Implementing a NIDS System for Protecting Computer and Wireless Networks using various machine learning approaches

Master's Project by

Joseph Eppich

Under supervision of Dr. Hisham Kholidy

In Partial Fulfillment of the Requirements

For the Degree of

Masters in Network And Computer Security

State University of New York, Polytechnic Institute

Utica, New York

ABSTRACT

Implementing a NIDS System for Protecting Computer and Wireless Networks using various machine learning approaches
Joseph Eppich

In modern Wireless Networks, security is critical. With the ever-evolving attacks on wireless networks, both public and private, the use of Network Intrusion Detection Systems is at an all-time high. While NIDS is needed more than ever, its current security structure is starting to show signs of becoming obsolete. With the alarming rate of attacks in the modern digital space, NIDS needs to have a way to react effectively. Setting up NIDS is too slow and can cause many issues when defending against these attacks, leaving wireless networks vulnerable. Three approaches are often discussed NIDS: signature-based, anomaly-based, as well as hybrid. Implementing Machine Learning would fall under an updated version of Anomaly Learning. The hope from these actions is that they will allow new attacks to be caught without user interference. This paper will discuss various forms of Machine Learning, NIDS, and the implementation of both into each other. This paper will discuss our current options in Machine Learning NIDS and explain how they've evolved thus far and the advantages at each stage. This paper, while mainly focusing on the machine learning implementation of NIDS, will touch briefly on how this implementation could strengthen current security in wireless networks.

DECLARATIONS

I declare that this research was done solely by myself with the advisement of Dr.Hisham Kholidy and the work provided in this paper has not been submitted in any form for another degree or diploma at any university or other institute. The information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given accordingly.

Joseph Eppich

05/01/2024

Contents

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background

1.1.1 Intrusion Detection Systems

With the world relying more and more on network frameworks to provide basic needs and services, something as simple as a blackout due to a malicious hacker could be catastrophic to society as a whole[1]. As such, implementing new ways to secure these networks is essential. With the ever-evolving field of machine learning being at the forefront of research lately, it would be foolish not to discuss implementing it into our intrusion detection systems. When talking about implementing machine learning into network intrusion detection systems, it's vital first to understand an intrusion detection system is. Most would describe an Intrusion Detection System as "a type of security software designed to automatically alert administrators when someone or something is trying to compromise information systems through malicious activities or security policy violations"[2] That explanation, found on technopedia.com, can be described as a bare-bones explanation of an IDS. Intrusion Detection Systems are often active in the background, where they are overlooked. IDS' can be broken down into three categories: NIDS, NNIDS, and HIDS. A NIDS is a Network Intrusion Detection System that oversees traffic passing through a network and ensures that traffic is safe. A NNIDS or Network Node Intrusion Detection System works similarly to a NIDS, but the significant difference is that a NNIDS works off of a single node rather than a whole subnet. This approach leads to a more secure single host than a NIDS, which attempts to secure an entire subnet of hosts.The final

version is a HIDS or Host Intrusion Detection System. This version works by monitoring a file system in an operating system and then reporting what is found back to an administrator. Each IDS version is a viable security measure, but we will look at NIDS for this report. The two detection methods used with NIDS are signature-based and anomaly-based. In a signature-based system, human-made rules are implemented to prevent specific traffic from entering a network; anything on that block list is blocked and not allowed into the server. This method can be very tedious but also has the apparent issue of new attacks being vulnerable to it; as a rule, it may only be set for that attack if the user is always up to date. The other method is known as anomaly-based detection, which allows traffic into a server while constantly examining anomalies. If an Anomaly is found, the NIDS will block it and add it to its block list. Anomaly-based detection is often also used outside the scope of a NIDS, where you will usually see places like Water treatment plants[3], Electrical utility companies[4][5][6], and payment processors[7] that use anomaly detection to increase their security. This paper will primarily examine anomaly-based approaches as we discuss how machine learning implementations can make them more robust and secure the network.

1.1.2 Wireless Networks

It would only make sense to explain what these NIDS tend to protect. For this research, we want to demonstrate how machine learning can strengthen the defense of NIDS, specifically in wireless networks. A wireless network is often described as a network of computers that transfers data connections between nodes without the assistance of a wire. These networks are usually used to provide wireless connections to the internet, whether it be for personal or professional reasons. [8] security is essential for a wireless versus a wired connection because you don't know who's connecting to your network, so you can't see every person's intent. For instance, if you were working on a Local Area Network(LAN), you would be able to know who's on the network and thus know who's attacking

you. With wireless technology, we connect outside our typical bubble and open ourselves to the internet. Because of this, we need to be cautious about what information we let into our internal networks. A NIDS will prevent flagged traffic from coming into the network. If an attack does get into a wireless network, it can affect any of the nodes in the network. A node in a wireless network can be anything from a cell phone to a laptop computer; pretty much anything connected to the network is a node.

1.1.3 Machine Learning

Machine learning is classified as a sub-field of artificial intelligence in which a machine is trained to understand its given data. Machine learning can be broken down into four categories: Supervised, unsupervised, semi-supervised, and reinforced learning. In the first category of supervised learning, we train the machine by inputting data to reach a known output; this version of machine learning can be broken down further into Classification and Regression.[9] Classification works by distributing data into categories already defined on the dataset based on their specific features. [9] Regression works by predicting the other features based on the data using the given features. [9] The second category is Unsupervised learning; in this category, we give input data to an algorithm, but the algorithm doesn't know the desired output. Unsupervised learning works by using relations and connections to categorize data accordingly. [9] Similar to Supervised learning, Unsupervised can be broken into two categories: Clustering and Association. In Clustering, the algorithm finds similar data groupings that are unknown. The algorithm determines the relations between the unknown data in the same dataset. The third category is semi-supervised learning; in this category, we take parts from both supervised and unsupervised learning. Semi-supervised is used when labeled data is less than unlabelled data.[9] Unlabeled data is used to deduce information about the data. The final category is Reinforcement learning. In this category, the algorithm learns based on a reward system. [9] It aims to create

the best and shortest path to the goal; by rewarding paths for desired results, we see other paths copy in interest to get a reward. We can see that what we desire to achieve in strengthening NIDS will likely involve using an unsupervised algorithm. Still, the methods we will discuss will use a mix of most of these categories to try and find the best and most robust machine learning implementation.

1.2 Problems

While current NIDS are usually successful in securing a network, they still have their own set of weaknesses. As mentioned previously, there are two detection methods: Signature-based and Anomaly-based. While each method has its own problems, this report focuses on Anomaly-based systems because the goal is to allow machine learning to minimize the human element. Anomaly-based detection systems have the weakness of what is known as "low and slow" attacks. These attacks work slowly, sending traffic over time to appear legitimate. [10] Some commonly known examples of "low and slow" attacks are RUDY, Slowloris, and Ping of Death. RUDY, or R U Dead Yet, is a type of denial of service (DoS) attack that uses a tool to send data to a web server slowly to seize up the server. This is a low and slow attack because the information sent to the server appears completely normal but is just being slowed down to a crawl. [10] The Slowloris attack works on the application layer by using partial HTTP requests to hold a connection open as long as possible to deny service to the endpoint. [10]. This appears as just an open connection to most anomaly-based systems, so it tends to make it through the security measures. Ping of Death is another common attack that works by the attacker sending a packet bigger than the maximum allowed packet size; doing so will cause the receiver to crash or freeze up. Similar to the other two, this attack is harder to recognize since the defense measure will see it as just a packet that was sent that was too large. [11] The hope is that machine learning implementations can recognize these kinds of attacks better than typical Anomaly-based NIDS.

1.3 Objectives

This paper looks at the currently researched implementations of Machine Learning to Network Intrusion Detection Systems. We will look at several attempted implementations, see their strengths and weaknesses, and compare them. We will also be looking at what differentiates the implementations from each other. After comparing the implementations, some thoughts on how they might be able to be improved upon, as well as if I believe they should be expanded upon, will be provided. In Chapter 2, we examined the methodologies used. Here, we will explore each method more deeply, examining the datasets used and the machine learning model being used. In Chapter 3, we will review the researched literature on this topic. In Chapter 4, we will look at and discuss the results of these methods and how they compare. In Chapter 5, we will discuss possible future works with the information we gained. Finally, in Chapter 6, we will conclude and briefly go over what was discussed in this paper.

Chapter 2

Methodology

In this section, we will review the datasets as well as the methods used in the literature being reviewed for this research. By breaking it down piece by piece, we can get a firm understanding before going into the literature review section of this paper. We will start by reviewing the three datasets highlighted in the research literature. These three datasets are KDD Cup '99, NLS-KDD, and UNSW-NB15. After going into depth with those, we will move on to the methods used in the research. Covering them now allows the literature review section to review them with prior knowledge of the processes. Some of the literature that will be discussed will use multiple techniques, so we are allowing for a smoother understanding of the specific implementations by outlining them ahead of time. The methods we will be going over are the following: Random Forest Classifier, SMOTE, Feature Selection, S-NDAE, Deep Neural Network, Non-Linear ReLu, K Nearest Neighbor, Naive Bayes, Support Vector Machine, J48 Classification, Logistic Regression, Decision Tree, PART, Adaptive Boost.

