5-2023

Comparing a Hybrid Multi-layered Machine Learning Intrusion Detection System to Single-layered and Deep Learning Models

Samantha Collen

Comparing a Hybrid Multi-layered Machine Learning Intrusion Detection System to Single-layered and Deep Learning Models

by

Samantha Collen

A Thesis

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science in

Information Assurance

May 2023

Thesis Paper Committee:
Jim Q. Chen, Chairperson
Shakour A. Abuzneid
Akalanka B. Mailewa
Abdullah Abu Hussein

Abstract

Advancements in computing technology have created additional network attack surface, allowed the development of new attack types, and increased the impact caused by an attack. Researchers agree, current intrusion detection systems (IDSs) are not able to adapt to detect these new attack forms, so alternative IDS methods have been proposed. Among these methods are machine learning-based intrusion detection systems. This research explores the current relevant studies related to intrusion detection systems and machine learning models and proposes a new hybrid machine learning IDS model consisting of the Principal Component Analysis (PCA) and Support Vector Machine (SVM) learning algorithms. The NSL-KDD Dataset, benchmark dataset for IDSs, is used for comparing the models' performance. The performance accuracy and false-positive rate of the hybrid model are compared to the results of the model's individual algorithmic components to determine which components most impact attack prediction performance. The performance metrics of the hybrid model are also compared to two deep learning Autoencoder Neuro Network models and the results found that the complexity of the model does not add to the performance accuracy. The research showed that pre-processing and feature selection impact the predictive accuracy across models. Future research recommendations were to implement the proposed hybrid IDS model into a live network for testing and analysis, and to focus research into the pre-processing algorithms that improve performance accuracy, and lower false-positive rate. This research indicated that pre-processing and feature selection/feature extraction can increase model performance accuracy and decrease false-positive rate helping businesses to improve network security.

Acknowledgements

Thank you to my Thesis Chair, Dr. Jim Chen for sharing your knowledge and guiding me through this process. Your encouragement was always appreciated. Thank you to Dr. Abu Hussein for helping me at the initial stage of finding my research interests and selecting a topic. Thank you to Dr. Akalanka Mailewa for providing me resources and helping me to narrow down my research topic to intrusion detection and machine learning. Thank you to Dr. Abuzneid for your encouragement and guidance with machine learning and python code. I am blessed to have had such an encouraging and knowledgeable committee to lead me through this journey. Thank you to everyone I did not mention that assisted me in this exploration. I could not have completed this process without your support, and I am truly grateful. A special thank you to my Academic Advisor, Dr. Lynn Collen for your guidance and direction, and to all my family for always being there for me.

Table of Contents

Chapter Page

List of Tables

Table Page

List of Figures

Figure Page

Chapter I: Introduction

Introduction

Advancements in technology have increased processing speeds, transmission rates, and computing memory capabilities at reduced costs, allowing innovations in new technologies such as 5G networks, Quantum Computing, biometric software, self-driving cars, SMART Homes, SMART Cities, and much more. Continued reliance on information systems and the exponential growth in the volume and variety of network devices and data have created more attack surface area and increased incentives for cybercriminals. Data is the most valued commodity and the financial incentives for stealing and selling personal and corporate data continues to grow. To paraphrase IBM's former Chairman, President, and Chief Executive Chairman, Ginni Rometty, data is the world's new natural resource and the new competitive advantage, and cybercrime is the greatest threat to every company in the world (Morgan, 2020). Advancements in technology have improved systems and connected the globe, but technology has also provided cybercriminals new tools and techniques to avoid detection from existing attack defense systems. The increase in the volume and variety of data, increase in attack surface area, and advancements in tools and techniques for cybercriminals have created new challenges to existing cybersecurity defense systems, but many researchers agree, Intrusion Detection Systems are the solution to boost network security (Maimo et al., 2018; Kaur & Singh, 2020; Sarker, 2022; Almasoudy et al., 2020; Al Jallad et al., 2020; Al-Yaseen et al., 2022; Saranya et al., 2020; Liu et al., 2021; Sun et al., 2020; Yange et al., 2020; Pingale & Sutar, 2022; Mijalkovic & Spognardi, 2022; Mushtaq et al., 2023).

Technology and the internet have connected the physical world to the digital global network. Technology and the internet have simplified our lives in many ways putting knowledge and convenience a mouse click away, and someday soon maybe a thought away (Neuralink, 2022). Digital networks allow us to communicate with friends through social media networks, to conduct business remotely with overseas companies, and to stream videos for educational and entertainment purposes. Every facet of our lives is connected to the global network (Sornsuwit & Jaiyen, 2019; Liu et al., 2021; Saranya et al., 2020; Mushtaq et al., 2023). According to Stanford University, nearly 50% of the current U.S. labor force works remotely (Morgan, 2020). With roughly 1 million more people joining the internet every day, a report from Cisco (2020) stated that by 2023 the number of network devices will be 3.6 times the global population. The number of internet users is expected to continue to grow to more than 7.5 billion users by 2030, up from 5 billion users in 2020 (Morgan, 2022). Additional devices produce more data transmission, more data, and thus more attack surface area. Data transmitted and stored on the internet has increased significantly the past few years. By the 2025, stored data is expected to reach more than 200 zettabytes, with half of it being stored in the cloud (Morgan, 2020; Morgan, 2022). The complexity of internet data, often referred to as big data, is defined through the 7v's 1) volume – size of data, 2) velocity – speed data is generated, 3) variety – different types of data, 4) value – worth of data, 5) veracity – trustworthiness of data, 6) variability – constant change of data meaning, and 7) visualization – ease of accessible or readable data. The exponential growth in data makes it difficult, if not impossible, for existing Intrusion

Detection Systems (IDS) to identify new threats (Saranya et al., 2020, Sun et al., 2020; Kaur & Singh, 2020; Liu et al., 2021; Kanna & Santhi, 2022; Mijalkovic & Spognardi 2022; Pingale & Sutar, 2022).

Advancements in technology have expanded the networks we communicate, work, and play in, but technology has also improved the tools and techniques cyber-attackers use to infiltrate network systems. The increase and sophistication of cyber-attacks has added to the challenges of detecting and predicting attacks on data and networks. Cyber-attack techniques include polymorphic viruses that can change file type to avoid existing virus detection software, and the use of Artificial Intelligence (AI) machine learning to allow cybercriminals easier and quicker monitoring of a systems vulnerabilities, better orchestrated social engineered phishing, ransom attacks, or Denial of Service Attacks (DoS), and faster brute force and dictionary password cracking, to name a few (Sarker, 2022). According to Mastercard 66% of small to medium sized businesses had at least one cyber incident in the past two years. After experiencing a breach, 60% of businesses will go out of business within 6 months (Morgan, 2020; Morgan, 2022). It is expected that the costs of global cybercrimes will "…grow by 15 percent per year over the next five years, reaching $10.5 trillion USD annually by 2025, up from $3 trillion USD in 2015" (Morgan, 2020; Morgan, 2022). Businesses are recognizing the importance of implementing cyber defense systems and are starting to augment software and applications with Artificial Intelligence (AI) that can process and analyze data on scales unimaginable for humans (Sarker, 2022).

Artificial Intelligence (AI), and its subfields, Machine Learning (ML), and Deep Learning (DL) are being explored as effective mechanisms to dynamically process, analysis, and predict attacks before they occur. The use of AI techniques in IDSs is the most successful solution to address the problems in cybersecurity (Maimo et al., 2018; Saranya et al., 2020; Sun et al., 2020; Yange et al., 2020; Halbouni et al., 2022; Sarker, 2022).

Artificial Intelligence is the ability of a machine to complete a task that normally would require human intelligence to complete. Artificial Intelligence uses data and algorithms to train machines to imitate intelligent human behavior for problem solving (Halbouni et al., 2022).

Machine Learning is a subset of Artificial Intelligence that uses data and algorithms to teach a machine to learn like humans learn, gradually improving from experiences over time. Machine Learning uses raw data and feature selection algorithms to classify data based on patterns, that the machine then analyzes, predicts, and reacts, to complete a specific task or solve a specific problem (Halbouni et al., 2022).

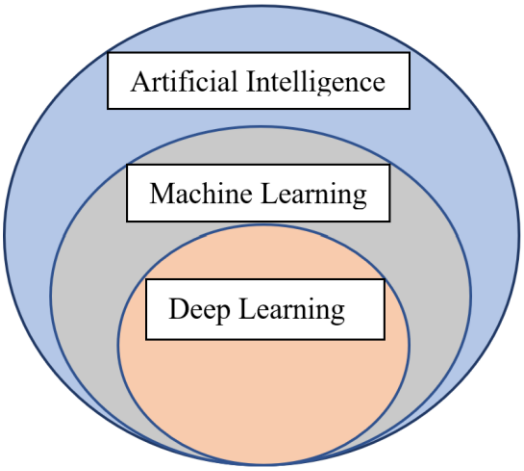
Deep Learning is a subset of Machine Learning that uses complex algorithms and deep Neural Networks (NN) that allow a machine to learn multiple complex tasks, similar to the way a human brain works. Deep machine learning approaches use multiple hidden layers of algorithms to analyze various kinds of data such as voice, text,

and audio, to create/find patterns for decision making. Halbouni et al. (2022), found that, “many studies and experiments have shown that deep learning is superior to machine learning because it can handle more complicated problems with greater accuracy and lower false alarm rates” (p. 19583). Rao et al. (2021), stated that “… unsupervised Deep Learning-based techniques [have] achieved remarkable breakthrough[s]” (p. 78). Deep Learning-based IDSs can improve detection rates of machine learning-based IDSs. Sarker (2022), and Sun et al., (2020) agree, stating that in many cases, Deep Learning (DL) outperforms other machine learning algorithms, especially when learning from big data sets. According to Henry et al. (2023), IDSs can raise the critical flag to help identify malicious network traffic. Stating that current approaches to IDSs are moving towards deep learning techniques. Due to deep learning models’ superiority of attack detection and lower false alarm rate this research includes two deep learning models for comparison. The relationship of AI and its sub-learning domains can be found in figure 1.

When referring to a machine that can learn and adapt based on data, the terms Artificial Intelligence, Machine Learning, and Deep Learning are often used interchangeably. For this research the term Machine Learning is used to refer to general AI learning techniques, while the term Deep Learning is used when referring specifically to the Deep Learning neural network models.

Figure 1

Relationship of Artificial Intelligence and its Subsets



There are four main Machine-Learning techniques, each with their own learning approaches and algorithms. They are 1) Supervised Learning 2) Unsupervised Learning, 3) Semi-supervised Learning, and 4) Reinforcement Learning. Supervised learning uses labeled data, unsupervised learning uses unlabeled data, while semi-supervised learning and reinforcement learning techniques can use both labeled and unlabeled data (Saranya et al., 2020; Henry et al., 2023).

Advancements in technology have led to the use of Machine Learning techniques in various applications. Among them is their use in Intrusion Detection Systems (IDS).

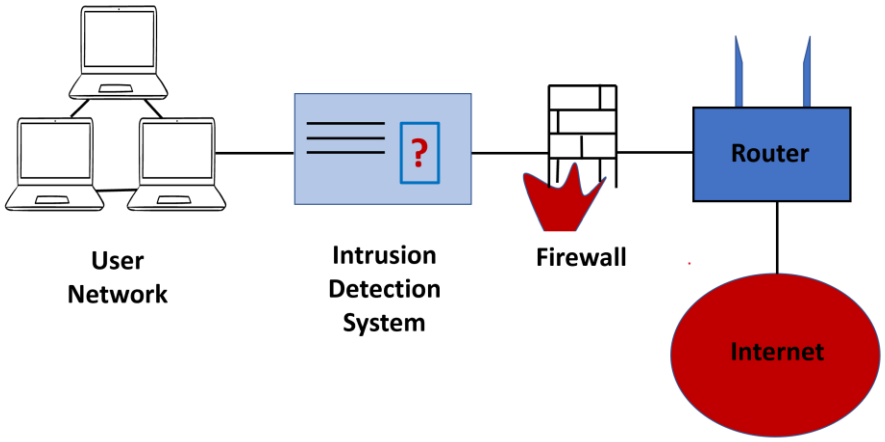
Intrusion Detection Systems (IDS) are the most important technology to secure network systems (Saranya et al., 2020; Sun et al., 2020; Liu et al., 2021; Meemongkolkait & Suttichaya, 2021; Pingale & Sutar, 2022). Patil et al., (2019), citing

Modi et al., (2013) and Modi and Acha, (2017), stated that cybercriminals can bypass firewall policy using the weaknesses of the operating system, but IDSs can detect and prevent intruders. An Intrusion Detection System is passive software / hardware that monitors network traffic for abnormal behavior. A properly functioning IDS logs inbound and outbound network traffic, constantly monitors and analyzes network activities for anomalies, and sends immediate alerts if there is abnormal network behavior (Tchakoucht & Ezziyyani, 2018). There are two main categories of Intrusion Detection Systems, based on the network infrastructure and location of the IDS in the network. The first category of IDS is 1) Host-based Intrusion Detection System (HIDS). This system places the IDS on the hosts' computers. HIDSs monitor traffic and system devices with access to the organization's inner network and the internet.

The second IDS category is 2) Network-based Intrusion Detection System (NIDS). A NIDS places the IDS in the network server just behind the firewall. A NIDS examines all inbound and outbound network traffic, analyzing data packets gathered from network devices such as routers (Halbouni et al., 2022). Since this research focuses on detecting attacks from outside the network, NIDSs are the IDS category of focus. See figure 2 for a visual of a NIDS scheme.

Figure 2

Network Intrusion Detection System Scheme



Due to the increased likelihood of attacks on network devices, new devices such as mobile phones are being developed with an IDS built into the hardware / software as an extra layer of security.

Within the two categories of IDSs, there are three major classes of Intrusion Detection Systems, 1) Signature-based / Misuse Intrusion Detection Systems, 2) Anomaly-based / Behavioral-based Intrusion Detection Systems, and 3) Hybrid-based Intrusion Detection Systems (Maimo et al., 2018; Almasoudy et al., 2020; Kanna & Santhi 2022; Mari et al., 2023; Henry et al., 2023).

Signature-based IDSs find signatures and patterns on the network and compares the patterns to a database of known attack signatures / patterns stored on the network. Advantages of Signature-based IDS are that they are easy to develop and require few computational resources. However, they tend to have high false-negative rates

(claiming normality when it is actually an attack). Signature-based IDSs can also only detect previously known attacks stored in a predefined attack database. New attacks go undetected and can breach the system. Maintaining a complete, up-to-date database manually can be time-consuming since new attack tools and techniques are prevalent and continually evolving (Maimo et al., 2018; Almasoudy et al., 2020; Kanna & Santhi 2022; Al-Yasen et al., 2022; Pingale & Sutar, 2022; Henry et al., 2023).

Anomaly-based Intrusion Detection Systems determine attacks based on statistical analysis. The normal network pattern is determined and set as the baseline pattern for the system. The Anomaly-based IDS monitors the network for abnormal patterns and deviations from the baseline to determine if an attack has occurred. A great advantage of Anomaly-based IDSs is that they can detect new unknown threats, however, they require time to train the system and establish the baseline. Anomaly-based IDSs tend to have a high false-positive rate (claiming it is an attack when it is actually not an attack) (Maimo et al., 2018; Lamrini et al., 2018; Almasoudy et al., 2020; Liu et al., 2021; Al Jallad et al., 2020; Al-Yasen et al., 2022; Albahar & Binsawad, 2020; Sun et al., 2020; Kanna & Santhi 2022; Henry et al., 2023; Mari et al., 2023).

