

Chapter One

Introduction

1.1 Background to the Study

Network intrusion is an accumulation of approaches developed and modified by the science establishment that are entrusted with the reliable gathering of traces from a scene of a crime, their inspection, Evaluation, and description of discovered substantiation that could accurately determine malicious activity, their strategies, and true intentions while in an investigative process of a network intrusion, which almost always leads to criminal prosecution. Computer intrusions have already been simplified over time. Due to technological advances in many categories, each firm specializes in occurrences in different conditions, particularly regarding cloud network analysis, investigation, IoT investigation evidence, mobile crime scene investigation, memory ballistics, and information investigations¹.

Network analysis strategies are concerned with security issues in network systems, and they prevalently use logs and manage to capture packet data to identify invaders and malicious behavior. Cybersecurity has become increasingly sophisticated with the tremendous number of resources distributed and approaches being used. It is challenging to precisely define the variables that must be considered to predict whether an occurrence has happened in a system setting. Historically, network intrusion detection systems (NIDS) were utilized as a network, broadcasting irregular actions and notifying the appropriate establishments. Frequently used NIDSs are grounded on signs in the form of event designs, executable files, or variances in measures. They lately saw 8,801,171,594 data record breaches in May 2020^{2,3}. These events indicate that even minor breaks can be tremendously detrimental, necessitating the importance of a searching recognition classification in such an ominous condition. Because of the hastily

growing connectivity of possessions and devices, they may see a worse upsurge. To spot such procedures, concerns, and households must use the network to comprehend the precise movements that are producing breaks and, as a result, stop any further occurrences. Numerous original datasets used to create NIDS have since been shown to be inadequate for usage in real-world settings. Researchers at the Australian Centre for Cyber Security (ACCS) and others worldwide saw the need for a high-quality dataset that could be used effectively in real-time settings, so they set out to create one. Consequently, this study developed what is now recognized as the UNSW-NB15 Dataset. They plan to utilize this data to build a NIDS model that reliably and correctly predicts assaults and other types of abnormal conditions⁴.

Companies currently rely heavily on IDSs to protect themselves against cybercriminals. IDS detects and validates traffic from multiple sources to identify intrusions and certain other privacy violations. Unlike firewalls, which increase security over domains to prevent intrusions, IDS examines a prospective attack whenever it happens, sends an alert, and may discontinue the link. Intrusion Detection Systems typically fall into two categories: those that monitor networks and those that watch individual hosts (HIDS). The former employs a network-wide packet analyzer, while the latter uses a local agent to comb through function calls, system file changes, and logs. Network security has become a top priority with the explosion in IoT device usage. Conventional intrusion detection techniques have drawbacks when used in an IoT network because of constraints and complexities⁵.

Numerous academics have looked at critical standard datasets over the past decades, including the Defense Advanced Research Projects Agency dataset and the Knowledge Discovery and Data Mining (KDD'99) Tools Competing dataset. However, these data are not currently considered difficult since they do not encompass real-world assault scenarios. New files, such as the Next-Generation Intrusion Detection System Dataset (NGIDS-DS) and the Web

Conference 2019 Data, have emerged to address this void by providing novel and timely exploitation scenarios on which to estimate the efficacy of cutting-edge intrusion detection systems. The files include thousands of subsystem connection logs from a Linux server during regular operation and during an intrusion^{6,7}.

Traditional machine learning methods can notice harmful patterns in network sessions, which can be blocked. Support Vector Machines, classification of association rules, Artificial Neural Networks, decision trees, ensembles, and random forest classifiers are all examples of methodologies in the literature^{8,9}. Machine learning methods are functional and traditional to raw information illustration of network circulation, for example, system call classifications. As a result, a recent research focus has been on the strategy and application of efficient dimensionality reduction methods that try to transfer categorizations of system calls to an original form that machine learning algorithms can analyze^{10,11}.