2.1 Research Design

The research design for this paper is quasi-experimental. The aim is to compare multiple types of machine learning implementations to network intrusion detection systems, which allows us to call our research quasi-experimental. The plan is to review each implementation, see what makes it strong, and compare it to the next one. We also aim to examine the ideas used and how to implement them further. By the conclusion of this research, we will give a theoretical implementation that will take the strengths from each piece of research and put them into

one.

2.2 Datasets

While all of these implementations that will be discussed aim to solve the same problem, many of them train based on different datasets. While this can be seen as a problem due to introducing different metrics into a study, it can be helpful to know how some datasets react to various methods when it comes to machine learning. This section will review the datasets used in the researched techniques.

2.2.1 KDD Cup '99 (KDD '99)

The KDD Cup '99 is a dataset created in 1999 using the DARPA'98 IDS evaluation program.[12] The DARPA'98 is seven weeks of TCPdump data network traffic, which yields approximately 5 million connection records. The 4.9 Million records present contain 41 features and are labeled as either standard or attack, with the attacks labeled as specific attacks. The attacks are broken into four categories: Denial of Service, User to Root, Remote to Local, and probing attacks.[12] Each attack has been picked to help test common attacks on a network.

2.2.2 NLS-KDD

This dataset, originating in 2009, was a successor to the KDD'99 dataset. This new dataset implementation aimed to find and solve inherent issues with the KDD '99 dataset [12]. This dataset hopes that it would be affordable to run [13] while allowing the need to not select small portions of the dataset at random to run [13]. This process will make results consistent and comparable across different research sets. The improvement from arguments one and two improves the efficiency of the older dataset by allowing the algorithm not to have to worry about biasing itself against redundant records. This lack of bias also increases the strength of the results from an evaluation. [13] These stronger evaluation results better compare different learning technique methods, as mentioned in item 3.

Item 4, as mentioned before, is about how much more efficient it is compared to KDD regarding sample sizing during the learning.

2.2.3 UNSW-NB15

The UNSW-NB15 dataset originated in 2015 and comprised approximately 2.5 million records.[14] The records in this dataset are classified as either normal or abnormal network traffic. The collection process for this dataset was done using the IXIA Traffic generator on three virtual servers. Two servers were configured to distribute the regular network traffic, while the final server was configured to distribute abnormal data. The attacks were then classified as either Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shell code, or Worms.[14] With this being the most recent out of the three datasets used in the research being reviewed, it is by far the most robust.

2.2.4 CIC-IDS2017

The CIC-IDS2017 dataset originated in 2017 and comprises data collected over five days[15]. The first day of the data contains only benign data to set a neutral ground for testing.[15] The next four days contain attacks in the form of Brute Force FTP and SSH, Heartbleed, Web Attack, Infiltration, Botnet and DDoS.[15] While this dataset isn't used in any of the papers regarding machine learning implementation, but a newer dataset is required when discussing future possible implementations.

2.3 Methods Used

2.3.1 Random Forest Classifier

Random Forest Classifier, or RFC, is a technique that works by combining multiple decision tree models to train for the most potent solution.[16] During training, the RFC will build several trees using a randomly selected subset of features and samples. During the prediction phase, each tree votes for the most popular class.

The class that has the most votes becomes the final prediction. These models tend to be immune to popular adversarial attacks[17]

2.3.2 SMOTE

SMOTE, properly known as the Synthetic Minority-Oversampling Technique, is a technique to address class imbalance.[18] SMOTE addresses this issue by generating synthetic samples to fill the holes for class imbalance.SMOTE works by calculating the K Nearest Neighbor for the minority class of the sample. After calculating the K Nearest Neighbor, SMOTE generates synthetic samples along the line segment that connects the neighbors.[18]

2.3.3 Feature Selection

Feature selection involves choosing a subset of relevant features from a larger group. It aims to hone in on the process for the most desired results. This will work to improve classifications[19] within the model.

2.3.4 S-NDAE

S-NDAE is a deep auto-encoder version, better known as Stacked Deep Auto-encoder. This Deep Auto-Encoder is different because the hidden layers work to sparse, non-negative representations of the input data. This specific technique is stacked because it connects multiple layers of auto-encoders where the previous layer's output is the next layer's input.

2.3.5 Deep Neural Network

Deep Neural Networks, Often called DNN, is a technique that works off multiple layers of nodes known as neurons.[20]. The technique strips away abstract and unneeded features from the input data at each layer. This technique is often used with tasks like image recognition and language processing.

2.3.6 Non-Linear ReLU

ReLU, which stands for Rectified Linear unit, is a commonly used activation function in learning algorithms.[21] ReLU introduces non-linear properties to learning models. ReLU's method removes the issue of a vanishing gradient in the learning model, which allows for efficiency.

2.3.7 K Nearest Neighbor

K Nearest Neighbor, or KNN, classifies new data points using its nearest neighbors.[22] This uses the test data to calculate the distance between the unclassified point and those classified around it. KNN then classifies the points based on a vote, and the classifier will be what the data point is classified as. KNN is often used to detect anomalies in networks[23] due to how it classifies data

2.3.8 Naive Bayes

Naive Bayes, often called NB, is a technique that calculates the probability of a class given the input features.[24] NB assumes the independence of the features, allowing for greater computational efficiency. Naive Bayes is great for things like text classification.

2.3.9 Support Vector Machine

Support Vector Machine, or SVM, works to find a hyperplane that best separates classes in a feature space. [25] SVM is designed to work well with high-dimensional data. SVM is most effective when working with cases where the data provided is not separated linearly and requires a transformation to a higher dimensional space.

2.3.10 J48 Classification

J48 works by recursively building a decision tree by partitioning the data based on features. This is done by partitioning for features that will maximize information

gain. This will continue until the criteria to stop is met; these criteria are typically based on tree depth or reaching peak purity. [26]

2.3.11 Logistic Regression

Logistic Regression is a very standard and commonly used technique. Logistic Regression works by modeling the probability of a binary outcome using logistic data.[27] Despite the name, Logistic Regression isn't a Regression algorithm but instead a Classification algorithm.

2.3.12 Decision Tree

Decision trees are another commonly used technique implemented in other techniques, such as J48 and RFC. This technique works by starting at a single point and traversing down.[28] After traversing down, the method will provide us with the desired outcome based on the provided data.

2.3.13 PART

PART, more properly called Partial Decision Tree, is a technique that uses only a part of a decision tree. PART builds a decision tree by selecting the best attribute based on the statistical significance of that tree. The greatest strength of this technique is that because it only uses a partial tree, the model is easier to interpret

2.3.14 Adaptive Boost

Adaptive Boost, or AdaBoost, combines multiple weak classifiers to create a strong classifier. The weak classifiers are trained on the training data. Before the next iteration, the model boosts the weight of the misclassified instances. The final model is the sum of the classifiers that were predicted to be weak. [29]

Chapter 3

Literature Review

3.0.1 Intrusion Detection and Prevention in Wireless Networks

This paper, written in 2017, is titled "Intrusion Detection and Prevention in Wireless Networks" by S. Al-janabi and I. AlShourbaji. This paper goes into depth with the security measures of NIDS and how they are currently implemented in wireless networks. The most vital information gained here is the strengths and weaknesses of the two IDS methods and current IDS limitations on a wireless network. When looking at the signature-based method, This paper tells us its advantages are its effectiveness and high detection and accuracy rate for known attacks.[30] This paper also tells us the disadvantages of a signature-based are a High false alarm rate for unknown attacks and vulnerabilities and the difficulty in keeping the knowledge base up to date.[30]. Regarding anomaly-based, we are told that its advantages are that it effectively detects new vulnerabilities and is less dependent on the operating system.[30] Its disadvantages are that it takes time to classify attacks, and it is difficult to activate alerts quickly enough.[30]. As for the limitations of IDS in a wireless network, we are told that the most glaring limitation is in anomaly-based detection, which flags false alarms too often, leading to complications in monitoring.[30] This information is essential because it tells us the disadvantages we want to improve upon. The following papers will go into more detail regarding machine learning implementations in hopes of fixing these problems.