Hybrid-based Intrusion Detection Systems combine signature-based IDS(s) and anomaly-based IDS(s) to incorporate the strengths of both models into one efficient system. Hybrid-based IDSs combine multiple types of machine learning algorithms to improve the performance of a single classifier. Hybrid intrusion detection systems are used to reduce irrelevant features and combine multiple classifiers, producing higher

efficiency in detection of all attack types (Maimo et al., 2018; Sornsuwit et al., 2019; Albahar & Binsawad, 2020; Saranya et al., 2020; Sun et al., 2020; Kanna & Santhi 2022).

The main security goal of an IDS is to identify an attack on a network. Attacks are any actions that attempt to compromise the Confidentiality, Integrity, and / or Availability of a system (Lamrini et al., 2018; Tchakoucht et al., 2018; Meemongkolkiat & Suttichaya, 2021). According to Yange et al. (2020), “most intrusions occur via network utilizing network related protocols to attack their targets” (p. 21). There are four major attack classes: 1) Denial of Service Attack (DOS), 2) Probe, 3) Remote to Local (R2L), and 4) User to Root (U2R). Denial of Service Attacks occur when the targeted system’s services are interrupted by the attacker(s) flooding the bandwidth or resources with illegitimate requests. Probe attacks aim to gather information from the targeted system from a source (often external). A Remote to Local (R2L) attack is an exploitation of system vulnerabilities in an attempt to gain access to a system’s account. A User to Root (U2R) attack occurs when a user tries to gain super user privileges (Tchakoucht et al., 2018). This research exploits all four attack classes to create a realistic attack environment for IDS evaluation.

Advancements in technology have helped people connect globally but have also created new tools and techniques for cybercriminals. The number of cyber breaches and total number of data records exposed per breach continue to grow. The sale of Big Data is one of the largest and fastest growing industries (Cisco, 2020). Existing tools

and techniques to protect data and network security include firewalls, access control mechanisms, and traditional intrusion detection systems used to detect and prevent network attacks are no longer suitable for detecting threats (Maimo et al., 2018; Saranya et al., 2020; Albahar & Binsawad, 2020; Sun et al., 2020; Yange et al., 2020; Sarker 2022).

The increase in Big Data and profitability of cybercrimes, and the increase in sophisticated attack tools and attack surface area have led to the need for research and development of hybrid multi-layered dynamic intrusion detection [and prevention] systems (Brooks, 2022). Due to the complexity and amount of data transmission across the network, it is important to implement a multi-layered dynamic protection system (Riera et al., 2022; Sornsuwit et al., 2019; Saranya et al., 2020). An attacker's first step to gaining access to a system from external networks begins with breaching the network perimeter. This is where a network intrusion detection system plays a crucial role. An IDS can protect data and the system from external network penetration. IDSs are the first line of defense for detecting and mitigating risk (Oluoha et al., 2021; Yange et al., 2020; Brooks, 2022).

This research compares the performance of a hybrid multi-layered dynamic Machine Learning-based IDS to its individual components to determine if a hybrid multi-layered combination machine-learning method can maintain accuracy while lowering false-positive, and false-negative rates. The performance of the multi-layered combination IDS is also compared to two Deep Learning-based IDSs to determine

accuracy, false-positive, and false-negative rates. Results from Riera et al. (2022) showed that a "… combination of machine learning algorithms and multi-phase models lead to improved prediction of web attacks" (p. 1). Sornsuwit et. al. (2019) and Saranya et al. (2020) had similar results, finding that multi-layered machine learning models were more efficient at detecting attacks on the network. Meemongkolkiat and Suttichaya (2021) stated that "Currently, several Machine Learning (ML) techniques are used to design and implement IDS[s] since ML techniques can capture the complex nature of cyberattacks" (p. 1).

According to the 2022 Allianz Risk Barometer, cyber incidents are the #1 concern of global businesses (Allianz, 2022). In the past few years cyber incidents have increased, and that number is expected to increase 15% every year for the next 5 years, and cost an estimated $10.5 trillion USD annually by 2025, not including the privacy, and societal impacts, nor human casualties (Morgan, 2020; Morgan, 2022). This paper analyzes and compares five different Intrusion Detection System Learning Models to determine strengths and weaknesses in accurately detecting attacks in an effect to lower false-positive rates and direct future research of IDSs.

Problem Statement

The increase in network traffic and profits from datamining have encouraged cyber-attacks. The exponential increase in security breaches has created a need for more dynamic, smart intrusion detection systems. There is a need for IDSs that can

analyze, detect, and predict unknown attacks from large amounts of data, moving from multiple types of devices, in various forms, over fast-moving networks. Many Machine Learning methods have been explored, each with their strengths and weaknesses based on dataset parameters, environment domain, and attack types. Unsupervised and Supervised Learning methods produce high false-positive and false-negative rates respectively. Research is needed to determine if combining Machine Learning techniques can maintain detection accuracy while decreasing the false-positive and false-negative rates. Two Deep Learning-based IDS models will also be explored as a possibility for maintaining detection accuracy while decreasing the false-positive and false-negative rates.

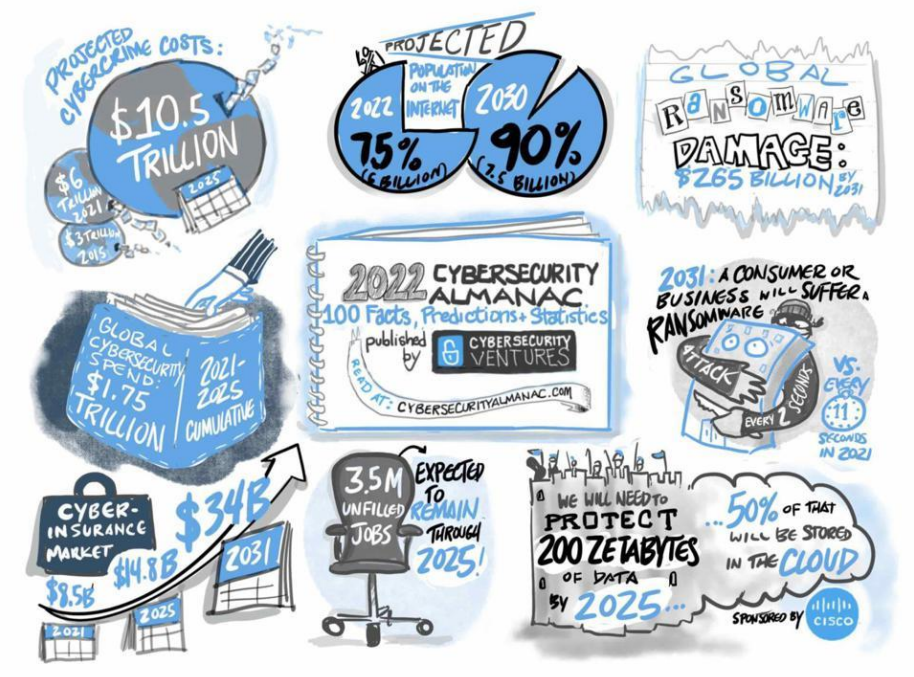
If the losses inflicted by cybercrime in 2021 ($6 trillion globally) were measured like the economic GDP of a country, cybercrime would have the third-largest economy, right after the USA and China (Morgan, 2020; Morgan, 2022). According to Cybersecurity Ventures, the loss from global cybercrimes is expected "…to grow by 15 percent per year over the next five years, reaching $10.5 trillion USD annually by 2025, up from $3 trillion USD in 2015" (Morgan, 2020; Morgan 2022). It is not a matter of if, but a matter of when the next attack will occur. Businesses, small and large are heeding the warnings and incorporating Machine Learning-based Intrusion Detection Systems to detect and predict new attacks and keep data, privacy, and systems safe (Maimo et al., 2018; Halbouni et al., 2022; Al-Yasen et al., 2022; Sarker, 2022).

Nature and Significance of the Problem

As stated earlier, the Allianz Risk Barometer showed that a cyber incident is the #1 concern of global businesses (Allianz, 2022). In the past few years cyber incidents have increased, and that number is expected to increase 15% every year for the next 5 years, and cost an estimated $10.5 trillion USD annually by 2025, not including the privacy, and societal impacts, nor human casualties (Morgan, 2020; Morgan, 2022). Figure 3: Statistical Predictions of Cybercrime shows the importance of developing improved methods for detecting and preventing cyberattacks through systems such as IDSs.

Figure 3

Statistical Predictions of Cybercrime with Permission from Morgan (2022)



This research analyzed and compared five Intrusion Detection System Learning Models, including a hybrid multi-layered dynamic IDS, to determine strengths and weaknesses in attack detection accuracy, false-positive rate, and false-negative rate in an effect to determine how complexity of layering algorithms effects attack detection performance. To gain a comprehensive attack environment, this research utilized the NSL-KDD Dataset and all four attack types (DoS, Probe, R2L, U2R). This research is intended to help direct future research and development of efficient dynamic Intrusion Detection Systems.

Objective of the Study

This research combined Unsupervised and Supervised Machine Learning algorithms to create a hybrid multi-layered dynamic IDSs and compared performance measures of accuracy, false-positive and false-negative rates to the individual algorithm results. The hybrid IDS was also compared to two Deep Learning-based IDSs looking at accuracy, false-positive, and false-negative rates.

Researchers like Sun et al. (2020), Riera et al. (2022) and Henry et al. (2023) have demonstrated the accuracy and importance of combined hybrid machine learning and deep learning algorithms to improve detection accuracy and decrease false-positive rates. This research analyzed Machine Learning and Deep Learning-based IDSs to assist future researchers in the development of accurate IDSs with lower false-positive

and false-negative rates. The term “outperform(ed)” is used to describe a model’s comparative increase in performance accuracy.

Research Questions

This research focused on three questions:

1) What are the effects of combining Supervised and Unsupervised Machine-based Learning methods on the accuracy, false-positive rate, and false-negative rate of a hybrid multi-layered intrusion detection system?
2) How does the accuracy, false-positive rate, and false-negative rate of the hybrid multi-layered IDS compare to the performance of its individual components?
3) How does the hybrid multi-layered IDS’s performance compare to Deep Learning-based IDSs?

Hypotheses

1) Ho: HYBRID = PCA = SVM = ANN #1 = ANN #2
ie. all models will perform equally.

2) H1: HYBRID >= PCA, SVM
ie. the hybrid model will outperform its individual components.

3) H2: HYBRID <= ANN #1, ANN #2
ie. the deep learning models will outperform the hybrid model.

Limitations of the Study

This research had some complications and limitations, however being aware of these considerations allowed for reduced impact on the outcomes.

The NSL-KDD dataset is contrived simulated data. Knowing this prior to building the machines for training and testing, multiple datasets were researched, and the NSL-KDD dataset thoughtfully selected because it is recognized as the benchmark for IDS data and widely used in machine learning-based IDS models (UNB, n.d.; Sapre et al., 2019; Mari et al., 2023). In fact, the NSL-KDD dataset, and its previous version KDDCup99 are the most widely used datasets for machine learning training and testing (Sapre et al., 2019). There is a lack of network-based IDS public data (UNB, n.d.), a lack of available labeled data (Lamrini et al., 2018), a lack of complete attack data (API attacks are often not included), and a lack of real-world network traffic data available for researchers. The NSL-KDD dataset is reasonable in size, making it efficient for running experiments without needing to randomly split the dataset into smaller train and test datasets, saving time and CPUs. Also, since the NSL-KDD dataset is widely used, it is a good baseline for comparing the proposed hybrid IDS model with other model results. Researchers say the NSL-KDD dataset accurately reflects current network traffic and known cyberattacks (UNB, n.d.) and based on the study by Sapre et al. (2019), is of much better quality than the KDDCup99 dataset. Future research could be expanded to include additional simulated datasets, or to test the hybrid IDS model in a live network environment to compare accuracy and false positive rates.

Another limitation to this study was that humans are still needed to code, train, and oversee the machines. Data cleansing/preprocessing can be a time consuming and redundant process, but preprocessing is one of the most important aspects of

quantitative research and data analysis. Although some data cleansing aspects such as normalization and dimensionality reduction were performed in Visual Studio Code 2019 with Python, humans still needed to validate the results and determine if the model accurately represented the dataset(s). Statisticians and data scientists often say “garbage in, garbage out” (GIGO). To make data useful for data mining and business intelligence, data must undergo data cleansing and data transformation. According to Larose and Larose (2015), data cleaning and pre-processing is a crucial, time-consuming process, that can take 10% - 60% of the overall machine learning data collection and analysis process (pp. 20-21). Some researchers use the 80/20 rule planning that pre-processing takes 80% of the overall data analysis processing time.

Definition of Terms

The following terms are referenced in this research:

Attack – Attacks are any actions that attempt to compromise the Confidentiality, Integrity, and / or Availability of a system. There are 4 classes of attacks (Denial of Service (DoS), Probe, Remote to Local (R2L), User to Root (U2R)).

Intrusion Detection System (IDS) – Passive software or hardware that monitors network traffic to detect and alert if abnormal behavior occurred. Can be signature-based (misuse) or anomaly-based (behavior).

Signature-based (misuse) IDS / Supervised Learning – Intrusion Detection System that matches network data to labeled attack database. Simple to setup but requires resources for storage and only detects known threats. Produces high false negatives.

Anomaly-based (behavior) IDS / Unsupervised Learning – Intrusion Detection System that finds irregularities from the baseline model using statistical analysis. Requires time to train but can find new threats. Produced high false positives.

Artificial Intelligence – Artificial Intelligence is the ability of a machine to complete a task that normally would require human intelligence to complete. Artificial Intelligence uses data and algorithms to train machines to imitate intelligent human behavior.

Machine Learning – Machine Learning is a subset of Artificial Intelligence that uses data and algorithms to teach a machine to learn like humans learn, gradually improving from experiences over time. Machine Learning uses raw data and feature selection algorithms to classify data based on patterns, that the machine then analyzes, predicts, and reacts to complete a specific task or solve a specific problem.

Deep Learning – Deep Learning is a subset of Machine Learning that uses complex algorithms and deep Neural Networks (NN) that allow a machine to learn multiple complex tasks, similar to that of the human brain. Deep Learning machine learning approaches use multiple hidden layers of algorithms to analyze various kinds of data such as voice, text, and audio, to create/find patterns for decision making.

False-positive (FP) rate – Is the probability that a false data point will be predicted as a true. For an IDS this would mean that a threat was detected when no actual threat was present. This measurement is also called a Type I error.

False-negative (FN) rate – Is the probability that a true positive data point will be predicted as a false. This measurement is also called the miss rate or a Type II error.

Principal Component Analysis (PCA) – Is an unsupervised machine learning algorithm used for dimensionality reduction. PCA outputs a new set of variables called principal components that can be input for supervised machine learning algorithms (Saranya, et. al., 2020; Larose & Larose, 2015).

Support Vector Machine (SVM) – Is a supervised machine learning algorithm used for both classification and regression. It is the most accurate of any classifier, good at generalization, and processing of any size dataset. SVM uses labeled data and outputs the separation of the data into classes by the hyperplane that maximizes the margin of all attack classes. SVM can be used for multi-class classification by implementing it in a cascade manner. It is often developed on the kernel used and parameters (Saranya, et. al., 2020, Larose & Larose, 2015).

Autoencoder – Is a deep learning, neural network classification method. It is composed of many hidden layers. Autoencoders are one of the primary ways of developing unsupervised learning models. The model consists of three components (input -

encoding, bottleneck - compression, output - decoder). It uses unlabeled data, compresses, and encodes it, and then reconstructs the data for representation.

