave like a linear model. The time it takes to train a model with a very high C value is usually longer. Lower C values, on the other hand, tend to result in more support vectors, which might delay prediction time. As a result, decreasing C needs a trade-off between training and prediction time. The figure below represents how a RBF will act on some non-linear data.

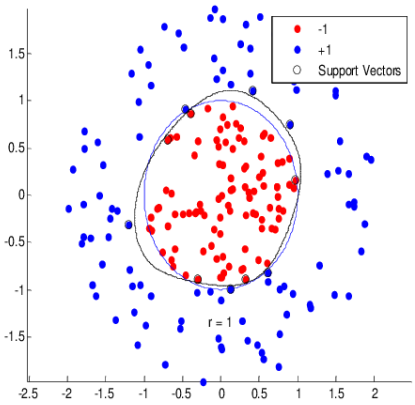

Fig 11. RBF Example on Random Data.

The network traffic that contains data which are examined by IDS are not always considered linear. Actually, real time data are non-linear, and that's a main reason for considering kernel functions in SVM. Although linear SVM can give very good performance in examining datasets that passes through IDS, researchers would prefer using the non-linear techniques of a SVM. By this way examination can be more realistic. RBF kernel was a technique that was tested in research of [8] which used the same steps that it did with the linear SVM experiment I mentioned above. The table below shows the results:

Non-Linear Two-Class SVM with Gaussian Radial Basis				
Dataset	**DR (%)**	**FPr (%)**	**FNr (%)**	**OSR (%)**
Attack01	62.5	0	37.5	99.66
Attack02	99.52	0	0.48	99.97
Attack03	93.51	0	6.49	99.94
DeAuth	97.78	0	2.22	99.25
Probing	97.85	18.32	2.15	81.67

Table 11. Detection Results: Non-Linear Two-Class SVM with RBF.

As seen from the table the RBF performed very well with the different datasets except with the first dataset labeled 'Attack01'. This DR% was very bad due to the overlapping between malicious and non-malicious instances for the metrics SEQ and ΔTime which features in the dataset that made the model unable to give correct classification. Another research paper that I mentioned before that used different machine learning algorithms on KDDCup-'99' dataset used also RBF kernel in SVM

.The results of the model was very close to that of linear SVM model with 0.811 for accuracy and 0.992 for precision. RBF in SVM can be used on various datasets examined in an IDS that is considered to be fastest algorithm in non-linear SVM techniques.

E. Logistic Regression Analysis:

Another supervised learning approach used to solve classification issues is logistic regression (LR). The logistic regression technique uses binary variables such as 0 or 1, Yes or No, True or False, Spam or not spam, and so on. It is a probability-based predictive analytical process. The logistic (sigmoid) function, which is a complex cost function, is applied in logistic regression. In addition, LR is special in ML because it has the ability to classify linear and continuous data. In logistic regression, the sigmoid function is applied to model the data between 0 and 1. A sigmoid function produce a S-shaped curve and can be written as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

The main difference between LR and linear SVM is that the hyperparameter produced (decision boundary) by LR is S-shaped while in linear SVM is a linear straight line for binary classifications. The concept of threshold levels is used, where values over the threshold level are rounded up to 1, while values below the threshold level are rounded down to 0.There are three types of logistic regression: Binary (0, 1), Multiclass (cat, dog, and fish), and Ordinal (low, medium, and high) [14]. For an IDS, binary logistic regression can be implemented in order to classify normal and malicious data.

F. Deep Learning:

Deep learning is a subset of artificial intelligence based on the machine learning. Deep learning has a working principle of imitating the human brain. It is essentially a machine learning method that employs a large number of nonlinear processing units to do feature extraction and modification. Each succeeding layer uses

the output from the preceding layer as its input. Neural Networks are used to implement deep learning.

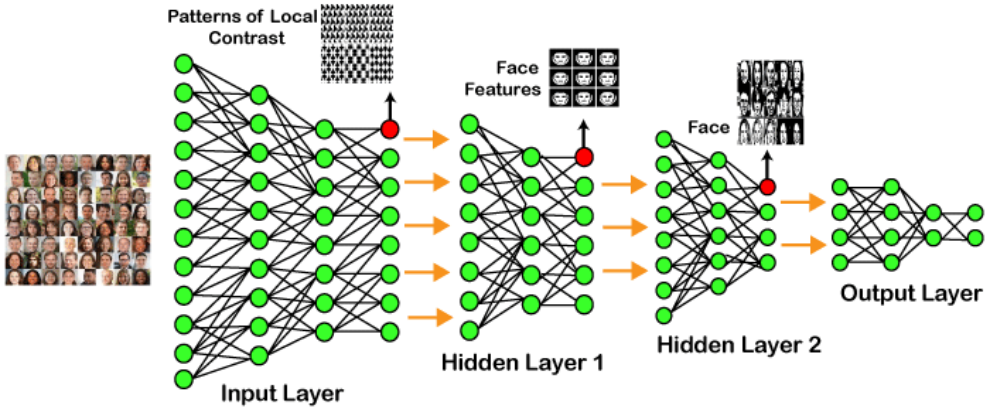

Fig 12. DL Example.