3.0.2 Intrusion Detection using Random Forests Classifier with SMOTE and Feature Reduction

The first paper being discussed is called "Intrusion Detection using Random Forests Classifier with SMOTE and Feature Reduction,"[31] as written by A. Tesfahun and D. Lalitha.[31] As the title suggests, this method seeks to improve NIDS using Random Forest Classifiers with SMOTE and Feature Selection. The experiments run in this examination of possible improvements are done in three total phases. The first phase is an implementation using just an RFC. The second phase used RFC and SMOTE testing, which yielded better results. The final incorporated all three: RFC, SMOTE, and Feature selection. The third testing phase proved the most effective, so it moved forward as their proposed model.[31] This paper, published in 2013, gives us an excellent baseline to look at how well the implementations have progressed over the last 11 years. The proposed method here works on a two-pronged framework. It begins with a dataset, in this case, the NLS-KDD dataset, split into training and testing data. The training branch goes onto the labeled data and loads it into a data processor. In this case, the data processor has both SMOTE and Feature Selection, which causes the data to go through SMOTE and then through Feature Selection. After the Training data has been pre-processed, it leaves the pre-processor and builds an RFC to print out results. Similar to the training set, the test set goes through a split, using the now reduced feature list obtained through the training phase. This process continues until the RFC is finished and classification results are given. Several metrics were taken from the experiments used in this paper, including Precision, false positive rate, and detection rate (often called recall). From the results in the paper, we can see that their method showed strong results in both Detection Rate and Precision metrics. It's essential to recognize that while the results look good on paper, this is an early iteration, dating back 11 years. While this information gives us a solid baseline to compare future iterations to, it's not at the forefront of the research.

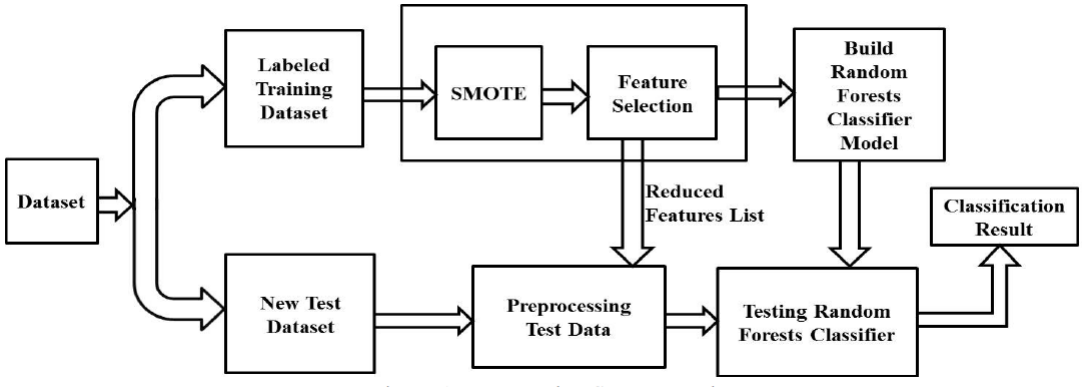

Figure 3.1: Proposed model SMOTE, Feature Reduction, RFC [31]

3.0.3 Deep Learning Approach to Network Intrusion Detection

The next paper takes place four years later, in 2017, and is named "Deep Learning Approach to Network Intrusion Detection"[32]. This paper is written by N Shone, T. Nguyen Ngoc, V. Dinh Phai, and Q. Shi and tackles a new approach that differs from the previous experiments approach. In this approach, we look at a comparison between Non-Symmetric Deep Auto-Encoders(NDAE) and Stacked Non-Symmetric Deep Auto-Encoder(S-NDAE) when implemented in a Deep Belief Network(DBN). This paper's methodology works by first having one NDAE, with an input layer using 41 nodes, which leads to the first hidden layer that uses 14 nodes. The next layer, another hidden layer, has 28 nodes; the following layer will mirror this. All the nodes in the initial NDAE are flowing multi-directional, meaning data can move in both directions. Once the first NDAE is completed, it will transfer its data to the second NDAE, causing the "stacking." this portion of the method is not multi-directional because only the hidden layers can be multi-directional.[32] This second NDAE Works the same way as the previous NDAE works, with the first layer being 14 nodes and the following two being 28. Once the second NDAE is completed, the information is put through an RFC to get results. This paper tracks five metrics: Accuracy, Precision, Recall, False Alarm, and F-score[32]. This method showed a strong result in Precision, which was its best metric during testing.

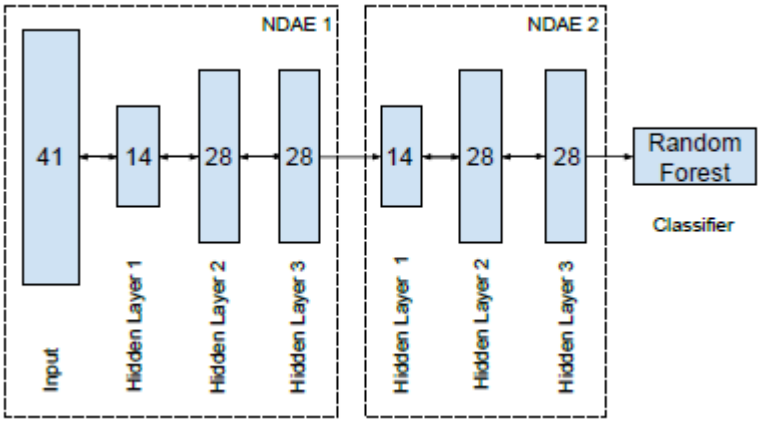

Figure 3.2: Proposed model NDAE [32]

3.0.4 Method of Intrusion Detection using Deep Neural Network

The third paper, also released in 2017, is called "Method of Intrusion Detection using Deep Neural Network".[33] This paper was written by J Kim, N. Shin, S. Y.Jo, and S. H. Kim[33], and seeks to implement Deep Neural Networks into NIDS. This specific implementation works by implementing four hidden layers holding 100 nodes.[33] A back-propagation model is used; this model is also called "Adam," which optimizes DNN Learning. The last important thing to note is that they use an activation function in the form of ReLU. This experiment worked on the KDD Cup '99 dataset but only used a subset of the dataset, approximately 10 percent, for training. The method was trained by first doing 10 percent safe and 90 percent attack data. This pattern continues in increments of 10 percent,20 percent safe data, and 80 percent attack until there's 90 percent safe data and 10 percent attack data. After testing, we are given three metrics: Accuracy, Detection, and False Alarms. This method showed strong results in both Accuracy and Detection.[33]

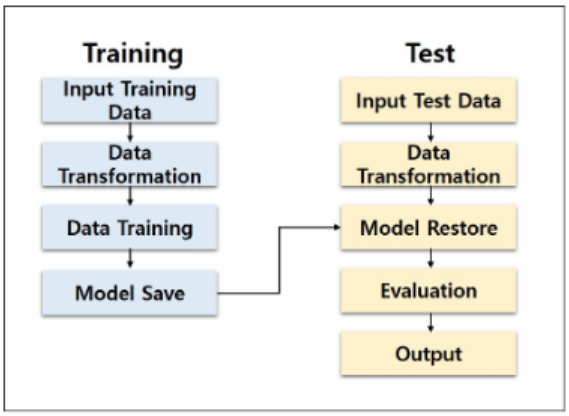

Figure 3.3: Proposed model DNN [33]

3.0.5 An Ensemble Approach for Intrusion Detection System Using Machine Learning Algorithms

The following paper, which I will discuss in my research, was released in 2018. This paper was titled "An Ensemble Approach for Intrusion Detection System Using Machine Learning Algorithms"[34] by R. K. S. Gautam and A. Doegar.[34] The proposed method for this paper is an ensemble method comprising a three-phased approach. The method starts by loading a dataset, which then goes through normalization. The normalization phase uses feature selection through information gain and then sends it to a classifier. The Classifier proceeds to split the data into one of three models. The three models utilized are Naive Bayes, Adaptive Boost, and PART. Tests are then run using them individually, and as an ensemble, the test results show that the ensemble approach has a stronger performance than the individual approach here. While performing well in all fields, this method performed exceptionally well in Precision.