NSL-KDD Dataset – The improved version of the KDD'99 Dataset, which is considered the benchmark for IDSs. The NSL-KDD Dataset removed data redundancies from the KDD'99 Dataset so that the data is not skewed towards the more frequent records and attack types, lowering the false-positive rate.

Summary

Advancements in technology have created challenges for detecting attacks on dynamic network systems. Due to the variety and volume of data, and smart tools and techniques used by attackers, existing intrusion detection systems are not capable of monitoring all network traffic nor detecting new attack types. That is where Machine Learning-based Intrusion Detection Systems can assist. They are the most important technology for securing network systems (Meemongkolkiat & Suttichaya, 2021; Maimo et al., 2018; Lamrini et al., 2018; Albahar & Binsawad, 2020; Saranya et al., 2020; Sun et al., 2020; Yange et al., 2020; Halbouni et al., 2022; Sarker, 2022), and the solution to these challenges, which is why they are explored in this research.

The remainder of this paper is structured as follows: Section 2 is a literature review summarizing the past and current research about machine learning-based Intrusion Detection Systems. Section 3 explains the methodology of the proposed hybrid multi-layered dynamic IDS model. Section 4 presents the research data and

completed statistical analysis, and Section 5 contains the research results, conclusions, and future research recommendations.

Chapter II: Background and Review of Literature

Introduction

The following literature review outlines the research results and recommendations from the most current research on machine learning and hybrid multi-layered IDSs. Additionally, research about popular intrusion detection system datasets is reviewed and outlined.

Background and Literature Related to the Problem

While determining relevant research, more than 80 journal articles were read and reviewed from journals including ELSEVIER, IEEE, Journal of Applied Artificial Intelligence, ICCEE Journal of Physics, Journal of Computer Networks, Journal of Information Security, and others. Research article topics focused on Intrusion Detection Systems, Machine Learning, and Anomaly Detection. The article publication date was limited to after 2017. Some search phrases included “Supervised Machine-learning based Intrusion Detection Systems to detect anomalies”, “Unsupervised Machine-learning techniques for NIDS”, and “Machine Learning and Deep Learning-based Intrusion Detection Systems for anomaly detection”.

Formulas for calculating model attack detection accuracy, precision, recall, and F1-score were researched as well as formulas for false-positive rate and false-negative rate. Information about displaying test results of IDS was researched and confusion matrices suggested, so further research was conducted on statistical analysis and displaying data results.

Artificial Intelligence, Machine Learning, and Deep Learning were researched, and articles read to determine their current usage in dynamic IDS and suggested future research gaps.

To determine which learning models and algorithms to use for this research, articles detailing learning models and learning algorithms were compared. The research suggests the most promising models include unsupervised, supervised, and deep learning methods, each with advantages and disadvantages. Supervised Learning models are easier to code but use more storage and processing to access the database of known attacks. It is difficult to maintain accurate labeling of data in Supervised Learning models due to the volume, variety, and speed of data transmission. Unsupervised Learning models can better detect and predict unknown attacks than Supervised-Learning models, however they tend to have a high false-positive rate (Larose & Larose, 2015; Saranya et al., 2020).

Research of the most widely used IDS datasets was performed, with the KDD'99 and its updated version NSL-KDD being the top performers due to the good size of the datasets, clear labeling, and use of four general categories of attacks (Sapre et al., 2019, UNB, n.d.). The NSL-KDD Dataset removed redundancies in records and attack type, reducing imbalances in the data (Latah et al., 2018; Sapre et al., 2019; Sornsuwit et al., 2020; UNB, n.d.). NSL-KDD Dataset also has predefined training and testing datasets allowing consistent testing parameters. Another widely used IDS Dataset is CIDIS2017, although the dataset is newer than the KDD family of datasets, CIDIS2017

does not include Active Persistent Threats (APT), and it is a very large dataset requiring cleansing and subset data selecting for training and testing (Sun et al., 2020; UNB, n.d.).

Research about designing and testing machine learning IDS in online applications such as SNORT and Jupyter were performed to determine the best option for this research. Both SNORT and Jupyter are flexible, open-source applications. SNORT is a rule-based Intrusion Detection [Prevention] System. There is an open community ruleset, and a subscription ruleset. SNORT is known to have difficulty keeping track of all network activity. It is unable to process high volumes of data, dropping data packets. SNORT also produces high false-positives rates (Cisco, 2022). Jupyter notebooks provide interactive computational environments for developing data science applications, such as machine learning models. Jupyter notebook is a server that allows in-browser editing, execution of code, and computation result display (Verma, 2021). The flexibility in programming language, large selection of machine learning algorithms, and scale of dataset usability made Jupyter a good fit for this research.

Riera et al. (2019) suggests combining various machine learning algorithms to increase accuracy and decrease false-negative and false-positive rates. Meemongkolkiat and Suttichaya (2021), Halbouni et al. (2022), Sun et al., (2020), and Yange et al. (2020) agree that machine learning IDS models are the most important technology for securing network systems. With the exponential growth in big data, and

advancements of attack tools and techniques, additional research of hybrid multi-layered dynamic Machine Learning-based IDSs is needed to detect and prevent current and future network cybersecurity breaches.

Literature Related to the Methodology

Mari et al. (2023), study focused on a hybrid machine learning-based IDS that used the NSL-KDD dataset and several algorithms. Adversarial network traffic was also generated and used to test the IDS's performance. The results showed that using generated traffic (that could avoid detection by the IDS), improved the proposed intrusion detection system's detection performance.

Riera et al. (2022), presented a combination of machine learning algorithms and multi-phase model for improved website attack prediction. Using a new multi-label dataset classification model and machine learning the results proved more accurate at predicting website attacks. More thorough labeling of traffic in new datasets was suggested.

Al-Yasen et al. (2022), presented an anomaly-based IDS that used Differential Evolution (DE) as a feature selection method with Extreme Learning Machine (ELM) classifier to select reduce features to improve model performance accuracy. Using the NSL-KDD datasets, the approach helped speed up processing time, increased accuracy (reducing the number of features to 9 features was 80.15% verses including all features was 76.44%) and reduced the false positive rate. Future research suggestions include

connecting the model to a live network, and using a complex classifier to increase the detection of U2R attacks).

Mijalkovic and Spognardi (2022), compared performance results of a proposed deep learning-based IDS model using the NSL-KDD and UNSW-NB15 datasets for binary and multiclass classification. The primary goal was to lower the false-positive rate and increase the detection of the minority attack types. The results showed the importance of training/testing subset distribution, feature selection, and class weight usage. The results found that in almost all experiments using feature selection and class weights lowered the false positive rate. More significantly, the class weights provided the best method for increased detection of minority classes. Suggestion for future work included improving dataset imbalances by adding minority attack classes, using automatic parameter tuning methods, and testing the proposed model in a live network to determine performance results.

Sapre et al. (2019), compared the KDDCup99 and NSL-KDD datasets using various machine and deep learning-based intrusion detection systems and the results were surprising. Results showed that the SVM and other machine learning models outperformed the Artificial Neural Network (ANN) and sophisticated Random Forests models using both datasets, with the SVM model outperforming all models in binary classification attack detection. This shows that the complexity of the model does not dictate better performance results. Also, surprisingly, contrary to other studies, the NSL-KDD dataset did not reduce false positives, nor detect more rare attack types as

expected compared to the KDDCup99 dataset. The authors said this was most likely due to the random splitting of the datasets for training and the features that were removed during pre-processing. It was concluded that the NSL-KDD dataset is of better quality than the KDDCup99 based on model results and comparison. Future research ideas include a layered machine learning model to improve the accuracy and low recall value obtained from the NSL-KDD dataset. Expanding the study to include additional machine learning algorithms was suggested.

Rao et al. (2021), proposed a novel two-stage hybrid NIDS that included Spare Autoencoder, and Deep Learning Neural Network (DNN). The proposed model performed better than other models and had a low false positive rate. The proposed model used multiple datasets including the NSL-KDD dataset. Future research suggestions included incorporating various autoencoders for additional dimensionality reduction and to save processing time.

Maimo et al. (2018), proposed an adaptive deep learning-based system for anomaly detection in 5G networks. A two-level deep machine learning model was proposed. Based on the limitations of resources a single layered deep learning model was suggested for future studies. Suggested future work included detecting and classifying threats to determine which single layered deep learning model is more appropriate for configuration and given throughput requirement. The use of real data to detect the accuracy of the anomaly detection architecture was also recommended.

Halbouni et al. (2018), noted the importance of new intrusion detection systems. Various machine learning and deep learning algorithms for intrusion detection were reviewed, with deep learning models found to be more accurate and have lower false alarm rates. It was recommended that new learning algorithms, new datasets, and merging of algorithms continue to be explored.

Lamrini et al. (2018), applied One-Class Support Vector Machine (OC-SVM) IDS model to detect anomalies in real network traffic with good results. Challenges for future work included how to monitor a real-time data stream without overwhelming the computing resources. A distributed framework for novelty detection was also proposed.

Sornsuwit et al. (2019), developed a new hybrid machine learning model for multiple cyber intrusion detection. Multiple algorithms were used in the hybrid model including correlation-based feature selection to reduce redundancies, and various datasets used to test the model. The results showed that the proposed model performed more efficiently than other systems and was able to detect new abnormalities hidden in the network.

Kaur & Singh (2020), proposed a hybrid IDS model that combined misuse and Deep Learning Recurrent Neural Network (RNN) algorithms to update a signature repository of an IDS. The model was successfully implemented on a live network and proved to detect web and other attack types. Future research recommendations included continuing to improve the proposed model, by training the model with a larger more recent dataset, working to reduce the attack detection time by incorporating a

GPU for processing, using an improved pattern matching algorithm, and continuing to improve the RNN to avoid vanishing gradient issues.

Sun et al. (2020), proposed a deep learning hybrid model consisting of feature extraction algorithms CNN - Central Neural Network and LSTM - Long Short-Term Memory. Category weights were used to optimize the unbalanced attack type data by reducing the number of attack type features. The results showed the proposed deep learning model performed better than the compared machine learning models. The deep learning model however had a low accuracy detection rate for certain types of attacks. Incorporating additional network traffic features was suggested for future research.

Al Jallad et al. (2020), proposed an anomaly-based deep learning IDS to increase the model's ability to generalize attack data and reduce the false-positive rate. The study used the NSL-KDD benchmark dataset to compare performance results of the proposed model to other traditional IDS optimizers. The proposed model resulted in a 10% decrease in false-positive rate compared to traditional models. Future research into deep learning-based IDSs and big data is suggested to increase detection accuracy and reduce false-positive rate.

Al and Dener (2021), proposed a new multi-classification Hybrid Deep Learning (HDL) NIDS consisting of Convolution Neural Network (CNN) and Long Short-Term Memory (LSTM). Nine different machine learning algorithms were compared to the proposed IDS model. Results found that the proposed model achieved good results in

detecting network attacks in imbalanced datasets, reaching accuracies of 99.83% (multiclass) and 99.17% (binary classification). Future research suggestions included combining different hybrid deep learning algorithms for accuracy, and for data balancing, and to test the proposed model in a live big data environment.

Kanna and Santhi (2022), proposed a hybrid IDS model using Convolutional and Long Short-Term Memory (LSTM) Neural Networks. The proposed model performance results were compared using multiple datasets including the NSL-KDD dataset. The model performance accuracy ranged from 97% - 98% for all datasets and had few false positives, shorter run time, and better classification results than other models. Future research suggestions included testing the proposed model using other datasets and examining real-world malicious traffic records to improve IDS models.

Pingale and Sutar (2022), proposed a new hybrid deep learning-based IDS that used Convolution Neural Network (CNN) to extract features and transform them into a vector format for further analysis. Deep Maxout Network and Deep Auto Encoder algorithms were used. The proposed model achieved good results (92% - 93% accuracy). Future research suggestions included utilizing deep learning classifiers and larger datasets to increase accuracy, and to use the Remora Optimization Algorithm (ROA).

Mohd et al. (2021), used the updated version of KDD’99 (NSL-KDD) dataset to test a hybrid hierarchical IDS, separating each attack type into a separate classifier level

for analysis. The model achieved the highest accuracy with a Smooth Support Vector Machine (SSVM) classifier algorithm. It was found that at various levels of attack classifiers, different algorithms performed more accurately, and concluded that hierarchical classifier-based IDSs were successful when comparing hierarchical classifiers. No future research direction was given.

Tchakoucht et al. (2018), proposed a lightweight IDS model to detect Probe and DoS attacks on the network with good accuracy and reduced computation time. Using the KDD'99 Dataset, results showed good accuracy and low false positive rate (Probe 90%, 0.5%, DoS 99.6%, 0.3%). Future work suggested the need to include all 4 attack types and to test in a real environment.

Liu et al. (2021), proposed a machine learning intrusion detection system model to increase accuracy and reduce false positive rate. The model utilized Generative Adversarial Networks (GAN) and feature selection to reduce dimensionality and rebalance the training dataset. Multiple baseline datasets were used for experimentation including NSL-KDD, UNSW-NB15, and CICIDS-2017. The results showed improved IDS performance compared to other models.

Almasoudy et al. (2020), used a wrapper feature selection IDS model based on Differential Evolution (DE) to reduce the number of features of the NSL-KDD datasets. The results showed good accuracy (80.15% and 87.53%) while reducing the false positive rate in some of the classifiers. Future research suggestions included using live

networks to test the model and using complex classifiers to better detect the U2R attacks that have behavior similar to that of “normal”.

Chen et al. (2022), looked at Machine Learning-based IDS models using the KDD’99 Dataset. Stating that the KDD’99 Dataset is the most popular dataset used to study IDSs. Chen et al. (2022), noted that there are few datasets that contain Advanced Persistent Threats (APT). The KDD’99 Dataset contains some APT data.

Yange et al. (2020), said that IDSs are needed to combat growing threats on computer networks. A Decision Tree IDS algorithm and the KDD’99 Dataset were used for model testing. Findings showed that the model had efficient attack detection and took less time to construct than other models, and that the size of the dataset and number of features had little to no effect on the efficiency of the system.

Latah et al. (2018), researched anomaly-based IDS for software defined networks. Using an anomaly-based IDS and the NSL-KDD Dataset, accuracy and false-positive rates were compared for the different classifiers, with PCA very successful for enhanced accuracy compared to other models. Suggestion for future research was to continue comparing machine learning models for achieving higher accuracy and lower false-positive and false-negative rates.

Saranya et al. (2020), performed a performance analysis on machine learning algorithms for IDSs. Using various machine learning algorithms and the KDD’CUP (KDD’99) Dataset it found that detection rate, false-positive rate, and accuracy are not

only dependent on the algorithm and the size of the dataset, but also the application area. Extending the study of machine learning algorithms in IDSs by taking real-time datasets was suggested.

Albahar and Binsawad (2020), developed and tested two new network intrusion detection system models and compared the models' performance results with other models. The models used an autoencoder layered neural network. The models' performances were tested using the NSL-KDD and UNSW-NP15 datasets, and both models' performance results were found to be better than that of other models. Future research was suggested to combine additional model algorithms such as the PCA to reduce features and improve accuracy. Also, hyperparameter tuning was suggested.

Sornsuwit et al. (2020), proposed a new hybrid machine learning IDS. Using various datasets, the NSL-KDD, and KDD CUP (KDD'99) datasets were among the top performers of efficiency compared to other models and could detect new abnormalities hidden in the network.

Mushtaq et al. (2023), proposed a novel hybrid deep learning-based IDS that utilized Autoencoder and Grated Recurrent Unit (GRU) algorithms. The proposed model used the NSL-KDD dataset to test binary and multi-class performance classifiers. The proposed model had good accuracy. The model was then tested on the UNSW-NB15 Dataset, and the model achieved good results. Exploring other deep learning models

was suggested for future research to address class imbalances for R2L and U2R attack types.