Deep Learning methods are classified into three groups according to the type of data it deals with:

- Deep Multi-Layer Perceptron (DMLP) or Artificial Neural Network (ANN)
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Reinforcement Learning (RL)

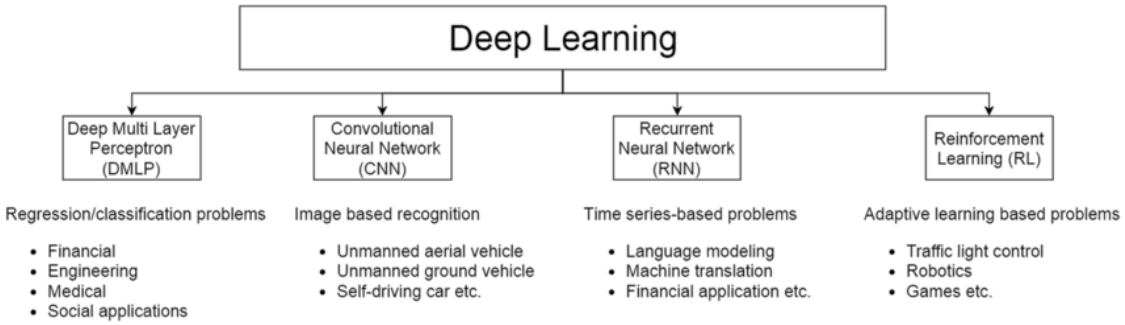

Fig 13. Deep learning Methods.

Deep Learning has shown throughout many experiments and researches its effectiveness in IDS. Many experiments with different conditions and tools were conducted on KDD Cup dataset for building a strong IDS that can differentiate

between normal and attack packets. These experiments shown in the table below were using different Deep Learning algorithms and the results of detection rates (DR) and false alarm rates (FAR) were calculated accordingly:

Deep learning model	Detection Rates (DR)	False Alarm Rate (FAR)
DNN [15]	99%	0.08%
RNN [16]	98.88%	10.04%
DBN [17]	92.33%	0.76%

Table 12. Test Results of Different DL models.

III. PROPOSED METHODS

A. Experiment Setting:

The computer configuration involved in this paper's experiment is as follows: CPU i7-7500U, 12 GB of memory, 500 GB SSD, installed Windows 10 operating system, using LIBLINEAR and LIBSVM libraries on MATLAB 2020.

In this research, numerous network intrusion classification tests are carried out using different machine learning algorithms, with each dataset including normal (negative) and a variety of attack (positive) samples. A multiclass combination matrix is formed when each model is applied to given datasets to display the model's performance. This matrix is called confusion matrix and it stores the records of actual and predicted results of each class in the datasets. The confusion matrix generates four major outcomes: true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs).

- The right predictors of a normal sample are **TN**.
- **FN** is the number of samples of the attack class that were incorrectly classified as normal.
- The right predictors of an attack sample are **TP**.
- **FP** is the number of samples of the normal class that were incorrectly classified as attack.

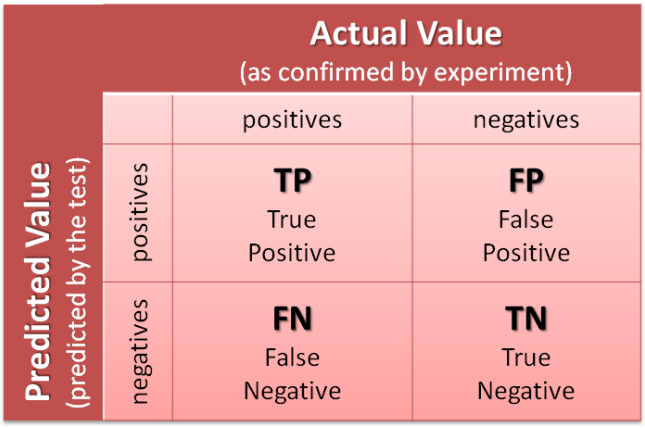

Fig 14. Confusion Matrix For Binary Datasets.

These outcomes were used to calculate four parameters to compare the performance of each model:

1. Accuracy: It refers to the number of samples successfully categorized by the classifier.

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

2. Precision: It is the classifier's accuracy, or the rate at which the attack is correctly identified from all samples categorized as an attack in the test set.

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

3. Recall: It refers to integrity of the model, detection rate (DR) or sensitivity of the model. It is the successful attack class labeled for each attack sample in the test set.

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

4. F1-Score: It refers to the harmonic mean of Precision and Recall.

$$F - \text{Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

B. Experiment Dataset: KDD-CUP 99 Dataset:

This research uses the widely known network intrusion dataset KDD-CUP 99 as an experimental dataset to check the efficiency of the network intrusion detection model provided in this paper. KDD-CUP 99 is a widely used dataset in the field of network intrusion detection [18]. In this paper, two experiments were designed in MATLAB using two different libraries on the same dataset and then results of performance and efficiency were recorded for comparison reasons.

The KDD-CUP 99 dataset is actively used in network intrusion detection and is available for download on the official website. MIT Lincoln Labs put up an infrastructure to collect raw TCP dump data for a local-area network (LAN) simulating a normal US Air Force LAN for nine weeks. They ran the LAN as if it were a real Air Force network, but they sprinkled it with various attacks. The training set has around 4.9 million records, whereas the test set contains approximately 300,000 records. For research purposes, only 10% of the training data ***"kddcup.data_10_percent.zip"*** and all testing datasets ***"corrected.zip"*** are used. Each sample is labeled as normal or attacked. To make a realistic measures and results, the training dataset was formed from 24 attack types whereas the testing dataset was formed from 14 attack types different from the training dataset. Also, related attacks are classified into one category, resulting in four primary attack categories*: DoS, Probe, R2L, and U2R.* The distribution in each dataset can be found in the table below:

Data type		Training set	Test set
Normal		97278	60593
Attack	DoS	391458	223298
	Probe	4107	2377
	R2L	1126	5993
	U2R	52	39
Total		**494021**	**292300**

Table 13. KDD-CUP 99 data details.