3.0.6 Comparative analysis of Machine Learning algorithms for Intrusion Detection

The fifth paper is " Comparative Analysis of Machine Learning Algorithms for Intrusion Detection" by V. Pai Devidas Adesh N. D., released in 2021.[35] This paper took a unique approach compared to the other documents in that it at-

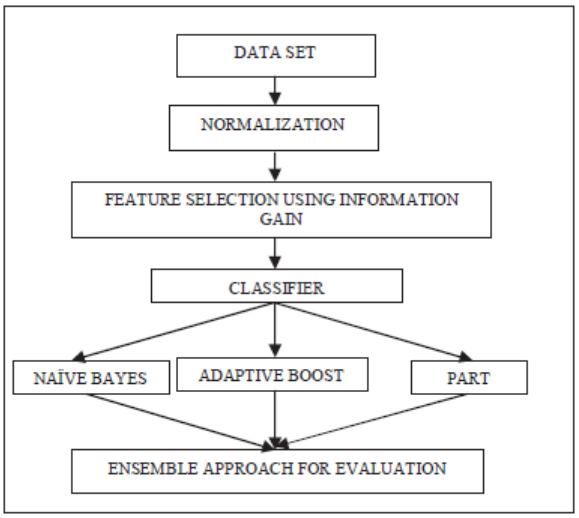

Figure 3.4: Proposed model ensemble method [34]

tempts to break down the data based on attack type by going through several models. This specific method used six testing methods: RFC, J48, Logistic Regression, Decision Tree, SVM, and Naive Bayes. While several models are used, the biggest constant is the use of the NSL-KDD Dataset. This is important because the NSL-KDD has its attacks separated into five separate attack categories, meaning each model can be adjusted to a specific attack for the best results. This method recorded five metrics: Accuracy, Precision, Recall, F1-Score, and ROC. ROC is a metric that is not often recorded but is the curve between true positive and true negative values; the area between them is the ROC. The most helpful metric here was ROC, which is a new metric that isn't often used, and it gives us a new statistic to understand model strength.

3.0.7 Intrusion Detection System Using Machine Learning Algorithms

The final paper studied is "Intrusion Detection System Using Machine Learning Algorithms," written by R.Tahri, Y. Balouki, A. Jarrar, and A. Lasbahani[36] released in 2022. The method used for this is three different methods spread between two different datasets. The three methods implemented are K nearest neighbor, Naive Bayes, and Support Vector Machine. All three methods un-

derwent a two-pronged approach before moving on to a second phase. After the conclusion of the first phase, the top two performers move on to the second dataset. After the completion of the second phase, the results are returned. The only metric given from this experiment is Accuracy. Accuracy, in this case, is essential because we see the strong performers run through two of the datasets, allowing us to see how they react to new data entered.

Figure 3.5: Proposed model of 3 model implementation [36]

Chapter 4

Discussion

This section will discuss the results of our research into machine learning implementation in network intrusion detection systems. Our first paper, released in 2013, proposed a method by which they implemented feature selection, SMOTE, and random forest classifiers. This paper proposed two different forms of this method, one with 19 features and one with 22. The two proposed methods show similar results with a higher feature count, which indicates slightly stronger results in some fields. The strength of the 22-feature model is shown when measuring the Detection Rate and Precision metrics. With Detection Rate, we see both models showing the same results until they arrive at R2L and U2R data, where we see a difference of .01 and .013, respectively.[31] With Precision, we see a similar result, but with a change of .008 and .01, respectively.[31] Its build time is The only strength that puts the lower feature model above the larger one. The build time of the lower-featured model was 391.39 seconds [31], beating out the 22-feature model by 3 seconds.

The following paper will discuss a method introduced using a stacked non-symmetric deep auto-encoder into a deep belief network. This implementation comes in 2017, four years after the previous paper. As mentioned previously, this method used three different deployments. The methods tested were against two distinct datasets, with the second dataset being tested twice with two different implementations on its testing. The data breakdown for this method shows us Accuracy, Precision, Recall, F-Score, and False Alarm. These categories are further broken into a deep belief network and an NSDAE. Our first set of results can be seen in Figure 4.2. While working on the KDD '99 dataset, SNDAE

		Random Forests	SMOTE + Random Forests	Proposed Models	
				SMOTE + Feature Selection + Random Forests	SMOTE + Feature Selection + Random Forests
Number of features		**41**	**41**	**19**	**22**
Detection Rate	Normal	1	0.999	0.999	0.999
	DoS	1	1	1	1
	Probing	0.996	0.996	0.996	0.996
	R2L	0.961	0.96	0.953	***0.963***
	U2R	0.596	0.959	0.949	***0.962***
FPR	Normal	0.002	0.002	0.002	0.002
	DoS	0	0	0	0
	Probing	0	0	0	0
	R2L	0	0	0	0
	U2R	0	0	0	0
Precision	Normal	0.998	0.998	0.998	0.999
	DoS	1	0.999	0.999	0.999
	Probing	0.999	0.999	0.999	0.999
	R2L	0.992	0.993	0.985	**0.993**
	U2R	0.912	0.968	0.963	**0.972**
Time to build the Model in seconds		500.81	445.27	**391.39**	394.39

Figure 4.1: Comparison table of the methods as found in the original material [31]

outclassed DBN in all categories except recall and false alarm. We see that false alarms are higher than DBN's by .05 percent, and recall is higher by .06 percent. Moving onto a more robust dataset in NSL-KDD, we see that in all fields, SNDAE outperforms DBN in every metric. This set was with five classes of attacks, so it wasn't as precise. The final data set in Figure 4.4 has 15 attack classes but only tests against SNDAE. With this more robust and accurate test, we see that the 13-class method outperforms the results of the 5-class. We see an increase of approximately 4.0 percent in Accuracy, approximately 4.0 in Recall, Approximately 3.0 in score, and a reduction of 4.0 in false alarms. The only thing that the 5 class wins in is Precision, where it beats it by 8.0 percent. How does this 2017 implementation compare to that of the 2013 one, though? Comparing the two papers is a little more complicated than you might think since they have differing methods and metrics. Based on what we know, we have to look at the best-performing model from both papers; in this case, that would be the 22-feature SMOTE with Feature Selection and RFC and the KDD 99 attempt using SNDAE. These two papers overlap in one metric being Precision. Looking at Precision for both of them, we can see that SNDAE was better at catching R2L

attacks but not in U2R. That being said, the fact that the 2017 implementation can collect so much data and recognize so many attacks, in my opinion, is the superior implementation at this point.

Attack Class	No. Training	No. Attacks	Accuracy (%)		Precision (%)		Recall (%)		F-Score (%)		False Alarm (%)	
			DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE
Normal	97278	60593	99.49	99.49	94.51	100.00	99.49	99.49	96.94	99.75	5.49	8.92
DoS	391458	223298	99.65	99.79	98.74	100.00	99.65	99.79	99.19	99.89	1.26	0.04
Probe	4107	2377	14.19	98.74	86.66	100.00	14.19	98.74	24.38	99.36	13.34	10.83
R2L	1126	5993	89.25	9.31	100.00	100.00	89.25	9.31	94.32	17.04	0.00	0.71
U2R	52	39	7.14	0.00	38.46	0.00	7.14	0.00	12.05	0.00	61.54	100.00
Total	494021	292300	97.90	97.85	97.81	99.99	97.91	97.85	97.47	98.15	2.10	2.15

Figure 4.2: Comparison table of the KDD '99 results of the experiment[32]

Attack Class	No. Training	No. Attacks	Accuracy (%)		Precision (%)		Recall (%)		F-Score (%)		False Alarm (%)	
			DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE
DoS	45927	5741	87.96	94.58	100.00	100.00	87.96	94.58	93.60	97.22	8.80	1.07
Normal	67343	9711	95.64	97.73	100.00	100.00	95.64	97.73	97.77	98.85	24.29	20.62
Probe	11656	1106	72.97	94.67	100.00	100.00	72.97	94.67	84.37	97.26	18.40	16.84
R2L	995	2199	0.00	3.82	0.00	100.00	0.00	3.82	0.00	7.36	0.00	3.45
U2R	52	37	0.00	2.70	0.00	100.00	0.00	2.70	0.00	5.26	0.00	50.00
Total	125973	18794	80.58	85.42	88.10	100.00	80.58	85.42	84.08	87.37	19.42	14.58

Figure 4.3: Comparison table of the results for the 5-class NSL-KDD experiment [32]

Label	No. Training	No. Attack	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	False Alarm (%)
'back'	956	359	36.77	100.00	36.77	53.77	0.00
'buffer_overflow'	30	20	0.00	0.00	0.00	0.00	100.0
'guess_password'	53	1231	0.00	0.00	0.00	0.00	0.00
'ipsweep'	3599	141	98.58	100.00	98.58	99.29	6.71
'neptune'	41214	4657	98.05	100.00	98.05	99.01	0.00
'nmap'	1493	73	100.00	100.00	100.00	100.00	0.00
'normal'	67343	9711	97.91	100.00	97.91	98.94	14.70
'pod'	201	41	92.68	100.00	92.68	96.20	28.30
'portsweep'	2931	157	95.54	100.00	95.54	97.72	44.03
'satan'	3633	735	82.45	100.00	82.45	90.38	13.92
'smurf'	2646	665	99.10	100.00	99.10	99.55	0.15
'teardrop'	892	12	100.00	100.00	100.00	100.00	75.51
'warezclient'	890	0	0.00	0.00	0.00	0.00	0.00
Total	125881	17802	89.22	92.97	89.22	90.76	10.78

Figure 4.4: Comparison table of the NSL-KDD 13 class experiment[32]

Similar to the previous model, the third method, released in 2017, used a deep neural network. This experiment ran data through a deep neural network at intervals. Each interval lowered the amount of standard data and heightened attack data. The percentage of attack packets was increased at a rate of ten percent per interval and climbed regularly. The main thing to note from the results in Figure 4.5 is that the false alarm rate rises as the attack rate increases. While we have other metrics in this method that are the most important, we can see that the DNN performed great in its other two metrics and that the most

critical information gained is how a machine learning model will react to more and more attack data. As such, comparing these to the other two models isn't as needed as the prior two models; it's better to take from this the information about how they respond to more significant attacks.