Henry et al. (2023), proposed a hybrid deep learning machine model that looked at various combinations of CNN (central neural network) and GRU (Gated Recurrent Unit interactive testing framework) sequences to improve network attack detection accuracy and false-positive rate. The results were compared to other models, showing that the proposed hybrid model had comparative results and could be used as an IDS in a real network environment.

Summary

As you can see from this literature review, research suggests the need for continued development and comparison of various machine learning IDSs to determine methods for achieving higher attack detection accuracy, and lower false-positive and false-negative rates. Further research of hybrid IDS models and deep learning anomaly-based IDSs was suggested. Research has shown that hybrid multi-layered machine learning-based IDSs and deep learning-based IDSs can achieve higher performance accuracy and lower false-positive rates when compared to conventional IDSs, however additional research is suggested.

As far as we can tell, little to no research has compared the performance of a hybrid multi-layered dynamic machine learning-based IDS to the results of its individual

components to learn what features can increase performance accuracy and lower false-positive rate. Due to these research gaps and suggestions, this research is proposed.

Chapter III: Methodology

Introduction

This section discusses the methodology used in this research. The models, dataset, and tools used to design, train, and test the intrusion detection system models are explained in detail.

Design of the Study

This is a quantitative study that determined the accuracy, and false-positive and false-negative attack detection rates of a hybrid multi-layered dynamic Intrusion Detection System compared to the performance of its individual components. The detection accuracy of the proposed hybrid IDS was also compared to two Deep Learning-based IDSs to see how the hybrid model compared in terms of performance accuracy. A quantitative study was chosen because it provided consistent variables for comparison between the various models' performance.

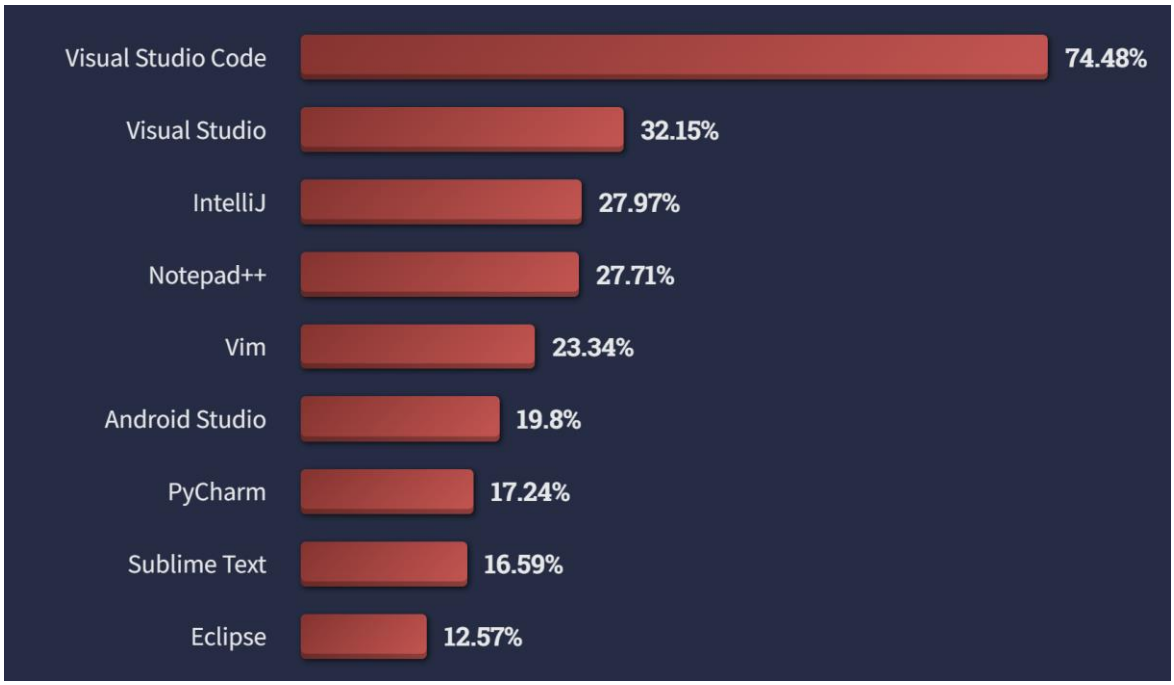
Data Collection

The research models were designed, trained, and tested using Excel, Visual Studio Code 2019, Anaconda, and Jupyter notebook (described later in this research), programmed using Python coding language. As shown in figure 4, according to a 2022 survey performed by Stack Overflow, Visual Studio Code is the most popular Integrated Development Environment (IDE) with more than 74% of respondents (up from 71% in 2021) stating they have worked with and want to continue to work with Visual Studio

Code for future environment developments such as those needed for analysis of machine learning IDSs (Stack Overflow, 2021; Stack Overflow, 2022).

Figure 4

The Most Used IDEs with Permission from Stack Overflow (2022)



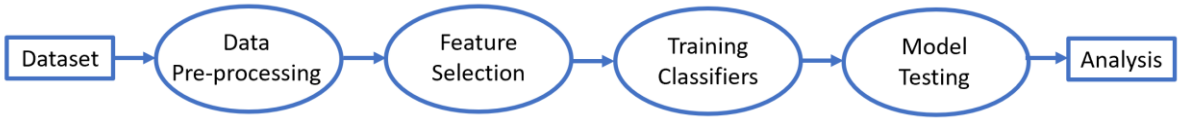
Visual Studio Code is a great tool for data analysis because of its many extensions and customizability. Data scientists can perform all data analysis tasks in Visual Studio Code (Visual Studio Code, 2023; Stack Overflow, 2021; Stack Overflow, 2022). “A good Visual Studio Code setup can make you a more productive data scientist” (Ferreira, 2022). According to Ferreira (2022), the combination of Visual Studio Code and its extensions make it “on a whole other level compared with the alternatives”. Jupyter notebook was used because it combines executable code, equations, and visualizations

into one location for simple reference and manipulation (Visual Studio Code, 2023). Python was the chosen programming language because Python is the language of data scientists and machine learning (Visual Studio Code, 2023; Terra, 2023). Python is a free, open-source general programming language. It is human readable and easy to use, requiring fewer lines of code than needed by older coding languages. Due to Pythons simplicity, it is widely used. In 2020, Python became the fourth most used programming language (Pruciak, 2021). Python is greatly used by developers and in academia, so there is a large community for assistance, and many libraries are available (Terra, 2023).

Following the general methodology for research and data analysis (as seen in figure 5), the method for developing the hybrid IDS model consisted of the following phases: 0) Dataset, 1) Data Pre-processing, 2) Principal Component Analysis – training and testing for dimensionality reduction and feature extraction, 3) Support Vector Machine – training and testing for classification and regression, and 4) Analysis.

Figure 5

Research Methodology Phases



Phase 0: Dataset

For this research the NSL-KDD Dataset was used for training and testing the hybrid multi-layer dynamic IDS, its individual components, and the Deep Learning-based IDSs. The NSL-KDD Dataset was chosen because it is the improved version of the KDD'99 dataset, which is considered the benchmark dataset for Intrusion Detection Systems (Tavallaee et al., 2009; Sapre et al., 2019; UNB, n.d.). Attack redundancies that were present in the KDD'99 version have been removed from the NSL-KDD Dataset, so the data is not skewed towards more frequent records and attack types, making it less prone to false-positive rates (Sapre et al., 2019; Sarker, 2022; UNB, n.d.). The NSL-KDD Datasets have 41 features, and 39 different attack types, from 4 attack categories (DoS, Probe, U2R, R2L), which were used in the feature selection phase of this research and to categorize the network traffic during training and testing. The feature and attack type columns can be found in Appendix A. The NSL-KDD Datasets were created at the University of New Brunswick (UNB, n.d.; Tavallaee, 2009; Sapre et al., 2019), consisting of over 150,000 data points of emulated network attack data (Ring et al., 2019). The data and attack record statistics can be seen in figure 6 for both the NSL-KDD Train+ and NSL-KDD Test+ data sets.

Figure 6

NSL-KDD Train+ and Test+ Dataset Statistics

NSL-KDD Dataset	Number of Records					
	Total	Normal	DoS	Probe	U2R	R2L
Train+	125973	67343 (53.00%)	45927 (37.00%)	11656 (9.11%)	52 (0.04%)	995 (0.85%)
Test+	22544	9711 (43.00%)	7458 (33.00%)	2421 (11.00%)	200 (0.90%)	2654 (12.10%)

The IDS models (pictured in figure 7) utilized the following machine learning algorithms:

1) Unsupervised Learning Model (green in figure 7)

– Principal Component Analysis (PCA)

2) Supervised Learning Model (red in figure 7)

– Support Vector Machine (SVM)

3) Hybrid Multi-layered Machine Learning Model (yellow in figure 7)

– Unsupervised Learning – Principal Component Analysis (PCA)

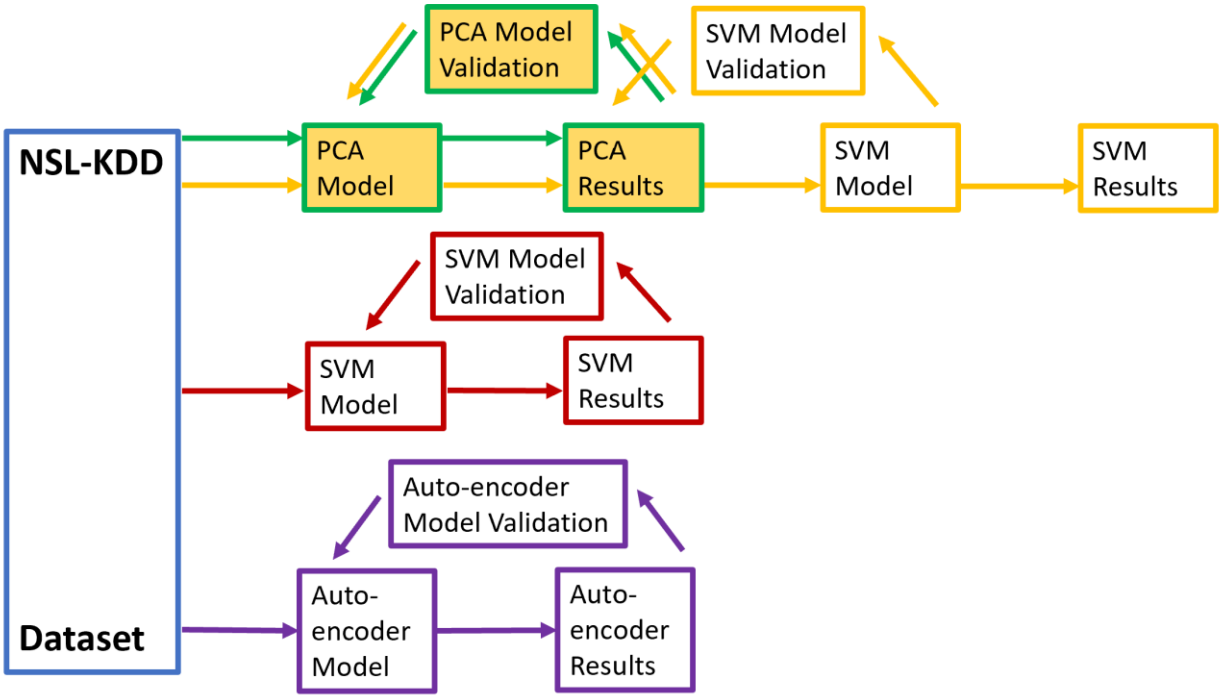
– Supervised Learning – Support Vector Machine (SVM)

4) Deep Learning (Artificial Neural Network) Model #1 and Model #2 (purple in figure 7)

– Unsupervised Learning – Autoencoder

Figure 7

Research Methodology Scheme



The following methodology was used for the training and testing of the hybrid machine learning model. The other model methodologies were similar, with slight variations of feature selection and algorithms.

Phase 1: Data Pre-processing

As part of the pre-processing and data exploration, records from both the NSL-KDD Train+ and NSL-KDD Test+ Datasets were 1) cleansed – looked at data for missing or duplicate values, and outliers, 2) transformed – created dummy variables

and normalized, and 3) reduced – feature extraction using the PCA. The initial data analysis and first pre-processing steps were completed in Microsoft Excel. To better understand the datasets, the records and features were examined. There were no null values, missing values, or outliers found in either the training or the testing datasets. Next the datasets were transformed. Although most of the dataset features were numerical, three features (Protocol type, Service, and Flag) and the attack type (Class label) were categorial. The categorical data was replaced with numeric dummy variables to allow mathematical computations and comparisons. For the complete breakdown of the dummy variables see Appendix B.

The datasets (Train+ and Test+) were then imported into Visual Studio Code 2019 and the identified explanatory/independent X variables and response/dependent y variable were separated into different datasets both for the training and testing data. The XTrain dataset was fit and transformed using StandardScaler (results can be seen in figure 8). StandardScaler standardizes (normalizes) the independent variables by subtracting the mean and dividing by the standard deviation. The XTest dataset was transformed using the StandardScaler to make the data suitable for the model, but not fit to the model as was the XTraining dataset. Not fitting the test dataset to the model helped avoid overfitting and maintained the test dataset as a naive dataset.

Figure 8

Standardized XTrain_Scaler Dataset

```
array([[-0.11015149, -0.44346643,  1.23437213, ..., -0.6248708 ,
         2.87440955,  2.75391372],
       [-0.11015149, -0.44346643,  1.23437213, ..., -0.6248708 ,
         2.87440955,  2.75391372],
       [-0.11015149, -0.44346643,  1.23437213, ..., -0.6248708 ,
         2.87440955,  2.75391372],
       ...,
       [-0.09786195, -0.44346643, -0.98573182, ..., -0.6248708 ,
        -0.38763462, -0.37638703],
       [-0.05868905, -0.44346643, -0.98573182, ..., -0.6248708 ,
        -0.38763462, -0.37638703],
       [-0.08442027, -0.44346643, -0.98573182, ..., -0.6248708 ,
        -0.38763462, -0.37638703]])
```

Phase 2: Principal Component Analysis (PCA)

The Principal Component Analysis (PCA) algorithm was then used for feature extraction and dimensionality reduction. The PCA transforms the large dataset into a smaller and uncorrelated dataset that is used later for training the Support Vector Machine (SVM). PCA has been found to reduce overfitting and improve clarity in observation (Latah & Toker, 2018; Larose & Larose, 2015). Based on the standard approach of PCA, this research used the following steps to select features, 1) extracted the features with the largest coefficients from the principal components, 2) selected the k eigenvectors that correspond to the k largest eigenvalues, 3) transformed the original dataset with corresponding features using the projection matrix from the k-selected eigenvectors, 4) validated the performance of the selected features and corresponding k component on the training results (Latah & Toker, 2018) using logistical regression.

The NSL-KDD scaled XTrain dataset was fit and transformed to the PCA model, and the scaled XTest dataset transformed using the PCA. The PCA analyzed the features and ranked them to determine the most closely correlated factors to Class label (attack type). The strongest correlated attribute features were selected to run a training optimization. The PCA variance ratio was completed across 1-18 components. The variance ratio found that setting the PCA to two components provided the strongest variability prediction, with component #1 making up 19.987% and component #2 making up 12.359% of the predictive variance.

Next a random data state of zero was chosen to check the fit of the scaled training X and y datasets to the logistic regression classifier. The predicted results were then graphed using a confusion matrix for clear visualization. The confusion matrix indicates the high predictive values on the diagonal, as seen in figure 9. Weights and biases were then used to observe the model's predictions to check model accuracy. The model data predictive accuracy was 100% in predicting the Class label.