Numerous machine learning (ML) models have been developed and Validated, with Feature Reduction (FR) approaches typically used to improve model performance. Yet, these algorithms are unreliable for use in production IoT networks despite promising outcomes for ML's recognition abilities utilizing a subset of Evaluation criteria. Recent work in this field has focused on improving advanced results on a particular set of data instead of drawing inferences from such an ML-based NIDS system. Therefore, a significant quantity of academic study has been compared to actual operations. This might be due to the considerable expense of mistakes in this field compared to others, but it could also be because these tactics are inaccurate. This is because they are often tested on a singular dataset, which may contain impractical features to acquire and keep in a real-time IoT network stream. Because machine learning is inherently flexible, its hyper-parameters are often tweakable for a given dataset¹².

This study proposes a novel integrative optimization technique based on an improved Genetic algorithm(I-GA) to fetch relevant information based on adaptive inertia weight and learning factors for a network framework with Locally Linear Embedding (LLE), SVM, and KNN classifiers results.

1.2 Statement of the Problem

Distinguishing malicious network activity from everyday traffic is arduous and inefficient. A researcher must sift through vast amounts of data to trace the steps of an intrusion on a data connection. Therefore, a system is needed to detect attacks and accurately depict real-time network activities. Intrusion detection systems (IDSs) are an essential component of a secure network. Owing to the enormous capacity of network data, one of the challenges with modern security systems is false alarm reports of network intrusion and intrusion detection accuracy¹³.

The selection of crossover and mutation procedures significantly impacts genetic algorithms' searchability, iteration efficiency, and precision. A common pitfall of traditional genetic algorithms is their propensity to become stuck at the local minimum. Therefore, a new genetic algorithm was proposed by starting with a basic genetic algorithm and adding the base concept of the orthogonal system design. It was implemented to the population's initial condition utilizing the multigenerational elite framework and the emergence of the iterative local search operator; both serve to evade getting fixed in a limited minimum and speed up the algorithm's convergence. As a result of a battery of numerical trials, it was shown that the new approach was indeed practical. In information security, complicated optimization tasks with high-dimensional and nonlinear effects are challenging to solve using conventional optimization techniques or algorithms of artificial intelligence. As a result, developing an efficient optimization strategy to address such issues is crucial^{14,15,16,17}.

1.3 Aim and Objectives of the Study

This study aims to develop an improved and hybridized optimization for classifications of intrusion detection in networks.

The specific objectives are to:

- i. Develop a machine learning model to fetch relevant information from a vast amount of the CICIDS2017 dataset.
- ii. classify the developed model above.
- iii. validate the performance of the developed model using accuracy, sensitivity, specificity, precision, recall, f1 score, and computational time.
- iv. compare the attained outcomes with existing works.

1.4 Research Questions

1. Why is it challenging to implement an efficient algorithm to develop an improved intrusion detection system?
2. How should threats be classified, and with what models?
3. How can promising intrusion detection models be Validated for best results?
4. How should the viability of an improved intrusion detection model be determined?

1.5 Justification of the Study

An intrusion Detection System (IDS) is a monitor that displays network system actions and detects malevolent acts. The rapid rise and widespread use of the internet create questions about how to protect and securely share digital information. This study's goal is to advance a new intrusion detection system that is both efficient and effective for tracking network infrastructure to detect anomalous network activity and misuse by proposing development technologies and highlighting the limitations and criteria for achieving promising intrusion

detection systems. Utilizing machine learning in security aims to address problems in the field and to study current experiences with potential technologies.

1.6 Scope of the Study

This study suggests a machine learning-based investigative process for detecting and responding to cyber risks in forensic networks. This is accomplished by gathering data from various devices and networks, which is then examined for security flaws. It's usually used to track and investigate user and device behavior, audit system settings and faults, test the integrity of any sensitive system and data files, do predictive analysis of activity trends based on matching documented vulnerabilities, and audit operating systems.

1.7 Limitations of the Study

A secondary dataset was used for this research; other researchers can use Real-time data whenever available. Secondly, more combinations of Algorithms can be used to train and test the Machine to have more robust comparisons.