Attack packet rate (%)	Accuracy (%)	Detection Rate (%)	False Alarm (%)
10	99.01	99.25	0.01
20	99.06	99.53	0.01
30	98.95	99.19	0.03
40	99.04	99.37	0.06
50	99.08	99.66	0.04
60	99.07	99.71	0.05
70	99.08	99.67	0.06
80	99.08	99.81	0.05
90	99.01	99.81	0.47

Figure 4.5: Comparison table of DNN approach[33]

The following implementation was created in 2018 and attempted an ensemble approach. Figure 4.6 shows how each part of the ensemble works without stripping it of any features. From this, we get our baseline for the method, from which we can base the rest of the experiment. We can see that all three methods perform relatively well in all three categories, with NB showing 92.7 percent accuracy, 98.9 percent precision, and 92.7 percent recall. PART showed us 99.9 percent accuracy, 99.9 percent precision, and 99.9 percent recall. Adaptive Boost showed 97.8 percent accuracy, 96.2 percent precision, and 97.9 percent recall.[34] Figure 4.7 shows us the results of the three models, but this time, feature se-

Classifier	Accuracy	Precision	Recall
Naïve Bayes	*92.7840*	*98.90*	*92.70*
PART	*99.9601*	*99.96*	*99.90*
Adaptive Boost	*97.8597*	*96.20*	*97.90*

Figure 4.6: Comparison table of all three parts without feature selection [34]

lection was activated. For NB, Accuracy and recall fall slightly, while Precision remains the same. PART shows us a decrease in all three categories, although all reductions are minor. Adaptive Boost shows a surprising rise in Accuracy while showing the exact statistics for Precision and recall. Finally, we have the feature

Classifier	Accuracy	Precision	Recall
Naïve Bayes	*91.9818*	*98.90*	*92.00*
PART	*99.9589*	*99.90*	*99.60*
Adaptive Boost	*97.9073*	*96.20*	*97.90*

Figure 4.7: Comparison table of three styles using feature selection [34]

selection results using an ensemble approach, which is the approach presented in this paper, and its results are presented in Figure 4.8. The ensemble approach is found to have performed better in all fields. The ensemble approach scored a 99.97 percent accuracy, beating out the other three methods. In Precision, the ensemble method showed a result of 99.99 percent precision. This precision rating, once again, beat out the other three methods when they acted individually. Finally, in recall, the ensemble approach gives a recall score of 99.98 percent. Comparing this to the other three methods present, it is easy to see that the en-

Classifier	Accuracy	Precision	Recall
Naïve Bayes	*91.9818*	*98.90*	*92.00*
PART	*99.9589*	*99.90*	*99.60*
Adaptive Boost	*97.8597*	*96.20*	*97.90*
Ensemble Approach	*99.9732*	*99.99*	*99.98*

Figure 4.8: Comparison table with the ensemble approach[34]

semble method helped provide a strong precision and accuracy score, which was better than that of previous iterations; as such, I would determine that overall, it is the current best method that has been discussed. It is essential to understand that the other methods still have their strengths and are not enviable, but this method from 2018 is the most viable.

The second-to-last paper is made in 2021 and attempts to challenge several models against each other, similar to the ensemble approach, but it doesn't try to use them to work together. The data for this implementation is split based on the attack, giving us a more in-depth look at how each attack fairs against each method. The first set of results seen in Figure 4.9 is for Denial of Service (DoS) attacks. As shown in Figure 4.9, RFC beats the other methods in all categories except for the F1-score, where the Decision Tree barely beats it out. Figure 4.10

Machine Learning algorithms	Accuracy (%)	Precision	Recall	F1-Score	ROC
Random Forest	**99.83**	**1**	**0.999**	0.999	**0.999**
J48	99.76	0.99	**0.999**	0.999	0.658
Logistic Regression	99.47	0.99	0.998	0.999	0.727
Decision Tree	99.66	0.99	0.999	**1.0**	0.618
SVM	99.19	0.98	0.999	0.999	0.5
Naïve Bayes	94.19	0.97	0.903	0.948	0.844

Figure 4.9: Comparison table for DoS attacks[35]

shows the results of a probe attack. Here, we see that RFC and J48 are tied in Accuracy; RFC performs better than the other methods in Precision. In Recall, F1-score and ROC RFC tie with SVM. Figure 4.11 shows us how each model fairs

Machine Learning algorithms	Accuracy (%)	Precision	Recall	F1-Score	ROC
Random Forest	**99.94**	**0.999**	**1.0**	**1.0**	**1.0**
J48	**99.94**	0.994	0.999	0.999	0.999
Logistic Regression	99.47	0.995	0.998	0.998	0.996
Decision Tree	99.91	0.993	0.999	0.998	**1.0**
SVM	99.90	0.994	**1.0**	**1.0**	**1.0**
Naïve Bayes	90.31	0.986	0.966	0.956	0.974

Figure 4.10: Comparison table for probe attacks[35]

against Remote to Local attacks. As can be seen, once again, RFC beats out the others in most categories. The only category in which RFC is the lone winner is Precision, which ties NB. Figure 4.12 shows us the results of the models placed

Machine Learning algorithms	Accuracy (%)	Precision	Recall	F1-Score	ROC
Random Forest	**99.99**	**0.999**	**1.0**	**0.999**	**1.0**
J48	99.89	0.998	0.999	0.998	0.994
Logistic Regression	99.81	0.997	0.992	0.992	0.997
Decision Tree	99.74	0.997	0.993	0.993	0.999
SVM	98.92	0.995	0.990	0.990	0.979
Naïve Bayes	98.92	**0.999**	0.882	0.931	0.953

Figure 4.11: Comparison table for R2L attacks[35]

against User to Root attacks. For this attack, we see some mixed results. RFC wins in Accuracy again but loses to J48 and Logistic Regression in Precision. For recall, we see RFC tie J48. Finally, with ROC, we see that the decision tree performs best. For how this paper compares to the rest, I believe the most

Machine Learning algorithms	Accuracy (%)	Precision	Recall	F1-Score	ROC
Random Forest	**99.85**	0.999	**0.998**	**0.998**	0.997
J48	99.67	**1**	**0.998**	**0.998**	0.936
Logistic Regression	99.16	**1**	0.995	0.995	0.982
Decision Tree	99.29	0.999	0.993	0.993	**0.999**
SVM	98.97	0.999	0.992	0.992	0.826
Naïve Bayes	88.85	0.999	0.942	0.960	0.948

Figure 4.12: Comparison table for U2R attacks[35]

valuable information gained from this paper is that the models have different

	Accuracy
KNN (k=3)	93.3333%
NB	95.55555%
SVM	97.77777%

Figure 4.13: Comparison table after being run through the NSL-KDD Dataset[36]

	Accuracy Rate on NSL-KDD
SVM	97,29
Naïve Bayes	67,26

Figure 4.14: Comparison table after being run through the UNSW-NB15 dataset[36]

strengths based on the attacks, so you may be able to tailor your implementation around the attacks you are facing. With this information, you can apply it to different implementations to see if they run better and give more desired results. The final paper to discuss the results is from 2022; this paper attempted to implement using three models on two datasets. Figure 4.13 shows the results of each model after being run through the NSL-KDD dataset. As shown in Figure 4.13, NB and SVM outperform K's nearest neighbor; as the method requests, those two move on to the following dataset. Figure 4.14 shows us the results after the two successful models get moved onto the UNSW-NB15 Dataset. As seen in Figure 4.14, SVM far outperforms NB in this implementation. Similar to the other comparisons, this is another implementation that can be done with assistance from the different implementations. It's far from the best we have reviewed thus far, but it is an important test to run while trying to find the best-case implementation.

Chapter 5

Future Works

When discussing possible future research on machine learning implementations in network intrusion detection systems, it is essential to review our current results. When examining our results, we will find a new implementation that would theoretically fix the issue of false alarms being set off in anomaly-based detection. We start with the UNSW-NB15 dataset, originating in 2015. I believe using a more recent dataset than those primarily implemented in the papers discussed will bring a more robust result to the implementation. We will run it from this dataset through SMOTE and Feature selection, as seen in the first paper on machine learning we discussed. From here, we will branch out, allowing the implementation to be tailored to the situation at hand. We will branch off into two sets of three models. The data will be run through each model, and then the two sets of three will each combine into two separate ensemble models. We will choose our models based on the information gained in paper 5 to ensure we are grabbing our best-performing models. We will then pit the two ensemble models against each other based on how they react to another dataset, in this case, CIC-IDS2017. As new datasets are released, switching out the datasets used in the implementation is imperative so that the model can grow with new information. From this, we choose the most precise outcome to ensure we are lowering our false alarm rate.