Figure 9

Confusion Matrix Logistical Regression Test Results

```
[[18880     0     0]
 [    0  2621     0]
 [    0     0  1043]]
```

Phase 3: Support Vector Machine (SVM)

Next the PCA Training dataset (XTrain and yTrain) results were fit to the Support Vector Machine model (SVM) and the score from the trained model calculated. The XTest dataset from the results of the PCA test was then run through SVM model. The SVM score and predicted test results' accuracy was calculated and found to be 100% accurate at predicting the Class label (attack type). See Appendix C - F for the complete code of all models.

Phase 4: Analysis

The hybrid multi-layered dynamic IDS test results were analyzed and compared to the test results of its individual components to determine if combining supervised-learning and unsupervised learning algorithms into a multi-layered dynamic IDS can maintain attack accuracy while decreasing false-positive and false-negative rates. The hybrid multi-layered dynamic IDS's performance results were also compared to two Deep Learning-based models to determine if the detected attack accuracy can be maintained while decreasing false-positive and false-negative rates. The complete model analysis can be found in Chapter IV.

Tools and Techniques

The Statistical performance measurements that were used to determine model accuracy of attack detection were as follows:

False-positive (FP) rate is the probability that a false data point will be predicted as a true. For an IDS this would mean that a threat was detected when no actual threat was present. This measurement is also called a Type I error.

$$\textit{False-positive Rate} = \frac{FP}{FP + TP}.$$

False-negative (FN) rate is the probability that a true positive data point will be predicted as false. This measurement is also called the miss rate or a Type II error.

$$\textit{False-negative Rate} = \frac{FN}{FN + TP}.$$

Precision is the degree of exactness of measurements. Precision shows how close two or more measurements are to each other. The finer the degree of measurement, the more precise it is. The formula for precision is:

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

Recall is the proportion of actual positives that were identified correctly. It is a percentage of the predictions that the model accurately predicted. The formula for Recall is:

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

Accuracy is a measurement of how close a value is to the actual (true) value. Accuracy shows how close measurements are to a specific value. The formula for accuracy is:

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

F-score is used to measure the accuracy of a model on a dataset. It is used to evaluate systems that use binary classifications, such as IDS that often categorize a data point as either normal or an attack. The F-score is the average (harmonic mean) of the precision and recall measurements. A perfect model has an F-score of 1. The formula for a standard F-score is:

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

The true-positive and false-positive rates were displayed in a confusion matrix as a clear visual representation of the systems' results.

Summary

This section detailed the research methodology, including the tools, dataset, and models used to train, test, and analysis the performance of the various machine learning intrusion detection models. The quantitative performance metrics to determine and compare precision, accuracy, false-positive and false-negative rates, and F1-score were explained.

Chapter IV: Data Presentation and Analysis

Introduction

This section presents the research data collected for the training and testing of the five intrusion detection system models. The data findings and statistical performance measures are presented in charts for clear representation and model comparison. The performance results did not support the research hypotheses. Analysis indicated that the hybrid IDS model did not outperform the PCA, nor the SVM, but it did outperform the ANN models. The analysis results indicated the value of pre-processing and feature selection/feature extraction in improving accuracy and lowering false-positive rate.

Data Presentation and Analysis

Performance measures including accuracy, precision, recall, and false-positive rate for each model were calculated and compared. The results showed that many different machine learning, and deep learning models could accurately predict the attack types. It was found that the combination of pre-processing algorithms for feature extraction / dimensionality reduction and classification impacted the performance accuracy and reduced the false-positive rate prior to training and testing with the models' algorithms. The collected data results from the five intrusion detection system models are displayed in Table 1.

Table 1

Comparison of Hybrid IDS Model to SVM, PCA, and ANN Models

Model	Accuracy (%)	False Positive Rate (%)	True Positive Rate (%)	Precision (%)	Recall (%)	F1-Score	R^2 Score (%)
Hybrid Machine Learning Model	100.00	0.00	100.00	100.00	100.00	1.00	87.879
Principal Component Analysis – Logistical Regression	100.00	0.00	100.00	100.00	100.00	1.00	86.243
Support Vector Machine	100.00	0.00	100.00	100.00	100.00	1.00	99.998
Autoencoder Neural Network #1	97.49						
Autoencoder Neural Network #2	97.18						

All models performed well, with performance accuracy ranging between 97% - 100%. Surprisingly, the machine learning models outperformed the neural network models. This shows that adding layers and more complex algorithms does not necessarily equate to better performance accuracy. The autoencoder neural network models consisted of 3 – 5 layers of algorithms and used 30 training interactions (Epochs) to achieve around 97% accuracy, whereas the machine learning models utilized the specified algorithm(s) once for training and reached a performance accuracy of 100%. Obtaining 100% accuracy is atypical and not statistically likely. To see this for multiple models is even more abnormal. This abnormality may have been caused

because the machine learning models were trained using the artificially developed NSL-KDD dataset, or because the model was really well fit to the data, or it could be because the randomly selected data initialization index was at the perfect location to achieve 100% accuracy and 0% false positive rate. This abnormality is a topic for future research.

Hybrid Machine-Learning Model Results

When looking at the hybrid machine learning model performance results, as seen in Table 2, we see that combining Supervised and Unsupervised Machine-based Learning methods created a performance accuracy rate of 100% and false-negative and false-positive rates of 0%. This high level of accuracy is not statistically likely. The good results from this multi-layered hybrid model does not necessarily mean that combining multiple algorithms increases model performance accuracy, nor does it demonstrate that the combination of algorithms reduces false-positive and false-negative rates, however this hybrid model did have good performance results, so the combination of algorithms may have been a contributing facture. Another possible contributor to the good accuracy results for the hybrid model is likely from the pre-processing and good transformation and fitting of the model to the dataset. It is a skill to properly fit a model to a dataset, making sure not to over or underfit the data. The performance results for the hybrid IDS model demonstrate a proper fitting of the data.

Table 2

Hybrid Machine Learning Model Performance Results

Model	Accuracy (%)	False Positive Rate (%)	True Positive Rate (%)	Precision (%)	Recall (%)	F1-Score
Hybrid Machine Learning Model	100.00	0.00	100.00	100.00	100.00	1.00

Individual Component Analysis Results

The individual components of the hybrid model were also tested individually for performance comparison. The models underwent the same pre-processing of linear regression, and standardization and then were trained and tested. Each model's performance accuracy, precision, recall, false-positive, and true-positive rates were calculated and compared. As seen in the results comparison Table 3, the individual components performed as well as the hybrid IDS model, with all models performing at 100% and 0% false-positive rate. This is not typical and is most likely due to the artificial data of the NSL-KDD dataset and very good fitting of the models. Since the models' accuracy were hard for model comparison, the R^2 score was also calculated to estimate model strength. Comparing the R^2 score, we see that the Support Vector Machine (SVM) model had a higher explanatory score.

Table 3

Comparison of Hybrid IDS to the Individual Components

Model	Accuracy (%)	False Positive Rate (%)	True Positive Rate (%)	Precision (%)	Recall (%)	F1-Score	R^2 Score (%)
Hybrid Machine Learning Model	100.00	0.00	100.00	100.00	100.00	1.00	87.879
Principal Component Analysis – Logistical Regression	100.00	0.00	100.00	100.00	100.00	1.00	86.243
Support Vector Machine	100.00	0.00	100.00	100.00	100.00	1.00	99.998

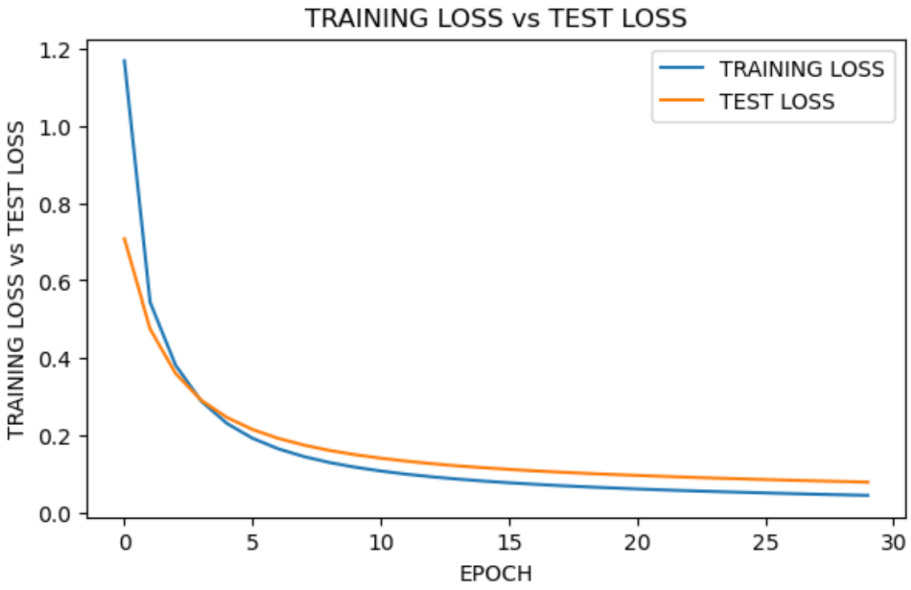
Artificial Neural Network Model #1

The deep learning models performed well, with an accuracy measure of around 97%. The results can be seen in figures 10 - 13. The first neural network sequential model utilized three layers and 256 nodes in the first layer. The activation algorithms were Rectified Linear Unit (relu) and softmax. Relu is used to transform and sum the weights of the input, and softmax is used to overcome oversaturation and vanishing gradient issues caused by other algorithms. The sparse_categorical_crossentropy loss function was used. This function is a lossy function that handles multi-class labels. Some data loss was anticipated. As seen in figure 10, the test loss (0.0789) was slightly higher than the training loss (0.0446). This is typical and was anticipated since the model was trained using the training data, while the test data was new to the model,

and it could not fully reproduce it. The test loss and training loss follow a similar slope showing that the model was well fit and closely represents the datasets.

Figure 10

ANN Model #1 – Training Loss to Test Loss Over Iteration

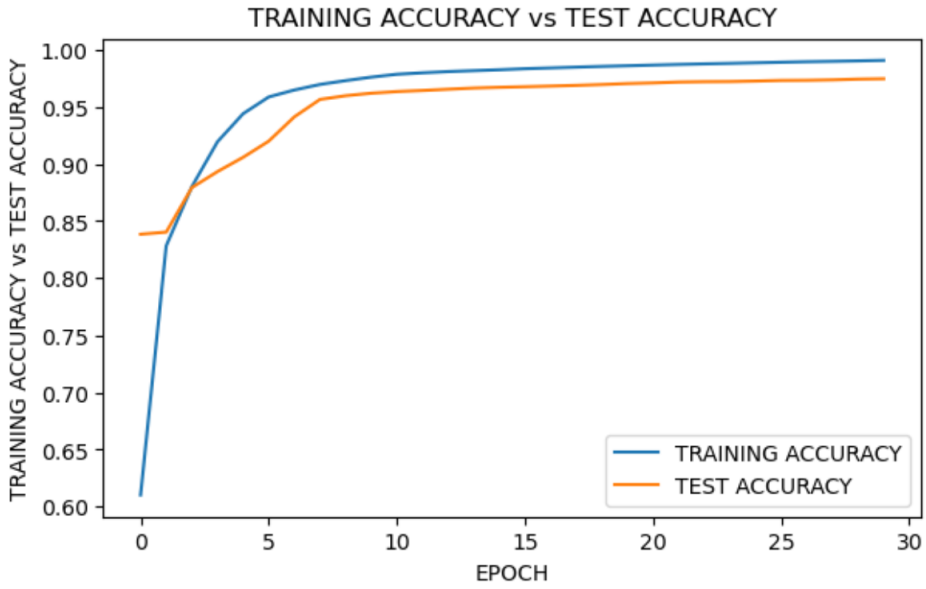


In figure 11 the training accuracy of model #1 shows a higher training accuracy than the test accuracy, 0.9911 compared to 0.9749. We expect training accuracy to be better than test accuracy, since the model has seen the training data before, during the training process. Another reason training accuracy could be higher than test accuracy, is because of a less complex test dataset. However, this is not the case with the NSL-KDD Test+ dataset. The testing dataset has more than 75% distinct records, from all attack types, and a similar percentage of normal and attack records to that of the training dataset. As observed in figure 11, the test accuracy jumps between the 2nd and 8th iterations (Epoch 2 - Epoch 8). This is most likely due to a well-fitted mini batch of

test data. The slope of the training accuracy and testing accuracy are similar in slope, demonstrating that the model is well trained and closely fitted to the data.

Figure 11

ANN Model #1 – Training Accuracy to Test Accuracy Over Iteration



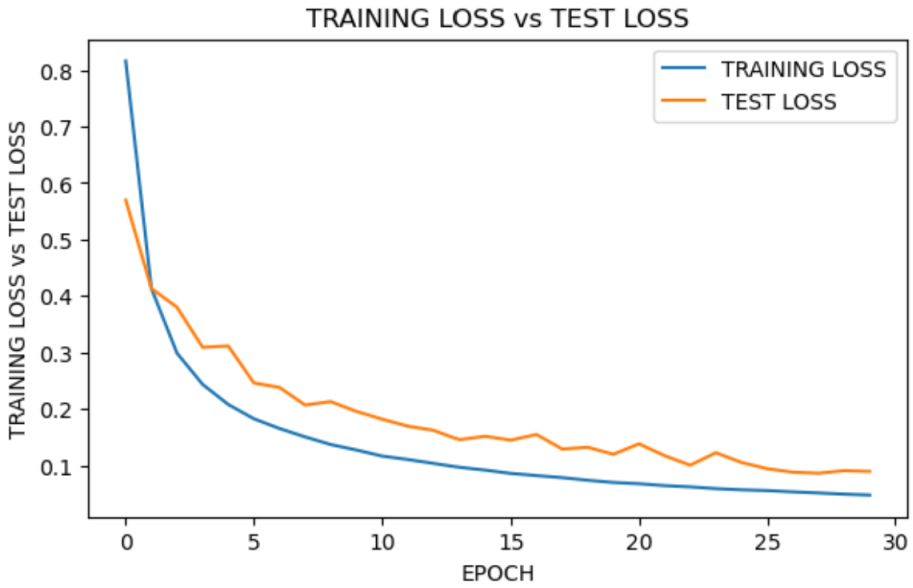
Artificial Neural Network Model #2

As with model #1, model #2, had some loss in both the training and test iterations, however in model #2 the test loss is closer to that of the training loss. Model #2 utilized five neural network layers, Adam (the updated Stochastic Gradient Descent optimizer (SGD)), and 30 training iterations. The sparse_categorical_crossentropy loss function was used, so some data loss was anticipated. As seen in figure 12, the test loss (0.0896) was higher than the training loss (0.0476). This is typical and was anticipated, since the model was trained on the training data, while the test data was

new to the model and could not be fully reproduced. The test loss and training loss follow a similar slope, showing that the models were closely related and well fit to the datasets. The choppiness of the test loss is due to the test data being tested in mini batches, whereas the training data test was performed using the entire dataset.