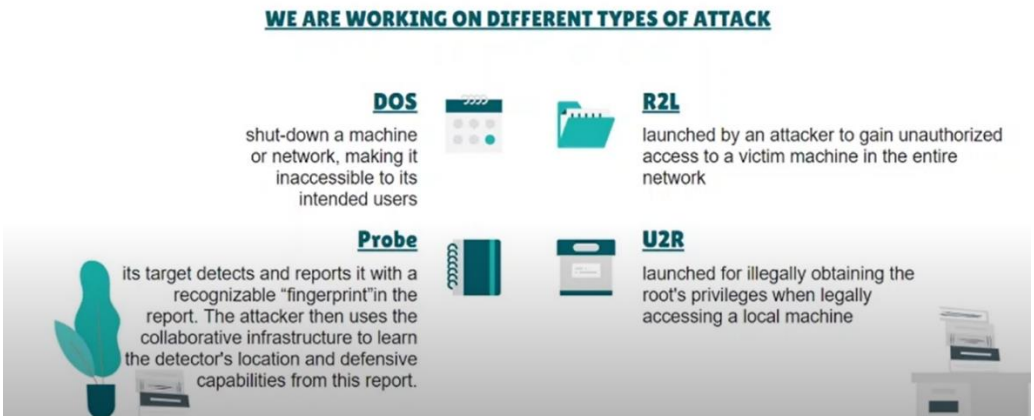

Fig 15. KDD CUP 99 Attack Categories.

Each instance in KDD CUP dataset contains 41 features and 1 additional feature for classification between normal and abnormal (attack/intrusion). The original features collected for the dataset were in forms of continuous, discrete or symbolic and therefore preprocessing was needed in order to be applied into classification models. The preprocessing method included mapping symbolic-valued data to numeric-valued data and scaling. The features can be classified into three groups:

1) **Basic Features:** This group contains the data collected from a TCP/IP connection. The majority of these characteristics result in an implicit delay in detection. These features are referred as higher-level features that can serve in differentiating between legitimate and malicious connections.

2) **Traffic Features:** These features will aid in identifying Probing and DoS attacks. This category includes features calculated with respect to a time frame and can further split into two categories:

a) Same-Host features: The features look at just the connections (packets of data) that have had the same destination host as the current connection in the last two seconds and calculate statistics on protocol behavior, service, and so on.

b) Same-Service features: The features look at just the connections (packets of data) that have had the same service(as the current connection in the last two seconds

3) **Content Features:** These features will aid in identifying R2L and U2R attacks. Unlike most DOS and probing intrusions, there appear to be no sequential

patterns in R2L and U2R attack logs. This is due to the fact that DOS and probing attacks include a large number of connections to a single host(s) in a short period of time, whereas R2L and U2R intrusions are encoded in data sections.

Feature Name	*Description*	*Type*
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of "wrong" fragments	continuous
urgent	number of urgent packets	continuous

Table 14. Basic Features.

Nr	Features	
	Name	Description
1	duration	duration of connection in seconds
2	protocol_type	connection protocol (tcp, udp, icmp)
3	service	dst port mapped to service (e.g. http, ftp, ..)
4	flag	normal or error status flag of connection
5	src_bytes	number of data bytes from src to dst
6	dst_bytes	bytes from dst to src
7	land	1 if connection is from/to the same host/port; else 0
8	wrong_fragment	number of 'wrong' fragments (values 0,1,3)
9	urgent	number of urgent packets
10	hot	number of 'hot' indicators (bro-ids feature)
11	num_failed_logins	number of failed login attempts
12	logged_in	1 if successfully logged in; else 0
13	num_compromised	number of 'compromised' conditions
14	root_shell	1 if root shell is obtained; else 0
15	su_attempted	1 if 'su root' command attempted; else 0
16	num_root	number of 'root' accesses
17	num_file_creations	number of file creation operations
18	num_shells	number of shell prompts
19	num_access_files	number of operations on access control files
20	num_outbound_cmds	number of outbound commands in an ftp session
21	is_hot_login	1 if login belongs to 'hot' list (e.g. root, adm); else 0
22	is_guest_login	1 if login is 'guest' login (e.g. guest, anonymous); else 0
23	count	number of connections to same host as current connection in past two seconds
24	srv_count	number of connections to same service as current connection in past two seconds
25	serror_rate	% of connections that have 'SYN' errors
26	srv_serror_rate	% of connections that have 'SYN' errors
27	rerror_rate	% of connections that have 'REJ' errors
28	srv_rerror_rate	% of connections that have 'REJ' errors
29	same_srv_rate	% of connections to the same service
30	diff_srv_rate	% of connections to different services
31	srv_diff_host_rate	% of connections to different hosts
32	dst_host_count	count of connections having same dst host
33	dst_host_srv_count	count of connections having same dst host and using same service
34	dst_host_same_srv_rate	% of connections having same dst port and using same service
35	dst_host_diff_srv_rate	% of different services on current host
36	dst_host_same_src_port_rate	% of connections to current host having same src port
37	dst_host_srv_diff_host_rate	% of connections to same service coming from diff. hosts
38	dst_host_serror_rate	% of connections to current host that have an S0 error
39	dst_host_srv_serror_rate	% of connections to current host and specified service that have an S0 error
40	dst_host_rerror_rate	% of connections to current host that have an RST error
41	dst_host_srv_rerror_rate	% of connections to the current host and specified service that have an RST error
42	connection_type	

Table 15. KDD CUP Features.