5.0.1 Implementation into Wireless networks

While much of this paper is created to discuss the machine learning implementation to network intrusion detection systems, explaining how this suggested imple-

mentation will help Wireless Networks remain safer is essential. As mentioned, the main goal is to fix the most significant limitation of current NIDS in wireless networks: false alarms. The implementation that would be added to the network for security would be the one that performed the best in the proposed training model explained above. This implementation would have already been trained and ready to deploy before the network. Figure 5.2 shows an example of how a NIDS would work to secure a network. As seen in Figure 5.2, Input data would be caught inbound to a network and have to pass through the NIDS. The NIDS would read the data and determine whether it is a threat. If the data is threatening, the NIDS will dispose of it. This implementation will help secure the network more effectively by reducing the amount of false alarms while also being able to adapt to attacks that are not yet known. This model can also be tailored for specific attacks; if the user would like to strengthen against a particular kind of attack, they have put stronger models against those types of attacks into the training so that they are added to the ensemble method.

5.0.2 Implementation into other domains

While this research pertains primarily to wireless networks, much of the information gained can be adapted to other mediums, specifically Cloud computing, 5G services, and IoT. During the research into this topic, it was evident that other mediums have also been investigated for these issues; as such, the research may benefit from adding some methods from these different domains. Some methods have been tested to improve the security of these systems. Some of these methods include: using ciphers to encrypt data[37][38][39],Implementing other models that pretain to specific domains, or creating whole new IDS' for the specific domains.

With the emergence of 5G, we have seen many advancements in the networking world.[40] 5G allowed more accessibility than ever before,[41] with faster speeds[42] and lower latency's[43] than ever before. The innovation of 5G services fueled innovation for technologies that were bandwidth-heavy[44]. These tech-

nologies could range from a simple cell phone to even a military-issued UAV[45]; the possibilities were practically endless. With these innovations came the disadvantage of a larger attack surface. 5G networks are very vulnerable to attacks due to the large surface created by such innovations.[46] However this does not mean that 5G networks are defenseless, many methods have already been derived to try and mitigate attacks, many of which could be implemented to the model suggested in this report. For instance, the benefits of using a DET model include improving decision-making and classification of attacks.[47] Some other methods used in 5G security research could include Federated Learning[48], as well as attack graphs to help predict vulnerabilities.[49][50][51] When researching 5G, you will also notice the mention of "network slicing" in several papers.[41][52][53]. Network slicing is a configuration allowing multiple virtual networks to be created on a common infrastructure.[54] Due to network slicing's issue with trust, it is easy to see that further security measures are needed to protect it.[55] Network slicing could be referenced to adapt the implementation as mentioned earlier, as implementing our implementation out of wireless networks and into 5G where network slicing is present may also prove beneficial. It may also benefit looking into future-proofing the proposed model for possible 6G security[55] to avoid safety gaps between generational growth.

Cloud-based systems are another domain that could benefit from machine learning implementation to NIDS. A cloud-based system can do many things, but one of its primary uses is distributing a service to multiple users. [56] With this being such a user-heavy system, security is imperative to safety on the networks[57]. However, typical IDS often don't work as strongly in a Cloud System[58], as such Cloud-based Intrusion Detection systems[59] were created. These have shown in many different forms, including autonomous options like CIDS(HAF-CIDS)[60][58] and Autonomic Cloud Intrusion Detection Framework (ACIDF)[61][62]. These frameworks typically work off of models like Markov Model[63] or a Probabilistic Suffix Tree (PST)[64]. While research has already

been done into autonomous intrusion detection[65] onto these cloud-based systems and other risk mitigation techniques[66], they can be further expanded with newer implementations of machine learning. Cloud systems are very susceptible to things like masquerade or impersonation attacks[67][68][69],as well more common attacks like DOS[70], which have multiple security measures tested to try and protect them.[71] Implementing the approach offered in this report could benefit, especially if trained around the premise of a masquerade/impersonation attack or DoS attack to strengthen these cloud-based systems against these attacks. Swarm-based prediction is another security measure that may be worth implementing in future iterations of this model.[72] While being used primarily on cloud and other big data domains[73], it may be worth researching this for the wireless network implementation model in depth. Another strong introduction to our presented model would be the implementation of the ICSD dataset, which is a dataset explicitly created for cloud-based IDS[74] but being a newer dataset can be used effectively to generate a stronger model.

When looking at the domain of IoT, it is easy to see the abundance of possible security vulnerabilities. While methods like QoS[75] have already been implemented in most IoT systems, further security implementations can only strengthen their security. IoT systems can range from things like the microwave in your house to the wireless sensors[76][77][78] used to enhance the security of your home. With such a large range of implementations, securing these systems to ensure their safe usage is more important than ever. With the introduction of things like Social IoT[79], these systems are now more susceptible to attacks than ever before. Implementing an SIoTsim into the proposed model could lead to stronger results in the security of IoT systems[80] and strengthen the effectiveness of a NIDS on the IoT network.

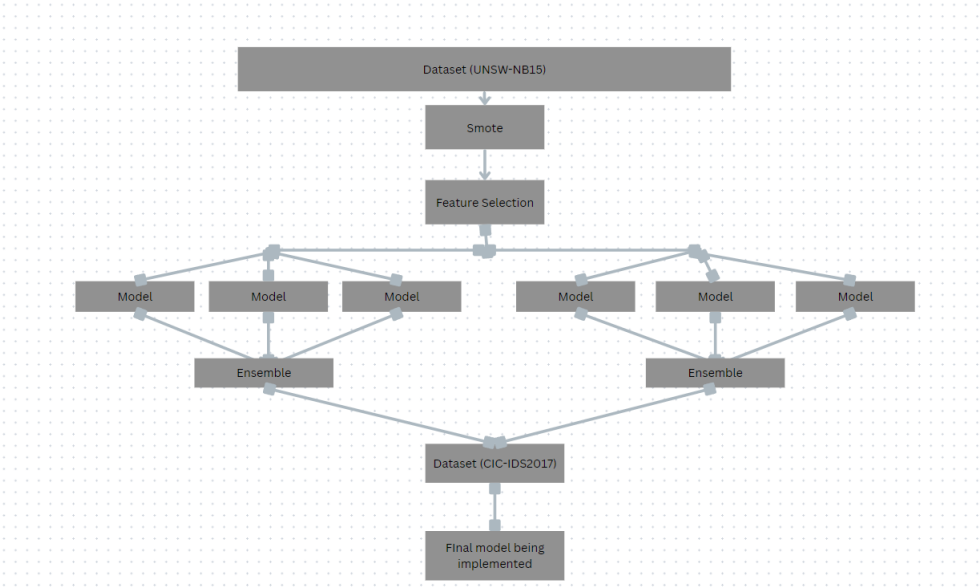

Figure 5.1: Proposed Training model for a stronger NIDS

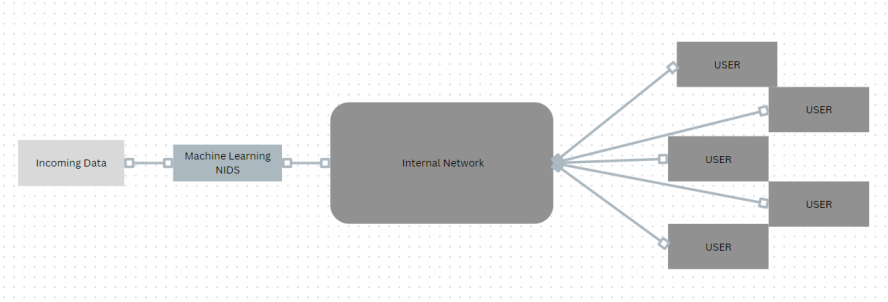

Figure 5.2: Example of a very basic network using NIDS

Chapter 6

Concluding Remarks

6.1 Summary

This conclusion will wrap up everything researched and discussed in this paper. First, we started with a short introduction to the material, giving a basic overview of the research idea and introducing topics like wireless networks, NIDS, and Machine learning. Second, we moved on to methodology, where we discussed the methods used within the papers being discussed to understand the material going into the literature review section. This included going over topics like the different types of machine learning models being discussed and the datasets that were being used. In the third chapter; we go into our literature review; here, we look at each paper and its proposed method or what they were looking to solve. In the fourth section, we went more in-depth with the results that were received from the papers we discussed in the literature review. In the fifth chapter, we give a possible avenue for future research; here, I explain how the research done can be implemented into each other to make a new and more robust model, while also adding in a new dataset that is more up-to-date in order to strengthen the strength of the model coming out of training in order to secure a wireless network. In this section we also went into detail on how future research can be expanded upon into other domains like 5G,Cloud,and IoT.