Figure 12

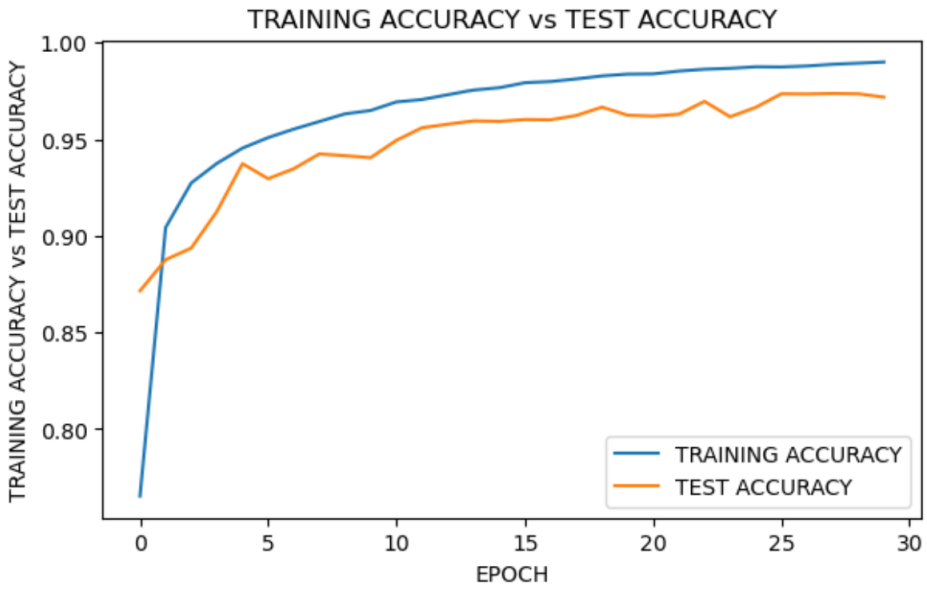
ANN Model #2 – Training Loss to Test Loss Over Iteration



In model #2 the training accuracy was slightly higher than the test accuracy, (0.9900) and (0.9718) respectively. This was anticipated because the model had seen the training data before. As seen in figure 13, the test accuracy hovers under the training accuracy, demonstrating good model fit, without overfitting or underfitting. The slope of the training accuracy and testing accuracy are similar in slope, demonstrating that the model closely represented the data.

Figure 13

ANN Model #2 – Training Accuracy to Test Accuracy Over Iteration



Note that by default every time Keras layers are initialized to run the neural network models, even after restarting the kernel, different random weights are assigned to different parameters, so results were inconsistent. To fix this issue and maintain consistent performance results, the random seed generators for numpy, python_random, and tensorflow were all set to a data starting index of zero.

The hybrid model outperformed both neural network intrusion detection models, as seen in the table 4 confusion matrix. The performance attack detection accuracy rate for the hybrid model was 100% whereas the Autoencoder Neural Network model #1 and model #2 reached 97.49% and 97.18% respectively. The high accuracy result of the hybrid model is most likely due to the linear regression and feature extraction during the pre-processing phase. The synthetic dataset was well fit to the model. The neural

network models both obtained a performance accuracy of around 97% for the test. The small accuracy differences between the hybrid and ANN models may be attributed to the random initialization indexes for the ANN being set to zero.

Table 4

Comparison of Hybrid IDS Accuracy to the Neural Network Models

Model	Training Accuracy (%)	Testing Accuracy (%)
Hybrid Machine Learning Model	100.00	100.00
Autoencoder Neural Network #1	99.11	97.49
Autoencoder Neural Network #2	99.00	97.18

Summary

The data and research results were presented and displayed in charts for easy performance comparisons of the five intrusion detection models. The statistical measurements used for analysis were calculated, displayed, and interpreted. It was determined that pre-processing had a great effect on model performance results and that added complexity and layers did not necessarily create more accurate prediction results.

Chapter V: Results, Conclusion, and Recommendations

Introduction

This section discusses the research study results and conclusions. The research questions are outlined and answered, and recommendations for future and continued research into the development of more accurate intrusion detection systems with lower false-positive rate using machine learning and deep learning techniques are given.

Results

This research analyzed the performance accuracy and false-positive rate of a hybrid multi-layered dynamic IDS, comparing the performance measures to the individual algorithm results, to determine the best algorithms to increase performance accuracy and reduce the false-positive rate. The hybrid IDS performance results were also compared to the performance of two Autoencoder Neuro Network Deep Learning-based IDSs. The results showed that model complexity, training iterations, and number of layers did not affect the model accuracy as much as pre-processing did. This research helped demonstrate the importance of pre-processing machine learning training data and the importance of having realistic, complete datasets to properly transform and fit the model. This research gives insight into the algorithms that provide data pre-processing; feature reduction / extraction, and data normalization, and gives future researchers methods of how to develop more accurate IDSs with a lower false-positive rate to detect and counter evolving attack types and increased attack surface area.

This research addressed three research questions:

1) *What are the effects of combining Supervised and Unsupervised Machine-based Learning methods on the accuracy, false-positive rate, and false-negative rate of a hybrid multi-layered intrusion detection system?*

The results from this research showed that combining Supervised and Unsupervised Machine-based Learning methods performed well with an accuracy = 100%, precision = 100%, recall = 100%, true-positive rate = 100%, false-negative rate = 0%, F1-Score = 1, and R^2 score = 87.879%. Exploration indicated a high level of accuracy but a lower explanatory level.

2) *How does the accuracy, false-positive rate, and false-negative rate of the hybrid multi-layered IDS compare to the performance of its individual components?*

When testing the hypothesis that the hybrid multi-layered IDS would outperform its individual components, the accuracy, false-positive rate, and false-negative rate of the hybrid multi-layered IDS were compared to the performance of the model's individual components, and the results showed that all machine learning models; the hybrid model, the Principal Component Analysis (PCA) model, and the Support Vector Machine (SVM) model had 100% accuracy, 0% false-positive rate, and 0% false-negative rate. These model performance accuracy results are not typical and are most likely due to the datasets being artificial and the models being well fitted to the dataset. As another measure of performance comparison, the explanatory strength of the models were calculated and compared. The R^2 score comparison found that the SVM

model independent variables were more strongly associated with the response variable (attack type - Class label). The R^2 score for the hybrid IDS model was 87.879%, the R^2 score for the PCA was 86.243%, and the R^2 score for the SVM was 99.998%. This research also showed that the combination of pre-processing algorithms for linear regression, feature reduction / extraction, and standardization impacted model performance more than specific machine learning algorithms. In fact, with the same pre-processed data, the machine learning models performed equally as well, with 100% accuracy.

3) *How does the hybrid multi-layered IDS's performance compare to Deep Learning-based IDSs?*

This study's results did not support the hypothesis that the deep learning-based IDSs would outperform the hybrid model. The proposed hybrid IDS model performance results were compared to the performance results of two different Deep Learning Autoencoder Neural Network IDS models and the results showed that the hybrid model outperformed the neural network models. Sapre et al., (2019) had similar findings for binary classification with the Support Vector Machine (SVM) and other machine learning algorithms outperforming the proposed deep learning-based IDS, in all categories of performance measurement (precision, recall, and F1-score). Just like the Sapre et al., (2019) study, the SVM model outperformed all other models based on performance accuracy and R^2. This demonstrates that increasing the complexity of a machine learning model by adding multiple algorithmic layers does not necessarily provide better

performance accuracy (Sapre et al., 2019). The hybrid model had an accuracy rate of 100% whereas the deep learning-based IDS test results had an accuracy of around 97%. The deep learning IDSs had variable prediction results. By default, when the Keras layers are initialized, different random weights are assigned to different parameters. Even after the kernel was restarted, the results were inconsistent, so to fix this issue and maintain consistent performance results, the random seed generators for numpy, python_random, and tensorflow were all set to a data starting index of zero. This did not provide the most accurate model results, but it gave a consistent baseline for comparison to the hybrid model's accuracy.

Conclusion

Advancements in technology have created additional network attack surface, allowed the development of new attack types, and increased the impact caused by an attack. Fueled by an increase in financial incentives to sell data, these new attacks threaten businesses, personal data, and privacy. Researchers agree that current intrusion detection systems are not able to adapt to detect these new attack forms, so alternative IDS methods have been proposed that incorporate machine learning and deep learning-based algorithms.

This research explored current related studies about intrusion detection systems and machine learning and proposed a new hybrid multi-layered dynamic machine learning-based IDS model consisting of the Principal Component Analysis (PCA) and Support Vector Machine (SVM) learning algorithms. The NSL-KDD dataset, benchmark dataset for IDSs, was used to compare the hybrid IDS model's performance accuracy

and false-positive rate to the results of the model's individual algorithmic components to determine which components most impact attack prediction performance and false positive rate.

The research results found that the hybrid model performed with an accuracy level of 100% and false positive rate of 0%. The Principal Component Analysis (PCA) and Support Vector Machine (SVM) models results also performed at 100% accuracy with a false positive rate of 0%. The performance metrics of the hybrid model were also compared to two Deep Learning Autoencoder Neuro Network (ANN) IDS models and the results found that the hybrid model outperformed the ANN models' performance accuracy of 97%. As seen in this research, performance comparison of the hybrid multi-layered intrusion detection system model and the model's individual components showed that the production of an accurate IDS with lower false-positive and false-negative rates is possible, and that the pre-processing phase greatly impacts the model's results.

The contribution of this research to machine-learning IDS modeling reinforces the value and importance of the pre-processing phase in identifying the key predictive features. The second contribution of this research is the demonstrated effectiveness of dimensionality reduction and feature extraction on model accuracy and false-positive rate across all model types.

Future Work

This research provides insight for future exploration into what creates accurate machine learning IDSs and lowers the false-positive rate. This research helps direct

future studies into the development of IDSs with a higher accuracy level and a lower false positive rate. Continued research into machine learning and deep learning-based IDSs is needed to protect networks, businesses, and data privacy against an increased attack surface area and continually evolving attack types.

Research model accuracy of 100% is not likely. Although other known previous research using the NSL-KDD Dataset did not have this statistical abnormality, other artificial datasets and datasets that include live network traffic could be used to evaluate if the models' performance were superficially inflated by the artificial NSL-KDD Dataset. Implementing the proposed hybrid IDS into a live network environment to test the performance accuracy would be another means of determining model attack predictability. Additionally, research into the usefulness of the algorithms in the pre-processing phase and the combination of pre-processing algorithms to machine learning-based testing algorithms could be performed.

As technology continues to improve and attacks continue to evolve, intrusion detection systems are one of the best layers of defense for network security. With the help of Machine Learning, researchers can continue to develop new methods of achieving efficient dynamic intrusion detection systems to protect against continually evolving network attacks.

References

Al, S., & Dener, M. (2021). STL-HDL: A new hybrid network intrusion detection system for imbalanced dataset on big data environment. *Computers & Security,* 110, 102435–. https://doi.org/10.1016/j.cose.2021.102435

Albahar, M. A., & Binsawad, M. (2020). Deep Autoencoders and Feedforward Networks Based on a New Regularization for Anomaly Detection. *Security and Communication Networks*, 1–9. https://doi.org/10.1155/2020/7086367

Al Jallad, K., Aljnidi, M. & Desouki, M.S. (2020). Anomaly detection optimization using big data and deep learning to reduce false-positive. *J Big Data, 7*, 68. https://doi.org/10.1186/s40537-020-00346-1

Allianz (2022). *Allianz Risk Barometer 2022*. https://www.allianz.com/content/dam/onemarketing/azcom/Allianz_com/press/document/Allianz_Risk_Barometer_2022_FINAL.pdf

Almasoudy, F.H., Al-Yaseen, W.L., & Idrees, A.K. (2020). Differential Evolution Wrapper Feature Selection for Intrusion Detection System. *Procedia Computer Science, 10*(438), 1230-1239. https://doi.org/10.1016/j.procs.2020.03.438

Al-Yaseen, W. L., Idrees, A. K., & Almasoudy, F. H. (2022). Wrapper feature selection method based differential evolution and extreme learning machine for intrusion detection system. *Pattern Recognition, 132*, 108912–. https://doi.org/10.1016/j.patcog.2022.108912

Brooks, C. (2022). Alarming cyber statistics for mid-year 2022 that you need to know. *Forbes*. https://www.forbes.com/sites/chuckbrooks/2022/06/03/alarming-

cyber-statistics-for-mid-year-2022-that-you-need-to know/?sh=7b690 a867864

Cisco (2020). *Cisco annual internet report* (2018-2023). Cisco. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf

Cisco (2022). *SNORT*. https://www.snort.org/

Ferreira, A. (2022). *VS Code for data science.* Towards Data Science. https://towardsdatascience.com/vs-code-for-data-science-aee82fe08bac

Halbouni, A., Gunawan, T.S., Habaebi, M.H., & Kartiwi, M. (2022). Machine learning and deep learning approaches for cybersecurity: a review. *IEEEAccess*, *10*(3151248), 19572-19585. https://doi.org/10.1109/ACCESS.2022.3151248

Henry, A., Gautam, S., Khanna, S., Rabie, K., Shongwe, T., Bhattacharya, P., Sharma, B., & Chowdhury, S. (2023). Composition of Hybrid Deep Learning Model and Feature Optimization for Intrusion Detection System. *Sensors* (Basel, Switzerland), *23*(2), 890-. https://doi.org/10.3390/s23020890

Kanna, P.R., & Santhi, P. (2022). Hybrid Intrusion Detection using MapReduce based Black Widow Optimized Convolutional Long Short-Term Memory Neural Networks. *Expert Systems with Applications*, *194*, 116545–. https://doi.org/10.1016/j.eswa.2022.116545

Kaur, S., & Singh, M. (2020). Hybrid intrusion detection and signature generation using Deep Recurrent Neural Networks. *Neural Computing & Applications, 32*(12), 7859–7877. https://doi.org/10.1007/s00521-019-04187-9

Lamrini, B., Gjini, A., Daudin, S., Armando, F., Pratmarty, P., & Travé-Massuyés, L. (2018). Anomaly Detection Using Similarity-based One-Class SVM for Network Traffic Characterization. *ResearchGate*. https://www.researchgate.net/publication/331257012_Anomaly_Detection_Using_Similarity-based_One-Class_SVM_for_Network_Traffic_Characterization

Latah, M. & Toker, l. (2018). Towards an efficient anomaly-based intrusion detection for software-defined networks. *IET Networks: The Institution of Engineering and Technology*, *7*(6), 453-459. https://doi.org/10.1049/iet-net.2018.5080

Larose, D.T., & Larose, C.D. (2015). *Data Mining and Predictive Analytics* (second edition). Wiley. pp. 20-45, 92-108

Liu, X., Li, T., Zhang, R., Wu, D., Liu, Y., & Yang, Z. (2021). A GAN and Feature Selection-Based Oversampling Technique for Intrusion Detection. *Security and Communication Networks,* 1–15. https://doi.org/10.1155/2021/9947059

Maimo, L.F., Gomez, A.L.P., Clemente, F.G., Perez, M.G., & Perez, G.M. (2018). A self-adaptive deep learning-based system for anomaly detection in 5G networks. *IEEEAccess*, *10*(2803446), 7700-7712. https://doi.org/10.1109/ACCESS.2018.2803446

Mari, A.-G., Zinca, D., & Dobrota, V. (2023). Development of a Machine-Learning Intrusion Detection System and Testing of Its Performance Using a Generative Adversarial Network. *Sensors* (Basel, Switzerland), *23*(3), 1315–. https://doi.org/10.3390/s23031315

Meemongkolkiat, N., & Suttichaya, V. (2021). Analysis on network traffic features for

designing machine learning based IDS. *Journal of Physics: Conference Series, 1993*(012029). https://doi:10.1088/1742-6596/1993/1/012029

Mijalkovic J., & Spognardi A. (2022). Reducing the False Negative Rate in Deep Learning Network Intrusion Detection Systems. *Algorithms, 15*, 258. https://doi.org/10.3390/a15080258