C. MATLAB Libraries: LIBSVM & LIBLINEAR:

LIBSVM (released in 2000) and LIBLINEAR (released in 2007) are two famous open-source machine learning libraries that were developed at National Taiwan University. LIBSVM is a kernelized support vector machine (SVM) implementation that supports classification and regression using the Sequential minimum optimization (SMO) technique. LIBLINEAR provides linear SVMs and logistic regression models trained using the coordinate descent approach. Both libraries are designed to work on different interfaces like Python, Ruby, and MATLAB. LIBSVM's purpose is to make it easier for users from other domains to employ SVM as a tool. LIBLINEAR is a linear classifier that features automatic parameter selection and adds regularization terms (L2 and L1) which means a more robust model. It is considered to be faster than LIBSVM since no kernels are implemented in building LIBLINEAR models. Both have a simple interface that allows users to simply integrate them into their own programs. LIBSVM can work with linear and non-linear data while LIBLINEAR can work only with linear data. In this research, KDD CUP 99 dataset is considered a linear dataset and thus results can be computed through LIBLINEAR models.

D. Experimental Results:

The experiments carried out are divided into two parts: the first one I used LIBSVM library to build two different models using Linear SVM and RBF. The second one I used LIBLINEAR library to build three different models using Logistic Regression with L1 Regularization, Logistic Regression with L2 Regularization, Linear SVM with L1 and L2 Regularization. The percentages of parameters evaluated from the results were: Time Consumed - Accuracy - Precision - Detection Rate (Recall) - F1-Score.

	Training Accuracy	Testing Accuracy	Training Precision	Testing Precision	Training DR	Testing DR	Training F1-score	Testing F1-score	Time
Linear SVM	80.47	80.45	69.26	64.20	51.32	51.17	58.96	56.95	24 hr
RBF	99.98	81.48	99.90	75.62	99.95	88.46	99.69	81.53	30 hr

Table 16. LIBSVM Experiment Results (%).

	Training Accuracy	Testing Accuracy	Training Precision	Testing Precision	Training DR	Testing DR	Training F1-score	Testing F1-score	Time
L2 - LR	99.68	92.03	99.44	85.53	99.56	94.43	99.50	89.76	38.06 sec
L1 - LR	99.65	91.13	99.37	84.36	99.55	93.87	99.46	88.86	54.02 sec
L1-L2 Linear SVM	99.08	81.28	97.94	75.27	99.23	87.79	98.58	81.05	27.50 sec

Table 17. LIBLINEAR Experiment Results (%).

E. Analysis and Comparison:

In the LIBSVM experiment, RBF model produced better results for training and testing data from the LINEAR SVM model but the RBF model took so much time to give these good results as it took it 30 hours while the LINEAR SVM model took only 24 hours. Here RBF model shows an example of overfitting model having high training accuracy and low testing accuracy. The goal of IDS is to detect intrusions correctly in the fastest time possible in order for the user to take actions of preventing, defending and deleting these attacks before it causes any damage to the devices. In the LIBLINEAR experiment, the training results in all three models were very close to each other and above 95%, which are very high results. The differences were noticed in the testing results and the time consumed by each model. L1-L2 Linear SVM model was the fastest as it required only 27.5 seconds building model and producing results, but this was not enough for it to be considered the best as the testing results in the other two models surpassed the Linear SVM testing results. Comparing L2-LR and L1-LR models, it is clearly visible that L2-LR outperformed L1-LR in both testing results and

time consumed where it took 38.06 seconds to produce testing results with 92.03% of accuracy. Between the libraries, LIBLINEAR results are the best due to the low time consumed as all results were collected in less than a minute whereas LIBSVM needed more than a day to collect the results of each model. Another advantage is that LIBLINEAR library models always produced high results for training and testing unlike to LIBSVM where results were in some models high and in other models low. L2-LR model from LIBLINEAR library is the winner for these experiments for being a robust model as it has fulfilled the requirements for building an IDS that are high training and testing results in addition to low time consumption.

IV. CONCLUSION

The identification of network intrusions is crucial for cyber security. Despite the fact that there has been a lot of research on network intrusion detection recently, this topic has received relatively little in-depth study, particularly when it comes to multiclass network intrusion detection. In this paper, binary IDS models based on different machine learning algorithms are proposed, to learn the best machine learning algorithm and study the effect of these algorithms on an IDS dataset. The experiments apply KDD-CUP99 dataset and compares the experimental results with the machine learning methods of LINEAR SVM, RBF, L1-Regularized LR, L2-Regularized LR, and L1-L2 Regularized LINEAR SVM. The experimental results indicate that, in comparison to other proposed methods, the network intrusion detection model used in this paper, which uses L2-Regularized LR from the LIBLINEAR library, improves accuracy, consumes less time, and generates better detection results for the detection of unknown attacks.The contributions of this thesis to the cyber security field are summarized in the following:

- Building a robust model that can detect intrusions in a very short period.
- Creating an automated system with high accuracy in detection.
- Being able to detect the new attacks presented in the testing data.
- Regularization in machine learning techniques used for building an IDS is essential for building more accurate and faster model.

This research has fully reinforced the benefits of machine learning techniques in IDS. Because the techniques are trained on a previous benchmarking dataset, as noted multiple times in this study, this is a drawback for this research. However, this might be eliminated by employing a new dataset containing the essences of the most recent attack techniques prior to the actual installation of this method to existing network systems to confirm the agility of the algorithms' real-world capabilities. According to the scientific findings of this thesis, machine learning techniques are a

robust approach for cyber security tasks, but even though their performance on experimental datasets is distinctive, it is still necessary to apply them to network traffic that contains more sophisticated and modern attack types in real-time. This is one of the possible directions for IDS research and will thus remain a future topic.