REFERENCES

[1] Q. Chen, H. A. Kholidy, S. Abdelwahed, and J. Hamilton, "Towards realizing a distributed event and intrusion detection system," in *International Conference on Future Network Systems and Security (FNSS 2017)*, Gainesville, Florida, USA, August 2017.

[2] Techopedia. (No date) What is an intrusion detection system (ids)? [Online]. Available: https://www.techopedia.com/definition/3988/intrusion-detection-system-ids

[3] I. Almazyad, S. Shao, S. Hariri, and H. A. Kholidy, "Anomaly behavior analysis of smart water treatment facility service: Design, analysis, and evaluation," in *2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, Giza, Egypt, 2023, pp. 1–7.

[4] M. M. Badr, M. I. Ibrahem, H. A. Kholidy, M. M. Fouda, and M. Ismail, "Review of the data-driven methods for electricity fraud detection in smart metering systems," *Energies*, vol. 16, no. 6, p. 2852, 2023, iF: 3.25.

[5] I. Elgarhy, M. M. Badr, M. Mahmoud, M. M. Fouda, M. Alsabaan, and H. A. Kholidy, "Clustering and ensemble based approach for securing electricity theft detectors against evasion attacks," *IEEE Access*, January 2023, iF: 3.55.

[6] I. Elgarhy, A. El-toukhy, M. Badr, M. Mahmoud, M. Fouda, M. Alsabaan, and H. A. Kholidy, "Secured cluster-based electricity theft detectors against blackbox evasion attacks," in *IEEE 21st Consumer Communications & Networking Conference (CCNC)*, Virtual Conference, January 6-9 2024.

[7] A. A. Khalil, M. A. Rahman, and H. A. Kholidy, "Fakey: Fake hashed key attack on payment channel networks," in *2023 IEEE Conference on Communications and Network Security (CNS)*, Orlando, FL, USA, 2023, pp. 1–9.

[8] Wikipedia contributors, "Wireless network," https://en.wikipedia.org/wiki/Wireless_network.

[9] Özer ÇELİK, "A research on machine learning methods and its applications," 2018. [Online]. Available: https://www.researchgate.net/publication/327547625_A_Research_on_Machine_Learning_Methods_and_Its_Applications

[10] Akamai, "Akamai Glossary: What is a Low-and-Slow Attack?" https://www.akamai.com/glossary/what-is-a-low-and-slow-attack#:~:text=A%20low%20and%20slow%20attack%20is%20a%20type%20of%20denial,very%20slow%20rate%20of%20volume, No date.

[11] Heimdal Security, "Heimdal Security Blog: What is Ping of Death?" https://heimdalsecurity.com/blog/what-is-ping-of-death/, No date.

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.

[13] University of New Brunswick Cybersecurity, "UNB-CIC Datasets - NSL-KDD," https://www.unb.ca/cic/datasets/nsl.html, No date.

[14] Z. Zoghi and G. Serpen, "UNSW-NB15 Computer Security Dataset: Analysis through Visualization," *Electrical Engineering & Computer Science, University of Toledo*, 2017.

[15] University of New Brunswick (UNB), "IDS 2017 dataset," https://www.unb.ca/cic/datasets/ids-2017.html.

[16] L. Breiman, "Random forests," in *Ensemble Machine Learning: Methods and Applications*, C. Zhang and Y. Ma, Eds. Springer, 2012, pp. 157–176.

[17] M. Alkhowaiter, H. A. Kholidy, M. A. Alyami, A. Alghamdi, and C. Zou, "Adversarial-aware deep learning system based on a secondary classical machine learning verification approach," *Sensors*, vol. 23, p. 6287, 2023, iF: 3.9. [Online]. Available: https://doi.org/10.3390/s23146287

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," 2002.

[19] H. A. Kholidy, A. Tekeoglu, S. Lannucci, S. Sengupta, Q. Chen, S. Abdelwahed, and J. Hamilton, "Attacks detection in scada systems using an improved non-nested generalized exemplars algorithm," in *12th IEEE International Conference on Computer Engineering and Systems (ICCES 2017)*, February 2018.

[20] Wikipedia contributors, "Deep learning," https://en.wikipedia.org/wiki/Deep_learning.

[21] Built In, "Relu activation function," https://builtin.com/machine-learning/relu-activation-function.

[22] Medium, "K-nearest neighbor," https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4.

[23] A.-u. Rahman, M. Mahmud, T. Iqbal, H. Kholidy, L. Saraireh, and et al., "Network anomaly detection in 5g networks," *Mathematical Modelling of Engineering Problems*, vol. 9, pp. 397–404, 2022.

[24] GeeksforGeeks, "Naive bayes classifiers," https://www.geeksforgeeks.org/naive-bayes-classifiers/.

[25] Towards Data Science, "Support vector machine: Introduction to machine learning algorithms," https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[26] Medium, "J48 classification (c4.5 algorithm) in a nutshell," https://medium.com/@nilimakhanna1/j48-classification-c4-5-algorithm-in-a-nutshell-24c50d20658e.

[27] ——, "Understanding logistic regression: A step-by-step explanation," https://medium.com/@satyarepala/understanding-logistic-regression-a-step-by-step-explanation-9a404344964b.

[28] Wikipedia contributors, "Decision tree," https://en.wikipedia.org/wiki/Decision_tree.

[29] Medium, "Understanding the adaboost algorithm," https://medium.com/@datasciencewizards/understanding-the-adaboost-algorithm-2e9344d83d9b.

[30] Unknown Authors, "Intrusion detection and prevention systems in wireless networks," *Kurdistan Journal of Applied Research*, vol. 2, no. 3, p. 6, August 2017.

[31] A. Tesfahun and D. L. Bhaskari, "Intrusion detection using random forests classifier with smote and feature reduction," in *2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies*, 2013, pp. 195–201.

[32] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 4, pp. 269–279, Nov. 2017.

[33] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Big Data and Smart Computing*.

[34] R. K. S. Gautam and A. Doegar, "An ensemble approach for intrusion detection system using machine learning algorithms," in *Proceedings of the 2017 International Conference on Intelligent Sustainable Systems (ICISS)*, 2017, pp. 696–700.

[35] V. Pai, Devidas, and N. Adesh, "Comparative analysis of machine learning algorithms for intrusion detection," in *IOP Conf. Ser.: Mater. Sci. Eng.*, ser. Article ID 012038, vol. 1013, 2021.

[36] R. Tahri, Y. Balouki, A. Jarrar, and A. Lasbahani, "Intrusion detection system using machine learning algorithms," in *ITM Web Conf.*, ser. Article ID 02003, vol. 46, 2022, pp. 1–6.

[37] A. Boualem, A. Berrouachedi, M. Ayaida, H. Kholidy, and E. Benkhelifa, "A new hybrid cipher based on prime numbers generation complexity: Application in securing 5G networks," in *2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, Giza, Egypt, 2023, pp. 1–8.

[38] H. A. Kholidy, "Accelerating stream cipher operations using single and grid systems," US Patent US 20 120 089 829 A1, April, 2012.

[39] H. A. Kholidy and K. S. Alghathbar, "Adapting and accelerating the stream cipher algorithm RC4 using ultra gridsec and himan and use it to secure himan data," *Journal of Information Assurance and Security (JIAS)*, vol. 4, no. 4, p. 274, 2009. [Online]. Available: http://www.mirlabs.org/jias/vol4-issue6.html

[40] A. A. Abushgra, H. A. Kholidy, A. Berrouachedi, and R. Jaziri, "Innovative routing solutions: Centralized hypercube routing among multiple clusters in 5g networks," in *2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, Giza, Egypt, 2023, pp. 1–7.

[41] H. A. Kholidy, T. Anjony, and I. Almazyad, "A survey of the AI and cybersecurity solutions used for network slicing in the 5G and beyond domain," November 2023.

[42] A. Fox, H. A. Kholidy, and I. Almazyad, "Current 5g federation trends: A literature review," *November 2023*.

[43] H. A. Kholidy and M. Abuzamak, "5g network management, orchestration, and architecture: A practical study of the monb5g project," *arXiv*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2212.13747

[44] H. A. Kholidy, M. A. Rahman, A. Karam, and Z. Akhtar, "Secure spectrum and resource sharing for 5g networks using a blockchain-based decentralized trusted computing platform," *arXiv*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2201.00484

[45] M. Abuzamak and H. Kholidy, "Uav based 5g network: A practical survey study," *arXiv*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2212.13329

[46] H. A. Kholidy and et al., "Secure the 5G and beyond networks with zero trust and access control systems for cloud native architectures," in *2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, Giza, Egypt, 2023, pp. 1–8.