Mohd, N., Singh, A., & Bhadauria, H. S. (2021). Intrusion Detection System Based on Hybrid Hierarchical Classifiers. *Wireless Personal Communications*, *121*(1), 659–686. https://doi.org/10.1007/s11277-021-08655-1

Morgan, S. (2020). Cybercrime to cost the world $10.5 trillion annually by 2025. *Cybersecurity Magazine.* https://cybersecurityventures.com/cybercrime-will-cost-the-world-16-4-billion-a-day-in-2021/

Morgan, S. (2022). 2022 Cybersecurity almanac: 100 facts, figures, predictions and statistics. *Cybersecurity Magazine*. https://cybersecurityventures.com/cybersecurity-almanac-2022/

Mushtaq, E., Zameer, A., & Nasir, R. (2023). Knacks of a hybrid anomaly detection model using deep auto-encoder driven gated recurrent unit. *Computer Networks* (Amsterdam, Netherlands: 1999), 226. https://doi.org/10.1016/j.comnet.2023.109681

Neuralink (2022). *Breakthrough Technology for the Brain*. Neuralink. https://neuralink.com/

Oluoha, O.U., Yange, T.S., & Okereke G.E. (2021). Cutting Edge Trends in Deception Based Intrusion Detection Systems – A Survey. *Journal of Information Security*,

12, 250-269. https://doi.org/10.4236/jis.2021.124014

Patil, R., Dudeja, H., & Modi, C. (2019). Designing an efficient security framework for detecting intrusions in virtual network of cloud computing. *Computers & Security*, *85*, 402–422. https://doi.org/10.1016/j.cose.2019.05.016

Pingale, S. V., & Sutar, S. R. (2022). Remora whale optimization-based hybrid deep learning for network intrusion detection using CNN features. *Expert Systems with Applications*, *210*, 118476–. https://doi.org/10.1016/j.eswa.2022.118476

Pruciak, M. (2021). Why is Python a Great Choice for Data Analysis? *Ideamotive*. https://www.ideamotive.co/blog/python-for-data-analysis

Rao, N.K., Rao, V.K., & P.V.G.D., P. R. (2021). A hybrid Intrusion Detection System based on Sparse autoencoder and Deep Neural Network. *Computer Communications, 180*, 77–88. https://doi.org/10.1016/j.comcom.2021.08.026

Riera, T.S., Higuera, J.B., Higuera, J.B.2, Herraiz, J.M., & Montalvo, J.S. (2022). A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques. *Computers and Security*, *20*(1016). https://doi.org/10.1016/j.cose. 2022.102788

Sapre, S., Ahmadi, P., & Islam, K. (2019). A Robust Comparison of the KDDCup99 and NSL-KDD IoT Network Intrusion Detection Datasets through Various Machine Learning Algorithms. *ArXiv*: 1912.13204. https://doi.org/10.48550/arXiv. 1912.13204

Sarker, I. H. (2022). Machine learning for intelligent data analysis and automation in cybersecurity: Current and future prospects. *Annals of Data Science, 2*(1007). https://doi.org/10.1007/s40745-022-00444-2

Sornsuwit, P., & Jaiyen, S. (2019). A new hybrid machine learning for cybersecurity threat detection based on adaptive boosting. *Applied Artificial Intelligence: An International Journal, 33*(5), 462-282. https://www.tandfonline.com/loi/uaai20

Stack Overflow (2021). *2021 Developer Survey*. Stack Overflow. https://insights.stackoverflow.com/survey/2021

Stack Overflow (2022). *2022 Developer Survey.* Stack Overflow. https://survey.stackoverflow.co/2022/

Sun, P., Liu, P., Li, Q., Liu, C., Lu, X., Hao, R., & Chen, J. (2020). DL-IDS: Extracting Features Using CNN-LSTM Hybrid Network for Intrusion Detection System. *Security and Communication Networks*, 1–11. https://doi.org/10.1155/2020/8890306

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. (2009). A Detailed Analysis of the KDD CUP 99 Data Set. *IEEE Symposium on Computational Intelligence for Security and Defense Applications,* 1-6. https://DOI: 10.1109/CISDA.2009.5356528

Terra, J. (2023). Why Python Is Essential for Data Analysis and Data Science. *Simplilearn.* https://www.simplilearn.com/why-python-is-essential-for-data-analysis-article

UNB (n.d.). *NSL-KDD dataset.* Canadian Institute of Cybersecurity. https://www.unb.ca/cic/datasets/nsl.html

Verma, I. (2021). Introduction to machine learning with Jupyter notebooks. *Red Hat, Inc*. https://developers.redhat.com/articles/2021/05/21/introduction-machine-learning-jupyter-notebooks#what_is_a_jupyter_notebook_

Visual Studio Code (2023). *Data Science in Visual Studio Code.* Microsoft. https://code.visualstudio.com/docs/datascience/overview

Yange, T.S., Onyekware, O., & Abdulmuminu, M.Y. (2020). A data analytics system for network intrusion detection using decision tree. *Journal of Computer Sciences and Applications, 8*(1), 21-29. https://DOI:10.12691/jcsa-8-1-4

Appendix A: NSL-KDD Dataset Features

Feature #	Feature Name	Feature #	Feature Name	Feature #	Feature Name
1	Duration	15	Su attempted	29	Same srv rate
2	Protocol type	16	Num root	30	Diff srv rate
3	Service	17	Num file creations	31	Srv diff host rate
4	Flag	18	Num shells	32	Dst host count
5	Source bytes	19	Num access files	33	Dst host srv count
6	Destination bytes	20	Num outbound cmds	34	Dst host same srv rate
7	Land	21	Is host login	35	Dst host diff srv rate
8	Wrong fragment	22	Is guest login	36	Dst host same src port rate
9	Urgent	23	Count	37	Dst host srv diff host rate
10	Hot	24	Srv count	38	Dst host serror rate
11	Number failed logins	25	Serror rate	39	Dst host srv serror rate
12	Logged in	26	Srv serror rate	40	Dst host rerror rate
13	Num compromised	27	Rerror rate	41	Dst host srv rerror rate
14	Root shell	28	Srv rerror rate	42	Class label

Appendix B: Dummy Variable Breakdown

NSL-KDD Train Dataset Categorical to Numerical Dummy Variables

- **Feature #2 Protocol Type** – TCP -> 1 (102689 records), UDP -> 2 (14993), ICMP -> 3 (8291)

- **Feature #3 Service** – ecri -> 1 (3077), private -> 2 (21853), http -> 3 (40871), all other -> 0 --- ftp_data (6860), other (4359), remote (78), 0_job (78), name (911), netbios_ns (347), eco_i (4586), tp (77520), mtp (3077), telnet (2353), finger (1767), domain_u (9043), supdup (544), uucp_path (689), Z39_50 (862), s0 (42267), csnet_ns (545), uucp (780), netbios_dg (405), 0m (405), urp_i (602), auth (955), domain (569), ftp (1765), bgp (710), ldap (410), gopher (518), vmnet (617), systat (477), 3_443 (530), efs (485), Whois (693), Imap4 (647), iso_tsap (687), echo (434), klogin (433), link (475), sunrpc (381), login (429), kshell (299), sql_net (245), time (1175), host0s (460), exec (474), ntp_u (168), discard (538), nntp (2960), courier (734), ctf (563), ssh (311), day0 (521), shell (65), netstat (360), pop_3 (264), nnsp (630), IRC (187), Pop_2 (78), Printer (69), tim_i (8), pm_dump (5), red_i (8), netbios_ssn (362), rje (86), X11 (73), 3_8001 (2), 3_2784 (1), aol (2), harvest – 0 (2), t0_u (3), urh_i (10)

- **Feature #4 Flag** – SF –> 1 (74945)
  all other -> 0 --- RSTO (1665), REJ (11233), S3 (49), RSTR (2421), S2 (127), S1 (365), SH (271), OTH (46)

- **Feature #41 Class label** – Normal -> 0 (67343), DoS -> 1 (45928) (apache2 (0), back (956), land (19), Neptune (41214), mailbomb (0), pod (201), processtable (0), smurf (2646), teardrop (892), udpstorm (0), worm (0)), Probe -> 2 (11656) (ipsweep (3599), mscan (0), nmap (1493), portsweep (2931), saint (0), satan (3633), U2R -> 3 (52) (buffer_overflow (30), loadmodule (9), perl (3), ps (0), rootkit (10), sqlattack (0), xterm (0)), R2L -> 4 (995) (ftp_write (0_write) (8), guess_passwd (53), httptunnel (0), imap (11), multihop (7), named (0), phf (4), sendmail (0), Snmpgetattack (0), spy (2), snmpguess (0), warezclient (890), warezmaster (20), xlock (0), xsnoop (0))

NSL-KDD Test Dataset Categorical to Numerical Dummy Variables

- **Feature #2 Protocol Type** – TCP -> 1 (18880 records), UDP -> 2 (2623), ICMP -> 3 (1043)

- **Feature #3 Service** – ecri -> 1 (752), private -> 2 (4772), http -> 3 (7889),
  all other -> 0 --- ftp_data (851), eco_i (262), telnet (1626), smtp (934), ftp (693), ldap (19), pop_3 (1019), courier (40), discard (26), imap4 (306), domain_u (894), mtp (32), systat (32), iso_tsap (48), other (838), csnet_ns (34), finger (136), uucp (96), whois (40), netbios_ns (36), link (41), Z39_50 (45), sunrpc (159), auth (67), netbios_dgm (25), 0_path (46), vmnet (43), domain (51), name (60), pop_2 (13), 3_443 (36), urp_i (23), login (50), gopher (34), exec (27), time (64), remote_job (14), ssh (26), kshell (24), sql_net (18), shell (16), host)s (23), echo (37), day0 (28), pm_dump (16), IRC (13), netstat (26), ctf (41), nntp (21), netbios_ssn (15), tim_i (6), supdup (27), bgp (46), nnsp (42), efs (33), k0 (21), ntp_u (10), printer (11), rje (8), t0_u (1), X11 (15)

- **Feature #4 Flag** – SF -> 1 (14875)
  all other -> 0 --- RSTO (775), S0 (2015), REJ (3850), S3 (249), RSTR (669), S2 (15), S1 (21), SH (73), OTH (4)

- **Feature #41 Class label** – Normal -> 0 (9711), DoS -> 1 (7460/7458) (apache2 (737), back (359), land (7), Neptune (4657), mailbomb (293), pod (41), processtable (685), smurf (665), teardrop (12), udpstorm (2), worm (2)), Probe -> 2 (2421) (ipsweep (141), mscan (996), nmap (73), portsweep (157), saint (319), satan (735)), U2R -> 3 (210/200) (buffer_overflow (20), loadmodule (2), perl (2), ps (158), rootkit (13), sqlattack (2), xterm (13)), R2L -> 4 (3191/2654) (ftp_write (0_write) (3), guess_passwd (1231), httptunnel (133), imap (307), multihop (18), named (17), phf (2), sendmail (14), Snmpgetattack (178), spy (0), snmpguess (331), warezclient (0), warezmaster (944), xlock (9), xsnoop (4))

Appendix C: Principal Component Analysis (PCA) Code

```
import pandas as pd #import panda for data manipulation
import numpy as np #import numpy for graphing data
import matplotlib.pyplot as plt
import pip #import pipping to see data
from sklearn.preprocessing import StandardScaler #import standarscaler for
standrdization
from sklearn.decomposition import PCA #import principal component analysis algorithm
for dimentionality reduction
from sklearn.linear_model import LogisticRegression #import logistic regression using
which uses log natural because the world is not linear
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
pip.main(["install", "openpyxl"]) #allows excel files to be read
import warnings #import warning notifications
warnings.filterwarnings("ignore") #ignore warnings

Train = pd.read_excel('KDDTrainSet.xlsx') #Read excel Trainset file
Test = pd.read_excel('KDDTestSet.xlsx') #Read excel Testset file

Train_X = Train.iloc[:,:41].values #indexes columns 0-40 (explanitory/indepenent
variables)
Train_y = Train.iloc[:,-41].values #indexes columns 41 only (response/dependent
variable)
Train_X.shape #writes out shape of data frame
Train_y.shape

Test_X = Test.iloc[:,:41].values #creates dataset with all but class label (attack type)
Test_y = Test.iloc[:,-41].values #creates dataset with only class label (attack type)
Test_X.shape #shows the shape of Test_X dataset

#(Train_X) #prints array of Train_X
#(Train_y) #prints array of Train_y
#Train.info() #Displays data type information
#Train #prints DataFrame

scaler=StandardScaler() #standardized the independent variables by subtracting the
mean and dividing by the standard deviation
XTrain_scaler = scaler.fit_transform(Train_X) #fit and transform Train_X dataset -
normalizing
XTest_scaler = scaler.transform(Test_X) #transform Test_X dataset - normalizing
XTrain_scaler #prints the XTrain scaled (normalized) dataset
```

```
pca = PCA(n_components=2) #applying PCA with top 2 principal components analysis
XTrain_pca = pca.fit_transform(XTrain_scaler) # fits and transforms scaled
XTrain_scaled dataset using the principal component analysis algorithm
XTest_pca = pca.transform(XTest_scaler) # transforms scaled XTest_scaled dataset
using the principal component analysis algorithm

XTrain_pca.shape #print shape of the XTrain_pca dataset
#XTest_pca.shape #print shape of the XTest_pca dataset

explained_variance = pca.explained_variance_ratio_ #determines how spaced apart the
components are
print(pca.explained_variance_ratio_) #shows how spaced apart the components are
from each other

Logclassifier = LogisticRegression(random_state = 0) #shows how related our
independent variables are to our dependent variable starting from first record
Logclassifier.fit(XTrain_scaler, Train_y) #fits the XTrain_scaler data

y_pred = Logclassifier.predict(XTest_scaler) #uses logarithmic classifier (natural log) to
predict the results of the XTest_scaler dataset
print (y_pred) #shows the results of the logarithmic classifer on the XTest_scaler
dataset

from sklearn.metrics import confusion_matrix #imports a confusion martix for graphing
results
cm = confusion_matrix(Test_y, y_pred) #tells what features to graph in a confusion
matrix
print (cm)
from sklearn.metrics import accuracy_score #import the accuracy package
clf = PCA() #prepares variable for training
clf.fit(XTrain_scaler, Train_y) #trains Support Vector Machine model
print ('score', clf.score(XTrain_scaler, Train_y)) #printed the R^2 score of accuracy of
the test data
print('Accuracy', accuracy_score(Test_y, y_pred)) #printed the accuracy of the test data

FP = cm.sum(axis=0) - np.diag(cm)
FN = cm.sum(axis=1) - np.diag(cm)
TP = np.diag(cm)
TN = cm.sum() - (FP + FN + TP)

FP = FP.astype(float)
FN = FN.astype(float)
TP = TP.astype(float)
```

```
TN = TN.astype(float)

print (FP)
print (FN)
print (TP)
print (TN)

# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
print (TPR)

# Precision or positive predictive value
PPV = TP/(TP+FP)
print (PPV)

#F1-score
F1 = 2 * (PPV * TPR) / (PPV + TPR)
print (F1)

# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
print (ACC)
from sklearn.metrics import classification_report
print (classification_report(Test_y, y_pred))
#cm.print_stats()

FP = np.logical_and(Test_y != y_pred, y_pred != -1).sum()
FN = np.logical_and(Test_y != y_pred, y_pred == -1).sum()
TP = np.logical_and(Test_y == y_pred, Test_y != -1).sum()
TN = np.logical_and(Test_y == y_pred, Test_y == -1).sum()
TPR = 1. * TP / (TP + FN)
FPR = 1. * FP / (FP + TN)
print (TPR)
print (FPR)
```