V. MATLAB CODES

1) Linear Support Vector Machine (linear SVM) (LIBSVM LIBRARY):

```
clear;

tic % timer on

tr=load('kddtrain.csv'); % reading training data

tr_label=tr(:,1);% the first column is label column

tr_data=tr(:,2:end); % rest of the data is training data


te=load('kddtest.csv'); % reading testing data

te_label=te(:,1);% the first column for the label

te_data=te(:,2:end);% the rest of the data is testing data


model=svmtrain(tr_label,tr_data,'-s 0 -t 0 -c 1');% building model based on training data linear svm is selected for parameters complexity (c=1)

[tr_dec,tr_accuracy_,tr_dec_values]=svmpredict(tr_label,tr_data, model);% accurracy for training

[te_dec,te_accurracy_,te_dec_values]=svmpredict(te_label,te_data, model);% accurracy for testing

toc % timer off
```

```
% Training data Parameters %

cm_tr=confusionmat(tr_label,tr_dec); % training confusion matrix

cm_tr_t= cm_tr';

diagonal= diag(cm_tr_t);

sum_of_rows= sum(cm_tr_t,2);

precision= diagonal./sum_of_rows;

overall_precision_tr= mean(precision) % training precision %

sum_of_columns =sum(cm_tr_t,1);

recall=diagonal./sum_of_columns';

overall_recall_tr =mean(recall)        % training recall %

fl_score_tr=2*((overall_precision_tr*overall_recall_tr)/(overall_precision_tr+overall
_recall_tr)) % training f1-score %

% Testing data Parameters %

cm_te=confusionmat(te_label,te_dec); % testing confusion matrix

cm_te_t= cm_te';

diagonal= diag(cm_te_t);

sum_of_rows= sum(cm_te_t,2);

precision= diagonal./sum_of_rows;

overall_precision_te= mean(precision)  % testing precision %
```

```
sum_of_columns =sum(cm_te_t,1);

recall=diagonal./sum_of_columns';

overall_recall_te =mean(recall)        % testing recall %

fl_score_te=2*((overall_precision_te*overall_recall_te)/(overall_precision_te+overal
l_recall_te))   % testing f1-score
```

2) Gaussian Radial Basis Function (RBF) (LIBSVM LIBRARY):

```
clear;

tic % timer on

tr=load('kddtrain.csv'); % reading training data

tr_label=tr(:,1);% the first column is label column

tr_data=tr(:,2:end); % rest of the data is training data


te=load('kddtest.csv'); % reading testing data

te_label=te(:,1);% the first column for the label

te_data=te(:,2:end);% the rest of the data is testing data


model=svmtrain(tr_label,tr_data,'-s 0 -t 2 -c 1 -g 1');% building model based on
training data rbf-svm is selected for parameters complexity (c=1) and gamma (g=1)

[tr_dec,tr_accuracy_,tr_dec_values]=svmpredict(tr_label,tr_data, model);%
accurracy for training

[te_dec,te_accurracy_,te_dec_values]=svmpredict(te_label,te_data, model);%
accurracy for testing
```

```
toc % timer off



% Training data Parameters %

cm_tr=confusionmat(tr_label,tr_dec); % training confusion matrix

cm_tr_t= cm_tr';

diagonal= diag(cm_tr_t);

sum_of_rows= sum(cm_tr_t,2);

precision= diagonal./sum_of_rows;

overall_precision_tr= mean(precision) % training precision %

sum_of_columns =sum(cm_tr_t,1);

recall=diagonal./sum_of_columns';

overall_recall_tr =mean(recall)        % training recall %

fl_score_tr=2*((overall_precision_tr*overall_recall_tr)/(overall_precision_tr+overall
_recall_tr)) % training f1-score %



% Testing data Parameters %

cm_te=confusionmat(te_label,te_dec); % testing confusion matrix

cm_te_t= cm_te';

diagonal= diag(cm_te_t);
```

```
sum_of_rows= sum(cm_te_t,2);

precision= diagonal./sum_of_rows;

overall_precision_te= mean(precision)  % testing precision %

sum_of_columns =sum(cm_te_t,1);

recall=diagonal./sum_of_columns';

overall_recall_te =mean(recall)        % testing recall %

fl_score_te=2*((overall_precision_te*overall_recall_te)/(overall_precision_te+overal
l_recall_te))   % testing f1-score %
```

3) L2-Regularized Logistic Regression (L2-LR) (LIBLINEAR LIBRARY):

```
clear all;

tic % timer on

tr=load('kddtrain.csv'); % reading training data

tr_label=tr(:,1);% the first column is label column

tr_data=tr(:,2:end); % rest of the data is training data

sp_tr_data=sparse(tr_data); % train data in sparse format


te=load('kddtest.csv'); % reading testing data

te_label=te(:,1);% the first column for the label

te_data=te(:,2:end);% the rest of the data is testing data

sp_te_data=sparse(te_data); % test data in sparse format
```

```
model=train(tr_label,sp_tr_data,'-s 0 -c 1'); % building model based on training data
L2-LR is selected for parameters complexity (c=1)

[tr_dec,tr_accuracy_,tr_dec_values]=predict(tr_label,sp_tr_data, model);% accurracy
for training

[te_dec,te_accurracy_,te_dec_values]=predict(te_label,sp_te_data, model);%
accurracy for testing

toc % timer off



% Training data Parameters %

cm_tr=confusionmat(tr_label,tr_dec); % training confusion matrix

cm_tr_t= cm_tr';

diagonal= diag(cm_tr_t);

sum_of_rows= sum(cm_tr_t,2);

precision= diagonal./sum_of_rows;

overall_precision_tr= mean(precision) % training precision

sum_of_columns =sum(cm_tr_t,1);

recall=diagonal./sum_of_columns';

overall_recall_tr =mean(recall)        % training recall

fl_score_tr=2*((overall_precision_tr*overall_recall_tr)/(overall_precision_tr+overall
_recall_tr)) % training f1-score %
```

```
% Testing data Parameters %

cm_te=confusionmat(te_label,te_dec); % testing confusion matrix

cm_te_t= cm_te';

diagonal= diag(cm_te_t);

sum_of_rows= sum(cm_te_t,2);

precision= diagonal./sum_of_rows;

overall_precision_te= mean(precision)  % testing precision

sum_of_columns =sum(cm_te_t,1);

recall=diagonal./sum_of_columns';

overall_recall_te =mean(recall)        % testing recall

fl_score_te=2*((overall_precision_te*overall_recall_te)/(overall_precision_te+overal
l_recall_te))   % testing f1-score
```