[47] H. A. Kholidy, A. Berrouachedi, E. Benkhelifa, and R. Jaziri, "Enhancing security in 5G networks: A hybrid machine learning approach for attack classification," in *2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, Giza, Egypt, 2023, pp. 1–8.

[48] H. A. Kholidy and R. Kamaludeen, "An innovative hashgraph-based federated learning approach for multi domain 5g network protection," in *2022 IEEE Future Networks World Forum (FNWF)*, Montreal, Canada, October 2022. [Online]. Available: https://fnwf.ieee.org/wp-content/uploads/sites/339/2022/10/AcceptedPaperScheduleV0.1.pdf

[49] H. A. Kholidy, "A triangular fuzzy based multicriteria decision making approach for assessing security risks in 5g networks," *arXiv*, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2112.13072

[50] H. Kholidy, "Multi-layer attack graph analysis in the 5g edge network using a dynamic hexagonal fuzzy method," *Sensors*, vol. 22, 2022, iF: 3.576. [Online]. Available: https://doi.org/10.3390/s22010009

[51] S. Iannucci, H. A. Kholidy, A. D. Ghimire, R. Jia, S. Abdelwahed, and I. Banicescu, "A comparison of graph-based synthetic data generators for benchmarking next-generation intrusion detection systems," in *IEEE Cluster*, Hawaii, USA, September 5 2017.

[52] H. A. Kholidy, A. Karam, J. H. Reed, and Y. Elazzazi, "An experimental 5G testbed for secure network slicing evaluation," in *2022 IEEE Future Networks World Forum (FNWF)*, Montreal, Canada, October 2022. [Online]. Available: https://fnwf.ieee.org/wp-content/uploads/sites/339/2022/10/AcceptedPaperScheduleV0.1.pdf

[53] M. Malkoc and H. A. Kholidy, "5G network slicing: Analysis of multiple machine learning classifiers," *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2310.01747

[54] H. A. Kholidy, "A smart network slicing provisioning framework for 5G-based IoT networks," in *2023 10th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, San Antonio, TX, USA, 2023, pp. 104–110.

[55] H. A. Kholidy, A. Karam, J. Sidoran, and et al., "Toward zero trust security in 5g open architecture network slices," in *IEEE Military*

Conference (MILCOM), CA, USA, November 29 2022. [Online]. Available: https://edas.info/web/milcom2022/program.html

[56] H. A. Kholidy, F. Baiardi, S. Hariri, E. M. ElHariri, A. M. Youssouf, and S. A. Shehata, "A hierarchical cloud intrusion detection system: Design and evaluation," *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, November 2012.

[57] H. A. Kholidy, F. Baiardi, and A. Azab, "A data-driven semi-global alignment technique for masquerade detection in stand-alone and cloud computing systems," Submitted, 2019, granted on January 2019, US 20170019419 A1.

[58] H. A. Kholidy, A. Erradi, S. Abdelwahed, and F. Baiardi, "Ha-cids: A hierarchical and autonomous ids for cloud environments," in *Fifth International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN)*, Madrid, Spain, June 2013.

[59] H. A. Kholidy and F. Baiardi, "Cids: A framework for intrusion detection in cloud systems," in *The 9th International Conference on Information Technology: New Generations (ITNG)*, Las Vegas, Nevada, USA, 2012.

[60] H. A. Kholidy, A. Erradi, S. Abdelwahed, and F. Baiardi, "A hierarchical, autonomous, and forecasting cloud ids," in *5th International Conference on Modeling, Identification and Control (ICMIC2013)*, Cairo, August 31-September 2 2013.

[61] H. A. Kholidy, A. Erradi, and S. Abdelwahed, "Online risk assessment and prediction models for autonomic cloud intrusion prevention systems," in *11th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, Doha, Qatar, November 2014.

[62] ——, "Attack prediction models for cloud intrusion detection systems," in *International Conference on Artificial Intelligence, Modelling and Simulation (AIMS2014)*, Madrid, Spain, November 2014.

[63] H. A. Kholidy, A. Erradi, S. Abdelwahed, and A. Azab, "A finite state hidden markov model for predicting multistage attacks in cloud systems," in *12th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, Dalian, China, August 2014.

[64] H. A. Kholidy, A. M. Yousouf, A. Erradi, H. A. Ali, and S. Abdelwahed, "A finite context intrusion prediction model for cloud systems with a probabilistic suffix tree," in *8th European Modelling Symposium on Mathematical Modelling and Computer Simulation*, Pisa, Italy, October 2014.

[65] H. A. Kholidy, A. Erradi, S. Abdelwahed, and F. Baiardi, "A risk mitigation approach for autonomous cloud intrusion response system," *Journal of Computing*, June 2016, iF: 2.42.

[66] H. A. Kholidy and A. Erradi, "A cost-aware model for risk mitigation in cloud computing systems," in *12th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, Marrakech, Morocco, November 2015.

[67] H. A. Kholidy, F. Baiardi, and S. Hariri, "DDSGA: A data-driven semi-global alignment approach for detecting masquerade attacks," *IEEE Transactions on Dependable and Secure Computing*, May 2014, iSI Impact factor: 6.791.

[68] H. A. Kholidy and F. Baiardi, "CIDD: A cloud intrusion detection dataset for cloud computing and masquerade attacks," in *Proceedings of the 9th International Conference on Information Technology: New Generations (ITNG)*, Las Vegas, Nevada, USA, 2012.

[69] H. Kholidy, "Detecting impersonation attacks in cloud computing environments using a centric user profiling approach," *Future Generation Computer Systems*, vol. 117, pp. 299–320, April 2021, iF: 7.307. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X20330715

[70] H. A. Kholidy, "Cloud-scada penetrate: Practical implementation for hacking cloud computing and critical scada systems," Department of Computer and Network Security, College of Engineering, SUNY Polytechnic Institute, 2020. [Online]. Available: http://hdl.handle.net/20.500.12648/1605

[71] ——, "Correlation based sequence alignment models for detecting masquerades in cloud computing," *IET Information Security Journal*, Sept. 2019, iF: 1.51. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-ifs.2019.0409

[72] A. Jakaria, M. Rahman, M. Asif, A. Khalil, H. Kholidy, M. Anderson, and S. Drager, "Trajectory synthesis for a uav swarm based on resilient data collection objectives," *IEEE Transactions on Network and Service Management*, 2022, iF: 4.75. [Online]. Available: https://ieeexplore.ieee.org/document/9928375?source=authoralert

[73] H. A. Kholidy, "An intelligent swarm-based prediction approach for predicting cloud computing user resource needs," *Computer Communications*, Feb 2020, iF: 5.047. [Online]. Available: https://authors.elsevier.com/tracking/article/details.do?aid=6085&jid=COMCOM&surname=Kholidy

[74] S. S. Haytamy, H. A. Kholidy, and F. A. Omara, "Integrated cloud services dataset," in *14th World Congress on Services, Held as Part of the Services Conference Federation, SCF 2018*, ser. Lecture Notes in Computer Science. Seattle, WA, USA: Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-94472-2

[75] H. A. Kholidy, H. Hassan, A. Sarhan, A. Erradi, and S. Abdelwahed, "Qos optimization for cloud service composition based on economic model," in *User-Centric IoT*, 2015, vol. 150.

[76] N. I. Haque, M. A. Rahman, D. Chen, and H. Kholidy, "BIoTA Control-Aware Attack Analytics for Building Internet of Things," *arXiv*, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2107.14136

[77] A. Boulem, C. De Runz, H. Kholidy, A. Bengheni, D. Taib, and M. Ayaida, "A new classification of target coverage models in wsns, survey and algorithms and future directions," in *The 7th International Conference on Information and Computer Technologies (ICICT 2024)*, Honolulu, Hawaii, March 15-17 2024.

[78] A. Boualem, C. De Runz, M. Ayaida, and H. A. Kholidy, "Probabilistic intrusion detection based on an optimal strong k-barrier strategy in wsns," *Peer-to-Peer Networking and Applications*, 2024. [Online]. Available: https://doi.org/10.1007/s12083-024-01634-w

[79] M. C. Zouzou, E. Benkhelifa, H. A. Kholidy, and D. W. Dyke, "Multi-context-aware trust management framework in social internet of things (MCTM-SIoT)," in *2023 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS)*, Valencia, Spain, June 19-22 2023, pp. 99–104.

[80] M. C. Zouzou, M. Shahawy, E. Benkhelifa, and H. Kholidy, "SIoTSim: Simulator for social internet of things," in *The 10th International Conference on Internet of Things: Systems, Management and Security (IOTSMS 2023)*, San Antonio, Texas, USA, October 2023.