Appendix D: Support Vector Machine (SVM) Code

```
import pandas as pd #import panda for data manipulation
import numpy as np #import numpy for graphing data
import matplotlib.pyplot as plt
import pip #import pipping to see data
from sklearn.preprocessing import StandardScaler #import standarscaler for
standrdization
from sklearn.svm import SVC #imports support vector machine algorithm for
classification, regression, and reduce outliers
#SVM is a cluster classification used to estimate where a random observation would lay
based on preobserved observations
from sklearn.linear_model import LogisticRegression #import logistic regression using
which uses log natural because the world is not linear
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
pip.main(["install", "openpyxl"]) #allows excel files to be read
import warnings #import warning notifications
warnings.filterwarnings("ignore") #ignore warnings

Train = pd.read_excel('KDDTrainSet.xlsx') #Read excel Trainset file
Test = pd.read_excel('KDDTestSet.xlsx') #Read excel Testset file

Train_X = Train.iloc[:,:41].values #indexes columns 0-40 (explanitory/indepenent
variables)
Train_y = Train.iloc[:,-41].values #indexes columns 41 only (response/dependent
variable)
Train_X.shape #writes out shape of data frame

Test_X = Test.iloc[:,:41].values #creates dataset with all but class label (attack type)
Test_y = Test.iloc[:,-41].values #creates dataset with only class label (attack type)
Test_X.shape #shows the shape of Test_X dataset

#(Train_X) #prints array of Train_X
#(Train_y) #prints array of Train_y
#Train.info() #Displays data type information
#Train #prints DataFrame

scaler=StandardScaler() #standardized the independent variables by subtracting the
mean and dividing by the standard deviation
XTrain_scaler = scaler.fit_transform(Train_X) #fit and transform Train_X dataset -
normalizing
XTest_scaler = scaler.transform(Test_X) #transform Test_X dataset - normalizing
XTrain_scaler #prints the XTrain scaled (normalized) dataset
```

```
Logclassifier = LogisticRegression(random_state = 0) #shows how related our independent variables are to our dependent variable
Logclassifier.fit(XTrain_scaler, Train_y) #fits the XTrain_scaler data

y_pred = Logclassifier.predict(XTest_scaler) #uses logarithmic classifier (natural log) to predict the results of the XTest_scaler dataset
print (y_pred) #shows the results of the logarithmic classifer on the XTest_scaler dataset

from sklearn.metrics import confusion_matrix #imports a confusion martix for graphing results
cm = confusion_matrix(Test_y, y_pred) #tells what features to graph in a confusion matrix

clf = SVC() #prepares variable for training
clf.fit(XTrain_scaler, Train_y) #trains Support Vector Machine model

from sklearn.metrics import accuracy_score #import the accuracy package
print('Accuracy', accuracy_score(Test_y, y_pred)) #printed the accuracy of the test data

print ('score', clf.score(XTrain_scaler, Train_y)) #got a score of the training data
print ('predicted', clf.predict(XTest_scaler)) #tested the SVM model with the test data

# false negative, true positive, true negative FP = cm.sum(axis=0) - np.diag(cm)
FN = cm.sum(axis=1) - np.diag(cm)
TP = np.diag(cm)
TN = cm.sum() - (FP + FN + TP)

FP = FP.astype(float)
FN = FN.astype(float)
TP = TP.astype(float)
TN = TN.astype(float)

print (FP)
print (FN)
print (TP)
print (TN)

# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
print (TPR)
```

```
# Precision or positive predictive value
PPV = TP/(TP+FP)
print (PPV)

#F1-score
F1 = 2 * (PPV * TPR) / (PPV + TPR)
print (F1)

# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
print (ACC)
# false positive/negative and true postive/negative
FP = np.logical_and(Test_y != y_pred, y_pred != -1).sum()
FN = np.logical_and(Test_y != y_pred, y_pred == -1).sum()
TP = np.logical_and(Test_y == y_pred, Test_y != -1).sum()
TN = np.logical_and(Test_y == y_pred, Test_y == -1).sum()
TPR = 1. * TP / (TP + FN)
FPR = 1. * FP / (FP + TN)
print (TPR)
print (FPR)
# import and print classification report
from sklearn.metrics import classification_report
print (classification_report(Test_y, y_pred))
```

Appendix E: Hybrid IDS Model Code

```
import pandas as pd #import panda for data manipulation
import numpy as np #import numpy for graphing data
import matplotlib.pyplot as plt
import pip #import pipping to see data
from sklearn.preprocessing import StandardScaler #import standarscaler for
standrdization
from sklearn.decomposition import PCA #import principal component analysis algorithm
for dimentionality reduction
from sklearn.svm import SVC #imports support vector machine algorithm for
classification, regression, and reduce outliers
#SVM is a cluster classification used to estimate where a random observation would lay
based on preobserved observations
from sklearn.linear_model import LogisticRegression #import logistic regression using
which uses log natural because the world is not linear
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
pip.main(["install", "openpyxl"]) #allows excel files to be read
import warnings #import warning notifications
warnings.filterwarnings("ignore") #ignore warnings

Train = pd.read_excel('KDDTrainSet.xlsx') #Read excel Trainset file
Test = pd.read_excel('KDDTestSet.xlsx') #Read excel Testset file

Train_X = Train.iloc[:,:41].values #indexes columns 0-40 (explanitory/indepenent
variables)
Train_y = Train.iloc[:,-41].values #indexes columns 41 only (response/dependent
variable)
Train_X.shape #writes out shape of data frame
Train_y.shape #shows shape of Train_y dataset

Test_X = Test.iloc[:,:41].values #creates dataset with all but class label (attack type)
Test_y = Test.iloc[:,-41].values #creates dataset with only class label (attack type)
Test_X.shape #shows the shape of Test_X dataset
#(Train_X) #prints array of Train_X
#(Train_y) #prints array of Train_y
#Train.info() #Displays data type information
#Train #prints DataFrame

scaler=StandardScaler() #standardized the independent variables by subtracting the
mean and dividing by the standard deviation
XTrain_scaler = scaler.fit_transform(Train_X) #fit and transform Train_X dataset -
normalizing
```

```
XTest_scaler = scaler.transform(Test_X) #transform Test_X dataset - normalizing
XTrain_scaler #prints the XTrain scaled (normalized) dataset

pca = PCA(n_components=2) #applying PCA with top 2 principal components analysis
XTrain_pca = pca.fit_transform(XTrain_scaler) # fits and transforms scaled
XTrain_scaled dataset using the principal component analysis algorithm
XTest_pca = pca.transform(XTest_scaler) # transforms scaled XTest_scaled dataset
using the principal component analysis algorithm
XTrain_pca.shape #print shape of the XTrain_pca dataset
#XTest_pca.shape #print shape of the XTest_pca dataset

explained_variance = pca.explained_variance_ratio_ #determines how spaced apart the
components are
print(pca.explained_variance_ratio_) #shows how spaced apart the components are
from each other

Logclassifier = LogisticRegression(random_state = 0) #shows how related our
independent variables are to our dependent variable starting from first record
Logclassifier.fit(XTrain_scaler, Train_y) #fits the XTrain_scaler data

y_pred = Logclassifier.predict(XTest_scaler) #uses logarithmic classifier (natural log) to
predict the results of the XTest_scaler dataset
print (y_pred) #shows the results of the logarithmic classifer on the XTest_scaler
dataset

from sklearn.metrics import confusion_matrix #imports a confusion martix for graphing
results
cm = confusion_matrix(Test_y, y_pred) #tells what features to graph in a confusion
matrix
print (cm) #print confusion matrix

clf = SVC() #prepares variable for training
clf.fit(XTrain_pca, Train_y) #trains Support Vector Machine model

print ('score', clf.score(XTrain_pca, Train_y)) #got a score of the training data
print ('predicted', clf.predict(XTest_pca)) #tested the SVM model with the test data

from sklearn.metrics import accuracy_score #import the accuracy package
print('Accuracy', accuracy_score(Test_y, y_pred)) #printed the accuracy of the test data

# false positive/negative and true psoitive/negative
FP = cm.sum(axis=0) - np.diag(cm)
FN = cm.sum(axis=1) - np.diag(cm)
```

```
TP = np.diag(cm)
TN = cm.sum() - (FP + FN + TP)

FP = FP.astype(float)
FN = FN.astype(float)
TP = TP.astype(float)
TN = TN.astype(float)

print (FP)
print (FN)
print (TP)
print (TN)

# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
print (TPR)

# Precision or positive predictive value
PPV = TP/(TP+FP)
print (PPV)

# F1-score
F1 = 2 * (PPV * TPR) / (PPV + TPR)
print (F1)

# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
print (ACC)
from sklearn.metrics import classification_report
print (classification_report(Test_y, y_pred))
#cm.print_stats()
# calculated and prints true/false positive rates
FP = np.logical_and(Test_y != y_pred, y_pred != -1).sum()
FN = np.logical_and(Test_y != y_pred, y_pred == -1).sum()
TP = np.logical_and(Test_y == y_pred, Test_y != -1).sum()
TN = np.logical_and(Test_y == y_pred, Test_y == -1).sum()
TPR = 1. * TP / (TP + FN)
FPR = 1. * FP / (FP + TN)
print (TPR)
print (FPR)
```

Appendix F: Autoencoder Neural Network (ANN) Code

```
import pandas as pd #install pandas for data preprocessing
import numpy as np #install numpy with machine learning package
import sklearn # import sklearn with machine learning algorithms
from sklearn.utils import shuffle #import randomizer
from sklearn.metrics import * #import all performance metrics
from sklearn.preprocessing import StandardScaler #import StandardScaler for
normalization
from sklearn.preprocessing import LabelEncoder #import labelEncoder
import matplotlib.pyplot as plt #import matplot lib for graphing
import warnings #import warning notifications
import tensorflow as tf
from keras.preprocessing import sequence #import sequence layer algorithm
from keras import optimizers #import optimizers for model optimization
from keras.utils import np_utils #import numpy utilities
from keras.models import Sequential #import sequential layer algorithm
from keras.layers import Dense, Dropout, Activation, Embedding, SimpleRNN,
BatchNormalization #import neural network algorthims
from keras.models import model_from_json #import json modeling
from keras.models import load_model #loads saved model
warnings.filterwarnings("ignore") #ignore warnings
import matplotlib_inline #import matploylib inline writer
#%matplotlib inline #import matploylib inline writer

Train = pd.read_excel('KDDTrainSet.xlsx') #Read excel Trainset file
Test = pd.read_excel('KDDTestSet.xlsx') #Read excel Testset file
Train_X = Train.iloc[:,:41].values #indexes columns 0-40 (explanitory/indepenent
variables)
Train_y = Train.iloc[:,-41].values #indexes columns 41 only (response/dependent
variable)
Train_X.shape #writes out shape of data frame
Train_y.shape #writes out shape of data frame

Test_X = Test.iloc[:,:41].values #creates dataset with all but class label (attack type)
Test_y = Test.iloc[:,-41].values #creates dataset with only class label (attack type)
Test_X.shape #shows the shape of Test_X dataset

scaler=StandardScaler() #standardized the independent variables by subtracting the
mean and dividing by the standard deviation
XTrain_scaler = scaler.fit_transform(Train_X) #fit and transform Train_X dataset -
normalizing
XTest_scaler = scaler.transform(Test_X) #transform Test_X dataset - normalizing
```

```
XTrain_scaler #prints the XTrain scaled (normalized) dataset

#imports random package and sets a constant starting index
import random as python_random
np.random.seed(0)
python_random.seed(0)
tf.random.set_seed(0)

#uses keras's sequential API with 256 neurons in the first layer
#second parameter input_dim corresponds to the input features
#relu as activation function and softmax is for the multiple classes
model1 = Sequential()
model1.add(Dense(64, input_dim = 41, activation = "relu", kernel_initializer =
"lecun_normal"))
model1.add(Dense(128, activation = "relu"))
model1.add(Dense(5, activation = "softmax"))
model1.summary() #prints parameters for each layer

#Loss function tell us the amount of loss (loosy system)
#Optimizer minimizes the losses
#Metrics is the mode of evaluation for the model
#sparse_categorical_loss works with multi-class classifcation problems
#adam the updated (SGD) Stochastic Gradient Descent optimizer
optim = optimizers.SGD(lr = 0.0001)
model1.compile(loss = 'sparse_categorical_crossentropy', optimizer = optim, metrics =
['accuracy'])
#fits the model to our data using 30 training iterations
history = model1.fit(XTrain_scaler, Train_y,
        validation_data = (XTest_scaler, Test_y),
        batch_size = 32,
        epochs = 30)

#use matplitlib to plot the accuracy to loss data
plt.figure(figsize = (15, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label = "TRAINING LOSS")
plt.plot(history.history['val_loss'], label = "TEST LOSS")
plt.title("TRAINING LOSS vs TEST LOSS")
plt.xlabel("EPOCH")
plt.ylabel("TRAINING LOSS vs TEST LOSS")
plt.legend(loc = "best")

plt.subplot(1, 2, 2)
```

```
plt.plot(history.history['accuracy'], label = "TRAINING ACCURACY")
plt.plot(history.history['val_accuracy'], label = "TEST ACCURACY")
plt.title("TRAINING ACCURACY vs TEST ACCURACY")
plt.xlabel("EPOCH")
plt.ylabel("TRAINING ACCURACY vs TEST ACCURACY")
plt.legend(loc = "best")

#y_pred = model1.predict(XTrain_scaler)
#y_pred = np.argmax(y_pred, axis=1)
#y_true = np.argmax(Test_y, axis=0)
#predicted = np.argmax(model1.predict(XTest_scaler),axis=0)
#print(precision_score(Test_y, y_pred))
#print(recall_score(Test_y, y_pred))

#saves model1 as a json file
model_json = model1.to_json()
with open("model1.json", "w") as json_file:
    json_file.write(model_json)
model1.save_weights('model1_weights.h5')
print("Saved model to disk")

#loads model1
json_file = open("model1.json", "r")
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
loaded_model.load_weights("model1_weights.h5")
print("Loaded model from disk")

#defines model2 layers
model2 = Sequential()
model2.add(Dense(32, input_dim = 41, activation = "relu", kernel_initializer =
"lecun_normal"))
model2.add(BatchNormalization())
model2.add(Dropout(0.2))
model2.add(Dense(32, activation = "relu"))
model2.add(Dense(5, activation = "softmax"))
model2.summary() #Lists model2 details

#SGD updated version of Adam is used to optimize the model
optim2 = optimizers.Adam(lr = 0.0001)
model2.compile(loss = 'sparse_categorical_crossentropy', optimizer = optim, metrics =
['accuracy'])
```

```
#Trains the model and validates it
history2 = model2.fit(XTrain_scaler, Train_y,
        batch_size = 32,
        epochs = 30,
        validation_data = (XTest_scaler, Test_y))

#plots the loss to accuracy of model2
plt.figure(figsize = (15, 4))
plt.subplot(1, 2, 1)
plt.plot(history2.history['loss'], label = "TRAINING LOSS")
plt.plot(history2.history['val_loss'], label = "TEST LOSS")
plt.title("TRAINING LOSS vs TEST LOSS")
plt.xlabel("EPOCH")
plt.ylabel("TRAINING LOSS vs TEST LOSS")
plt.legend(loc = "best")

plt.subplot(1, 2, 2)
plt.plot(history2.history['accuracy'], label = "TRAINING ACCURACY")
plt.plot(history2.history['val_accuracy'], label = "TEST ACCURACY")
plt.title("TRAINING ACCURACY vs TEST ACCURACY")
plt.xlabel("EPOCH")
plt.ylabel("TRAINING ACCURACY vs TEST ACCURACY")
plt.legend(loc = "best")
```