4) L1-Regularized Logistic Regression (L1-LR) (LIBLINEAR LIBRARY):

```
clear all;

tic % timer on

tr=load('kddtrain.csv'); % reading training data

tr_label=tr(:,1);% the first column is label column

tr_data=tr(:,2:end); % rest of the data is training data

sp_tr_data=sparse(tr_data); % train data in sparse format
```

```
te=load('kddtest.csv'); % reading testing data

te_label=te(:,1);% the first column for the label

te_data=te(:,2:end);% the rest of the data is testing data

sp_te_data=sparse(te_data); % test data in sparse format




model=train(tr_label,sp_tr_data,'-s 6 -c 1');% building model based on training data
L1-LR is selected for parameters complexity (c=1)

[tr_dec,tr_accuracy_,tr_dec_values]=predict(tr_label,sp_tr_data, model);% accurracy
for training

[te_dec,te_accurracy_,te_dec_values]=predict(te_label,sp_te_data, model);%
accurracy for testing

toc % timer off



% Training data Parameters %

cm_tr=confusionmat(tr_label,tr_dec); % training confusion matrix

cm_tr_t= cm_tr';

diagonal= diag(cm_tr_t);

sum_of_rows= sum(cm_tr_t,2);

precision= diagonal./sum_of_rows;

overall_precision_tr= mean(precision) % training precision
```

```
sum_of_columns =sum(cm_tr_t,1);

recall=diagonal./sum_of_columns';

overall_recall_tr =mean(recall)       % training recall

fl_score_tr=2*((overall_precision_tr*overall_recall_tr)/(overall_precision_tr+overall
_recall_tr)) % training f1-score %

% Testing data Parameters %

cm_te=confusionmat(te_label,te_dec); % testing confusion matrix

cm_te_t= cm_te';

diagonal= diag(cm_te_t);

sum_of_rows= sum(cm_te_t,2);

precision= diagonal./sum_of_rows;

overall_precision_te= mean(precision)  % testing precision

sum_of_columns =sum(cm_te_t,1);

recall=diagonal./sum_of_columns';

overall_recall_te =mean(recall)       % testing recall

fl_score_te=2*((overall_precision_te*overall_recall_te)/(overall_precision_te+overal
l_recall_te))   % testing f1-score
```

5) L1-L2-Regularized Linear Support Vector Machine (L1-L2-Linear SVM) (LIBLINEAR LIBRARY):

```
clear all;

tic % timer on
```

```
tr=load('kddtrain.csv'); % reading training data

tr_label=tr(:,1);% the first column is label column

tr_data=tr(:,2:end); % rest of the data is training data

sp_tr_data=sparse(tr_data); % train data in sparse format


te=load('kddtest.csv'); % reading testing data

te_label=te(:,1);% the first column for the label

te_data=te(:,2:end);% the rest of the data is testing data

sp_te_data=sparse(te_data); % test data in sparse format



model=train(tr_label,sp_tr_data,'-s 5 -c 1');% building model based on training data
L1-L2-Linear SVM is selected for parameters complexity (c=1)

[tr_dec,tr_accuracy_,tr_dec_values]=predict(tr_label,sp_tr_data, model);% accurracy
for training

[te_dec,te_accurracy_,te_dec_values]=predict(te_label,sp_te_data, model);%
accurracy for testing

toc % timer off

% Training data Parameters %

cm_tr=confusionmat(tr_label,tr_dec); % training confusion matrix

cm_tr_t= cm_tr';
```

```
diagonal= diag(cm_tr_t);

sum_of_rows= sum(cm_tr_t,2);

precision= diagonal./sum_of_rows;

overall_precision_tr= mean(precision) % training precision

sum_of_columns =sum(cm_tr_t,1);

recall=diagonal./sum_of_columns';

overall_recall_tr =mean(recall)        % training recall

fl_score_tr=2*((overall_precision_tr*overall_recall_tr)/(overall_precision_tr+overall
_recall_tr)) % training f1-score %


% Testing data Parameters %

cm_te=confusionmat(te_label,te_dec); % testing confusion matrix

cm_te_t= cm_te';

diagonal= diag(cm_te_t);

sum_of_rows= sum(cm_te_t,2);

precision= diagonal./sum_of_rows;

overall_precision_te= mean(precision)  % testing precision

sum_of_columns =sum(cm_te_t,1);

recall=diagonal./sum_of_columns';

overall_recall_te =mean(recall)        % testing recall
```

```
fl_score_te=2*((overall_precision_te*overall_recall_te)/(overall_precision_te+overal
l_recall_te))   % testing f1-score
```

REFERENCES

BOOKS

AMROLLAHI M., HADAYEGHPARAST S., KARIMIPOUR H., DERAKHSHAN F., SRIVASTAVA G. (2020) **Enhancing Network Security Via Machine Learning: Opportunities and Challenges**. In: Choo KK., Dehghantanha A. (eds) Handbook of Big Data Privacy. Springer, Cham. https://doi.org/10.1007/978-3-030-38557-6_8

A. DOHEREY, A. SINGH AND A. KUMAR (2022), "**Intrusion Detection using Dense Neural Network in Network System**" 2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), pp. 484-488, doi: 10.1109/CyberneticsCom55287.2022.9865436.

AMARUDIN, R. FERDIANA AND WIDYAWAN (2020) "**A Systematic Literature Review of Intrusion Detection System for Network Security: Research Trends, Datasets and Methods**" 2020 4th International Conference on Informatics and Computational Sciences (ICICoS), pp. 1-6, doi: 10.1109/ICICoS51170.2020.9299068

D. PARKER (1998), **Fighting computer crime**. New York: Wiley.

E. BOU-HARB, M. DEBBABI, and C. ASSI, (2014) "**Cyber scanning: A comprehensive survey**" in IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1496-1519.

I. A. ABDULMAJEED AND I. M. HUSIEN, (2022) "**Machine Learning Algorithms and Datasets for Modern IDS Design**" 2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), pp. 335-340, doi: 10.1109/CyberneticsCom55287.2022.9865255.

J. B. and C. J. (2017) "**The promise of machine learning in cybersecurity**", SoutheastCon, 2017,

J. MCHUGH, (2000) "**Testing Intrusion detection systems**" ACM Transactions on Information and System Security (TISSEC), vol. 3, no. 4, pp. 262–294.

M. ABADI, P. BARHAM, J. CHEN, Z. CHEN, A. DAVIS, J. DEAN, M. DEVIN, S. GHEMAWAT, G. IRVING, M. ISARD et AL., (2016) "**Tensorflow: A system for large-scale machine learning.**" in OSDI, vol. 16, pp. 265283.

STOLFO, SALVATORE & LEE, WENKE & PRODROMIDIS, ANDREAS & CHAN, PHILIP. (1999). **Cost-Based modeling and evaluation for data mining with application to fraud and intrusion detection: Results from the JAM project.**

S. KUMAR, S. GUPTA and S. ARORA (2021) "**Research Trends in Network-Based Intrusion Detection Systems: A Review**" in IEEE Access, vol. 9, pp. 157761-157779, 2021, doi: 10.1109/ACCESS.2021.3129775.

W. STALLINGS, **Network Security Essentials Applications and Standards**. Prentice Hall, 2013

Z. ALOM, V. R. BONTUPALLI, and T. M. TAHA (2015), "**Intrusion detection using deep belief networks**" in Proceedings of the 2015 National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, June 2015.

JOURNALS:

J. KIM, N. SHIN, S. Y. JO ET AL., "**Method of intrusion detection using deep neural network**" in Proceedings of the 2017 IEEE International Conference on Big Data and Smart Computing (Big Comp), Jeju, South Korea, February 2017.

J. KIM, H. L. T. THU ET AL., "**Long short term memory recurrent neural network classifier for intrusion detection**" in Proceedings of the 2016 International Conference on Platform Technology and Service (Plat Con), Jeju, South Korea, January 2016.

K. GHANEM, F. APARICIO-NAVARRO, K. KYRIAKOPOULOS, S. LAMBOTHARAN AND J. CHAMBERS, "**Support Vector Machine for Network Intrusion and Cyber-Attack Detection**", 2017 Sensor Signal Processing for Defence Conference (SSPD), 2017.

MAHESHKUMAR SABHNANI AND GURSEL SERPEN."Application of machine learning algorithms to kdd intrusion detection dataset within misuse detection context", **In Proceedings of International Conference on Machine Learning: Models, Technologies, and Applications**, pages 209–215, Las Vegas, Nevada, USA, 2003

R. VIGNESWARAN, R. VINAYAKUMAR, K. SOMAN AND P. POORNACHANDRAN, "**Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security**", 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 201

ELECTRONIC JOURNALS:

BAMBRICK N., (2020). **Support Vector Machines**. [Online] Available at:

https://www.kdnuggets.com/2016/07/support-vector-machines-simpleexplanation.html

De Bari Y., (2019). **The Future of Tomorrow: Automation of**

Cybersecurity. Infosys. [Online] Available at:

https://www.infosys.com/about/knowledgeinstitute/insights/documents/future-tomorrow.pdf

ENISA. (2020). **ENISA Threat Landscape 2020 – Web-based Attacks**.

DOI: 10.2824/552242

[Online] Available at: https://www.enisa.europa.eu/publications/webbased-attack

H. SARJAN, A. AMELI AND M. GHAFOURI, "**Cyber-Security of Industrial Internet of Things in Electric Power Systems**" in IEEE Access, 2022, doi: 10.1109/ACCESS.2022.3202914.

R.-E. FAN, K.-W. CHANG, C.-J. HSIEH, X.-R. WANG, AND C.-J. LIN. **LIBLINEAR: A Library for Large Linear Classification**, Journal of Machine Learning Research 9(2008), 1871-1874. Sage Journals.[Online] Available at: https://journals.sagepub.com/doi/10.1177/000812561986492

SEGAL, E. (2020). **The Impact of AI on Cybersecurity**. IEEE Computer Society. [Online] Available at: https://www.computer.org/publications/tech-news/trends/the-impact-of-aion-cybersecurity

ServReality. (2020). **Artificial Intelligence in Cybersecurity**. [Online] Available at: https://servreality.com/blog/artificial-intelligence-incybersecurity-pros-and-cons

URL-1 "**Regularization in Machine Learning - Javatpoint**", www.javatpoint.com, 2022. [Online]. Available: https://www.javatpoint.com/regularization-in-machine-learning, [Accessed: 10- May- 2022].

URL-2 "**Regression Analysis in Machine learning - Javatpoint**", www.javatpoint.com, 2022. [Online]. Available: https://www.javatpoint.com/regression-analysis-in-machine-learning . [Accessed: 01- Jun- 2022]

URL-3 **Risk Based Security**. (2020). 2019 Year End Report. Data Breach QuickView. [Online] Available at: https://pages.riskbasedsecurity.com/hubfs/Reports/2019/2019%20Year%20End%20Data%20Breach%20QuickView%20Report.pdf

URL-4 National Technology Security Coalition (NTSC). (2020). **Cybersecurity Report 2020**. [Online] Available at: https://www.ntsc.org/assets/pdfs/cyber-security-report-2020.pdf

OTHER RESOURCES

A. HENDERSON. (2019). **The CIA Triad: confidentiality, integrity, availability**. Panamore Institute. [Online] Available at: http://panmore.com/the-cia-triad-confidentiality-integrity-availability

CHIH-CHUNG CHANG AND CHIH-JEN LIN, **LIBSVM : a library for support vector machines**. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

J. DE GROOT. (2020). **What is Cybersecurity? Definitions, Best Practices and More**. Digital Guardian. [Online] Available at: https://digitalguardian.com/blog/what-cyber-security

URL-1 "**OWASP TOP 10 2017 COVERAGE The Ten Most Critical Web Application Security Risks JUNE 2017**" [Online] Available: https://docs.broadcom.com/doc/web-application-firewall-owasp-top-10-2017-coverage-en (Accessed: 05-04-2022).

URL-2 "**Evaluation of Machine Learning Algorithms for Intrusion Detection System**", Medium, 2022. [Online]. Available: https://medium.com/cuelogic-technologies/evaluation-of-machine-learning-algorithms-for-intrusion-detection-system-6854645f9211 [Accessed: 05- May- 2022].

URL-3 "**Cross-Validation in Machine Learning**", Medium, 2022. [Online]. Available: https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f [Accessed: 30- May- 2022].

URL-4 University of California, **KDD CUP 1999 Data Set**, University of California, Irvine, CA, USA, 1999, [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/ [Accessed: 01- Jul- 2022]

URL-5 **KDD Cup 1999:Computer network intrusion detection** [Online]. Available: https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data [Accessed: 05-Mar- 2022].

URL-6 University Of California. **The UCI KDD Archive**, University of California. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/task [Accessed: 01- Jul- 2022]

URL-7 Cybersecurity & Infrastructure Security Agency (CISA). (2019). **What is Cybersecurity**. [Online] Available at: https://us-cert.cisa.gov/ncas/tips/ST04-001

URL-8 IBM. (2020). **Cost of a Data Breach Report 2020**. [Online] Available at: https://www.ibm.com/security/data-breach

URL-9 Expert.ai. (2020). **What is Machine Learning?** [Online] Available at: https://www.expert.ai/blog/machine-learning-definition/

URL-10 **Intrusion Detection System (IDS): Signature vs AnomalyBased, 2021**. [Online] Available at: https://www.n-able.com/blog/intrusiondetection-system

Self-starter and dedicated engineer; professional in analyzing and investigating a variety of cybersecurity issues. An individual with in-depth knowledge of cybersecurity and public safety. Looking forward to joining a fast-paced company to apply cybersecurity, networking, and attack vectors. A cybersecurity engineer with expertise to all networking issues, hacking, and protecting organizational data. In addition to knowledge in electrical and electronic processes and applications such as designing, testing, troubleshooting, implementing, developing, and modifying electrical and electronic equipment's.

FADI HAMMAD

Cyber Security Engineer

Electrical Engineer

Electronics Engineer

Expertise

- Ethical Hacking and Threat Modelling.
- Information Security
- Cybersecurity analytics
- Firewall and Intrusion Detection-Prevention Protocols.
- Windows and Kali Linux Operating Systems.
- Identity and Access Management Principles.
- Electrical wiring and planning.
- Machine Learning and Artificial Intelligence

Skills

- MATLAB
- AutoCAD
- Microsoft Office
- Problem-solver
- Team Player
- Self Motivated
- Punctual
- Time management

Languages

- Arabic
- English
- Turkish

Experience

July 2020 - August 2021

Electrical and Electronic Engineer

G. YAMMINE Company - Lebanon

- Lead the construction process and was responsible for electrical wiring, planning, and drafting.
- Followed up all electrical site work and ensured the work quality and project time table.
- Arranged required manpower and resources to execute the plan of work.
- Supervised the installation of Fire Alarm, Access Control, CCTV, and Intercom.
- Coordinated with all sub-contractors and suppliers.
- Helped design and maintain power generation assets, the natural gas pipeline, and the electrical distribution system.

July 2019 – September 2019

Electrical & Electronic Engineer's Assistant,

OGERO COMPANY- LEBANON

Trained as an intern and was responsible for telecommunications

platforms, fixed LTE & IMS, transmission networks, and access networks

Education

September 2020 - Aug 2022

MSc. Electrical and Electronics Engineering

Thesis in Machine Learning Techniques in Cyber Security

ISTANBUL AYDIN University - Turkey

- CGPA: 3.88

September 2016 - June 2020

Bachelor in Electrical and Electronic Engineering

Near East University - Cyprus

- CGPA: 3.74
- Awarded 2nd place in the engineering department in 2020.
- Placed on the Deans High Honors List for Fall 2016-2017 semester, Spring 2017-2018 semester, Fall 2018-2019 semester, and Spring 2018-2019 semester.

Courses & Certifications

Post-Graduate Program in Cyber Security Course - Simplilearn

Certifications

- MIT Schwarzman College of Computing - Cyber security: technology, application and policy
- Build a hacker mindset and defend against future attacks - (CEH Certification Training Course)
- Cyber Security Capstone Project
- Design systems to secure applications, networks, & device
- Design, engineer and manage the overall security posture of an organization - (CISSP Certification Training Course)