1.8 Operational Definition of Terms

Algorithm: An algorithm is a procedure for solving a problem or performing a computation. Algorithms act as an exact list of instructions that conduct specified actions step by step in hardware- or software-based routines.

Intrusion Detection: An act or a process of monitoring a network for malicious activity or policy violations.

Intrusion Detection System: This system monitors network traffic for suspicious activity and alerts when such action is discovered.

Dataset: A collection of related, discrete items of associated data that may be accessed individually, in combination, or as a whole entity.

Artificial Intelligence: This is the simulation of human intelligence processes by machines, especially Computer Systems.

Machine-Learning: The subset of Artificial Intelligence focuses on building systems that learn or improve performance based on the data they consume.

Dimensionality Reduction: is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension. Working in high-dimensional areas can be undesirable for many reasons; raw data are often sparse due to the curse of dimensionality, and analyzing the data is usually computationally intractable (hard to control or deal with). Dimensionality reduction is expected in fields dealing with large numbers of observations and variables, such as signal processing, speech recognition, neuroinformatics, and bioinformatics.

Feature Selection: is a process of automatically or manually selecting the subset of the most appropriate and relevant features to be used in model building. Feature selection is performed by including the essential features or excluding irrelevant ones in the dataset without changing them.

Feature Extraction refers to transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.

Endnotes

- ¹ J. Kallipan, K. Srinivasan, M. Quisar, S. Karpagam, C. Chuan-Yu, & C. Suganthan. "Performance Evaluation of Egression Models for the Prediction of the COVID-19 Reproduction Rate". **Frontiers in Public Health**, 9, 2021, 729795.
- ² Q. Sirajuddin, L. Jianqiang, A. Faheem, T. Saima, H.K Zahid, & W. Ahsan, "Analysis of Challenges in Modern Network Forensic Framework," **Security and Communication Networks**: 2021, 1–13.
- ³ J-P. Yaacoub, N. Hassan, S. Ola, & C. Ali. "Security Analysis of Drones Systems: Attacks, Limitations, and Recommendations." **Internet of Things** 11: 2020, 100218.
- ⁴ A. Kakareka. "Detecting System Intrusions." In Computer and Information Security Handbook, Morgan Kaufmann, 2013, pp. 47-62
- ⁵ S. Kumar, V.K. Sunanda, & A. Sakshi. "A Statistical Analysis on KDD Cup '99 Dataset for the Network Intrusion Detection System." **Applied Soft Computing and Communication Networks: Proceedings of ACN**: 2020, 131–157.
- ⁶ A. Khraisat, G. Iqbal, V. Peter, & K. Joarder. "Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges." **Cybersecurity** 2: 2019, 1-22.
- ⁷ Y. Ren, Z. Le, & N.S. Ponnuthurai. "Ensemble Classification and Regression-Recent Developments, Applications and Future Directions." **IEEE Computational Intelligence Magazine** 11, 2016, 41-53.
- ⁸ R. Guidotti, M. Anna, R. Salvatore, T. Franco, G. Fosca, & P. Dino P. "A Survey of Methods for Explaining Black Box Models." **ACM Computing Surveys (CSUR)** 51, 2018, 1-42.
- ⁹ M.S Mahdavinejad, R. Mohammadreza, B. Mohammadamin, A. Peyman, B. Payam, & P. S. Amit. "Machine learning for Internet of Things data analysis: A survey." **Digital Communications and Networks** 4, 2018, no. 3 161-175,
- ¹⁰ J. Schmidt, M.R.G Marques, B. Silvana, & M.A.L Marques. "Recent Advances and Applications of Machine Learning in Solid-State Materials Science." **npj Computational Materials** 5, 2019, no. 1: 83.
- ¹¹ M.N Mahdi, H.M.Z. Mohd, R.A. Abdul, I. Roslan, Y. Yunus, K.C. Lim, S.B.M.A. Muhammad, N. Hayder, & H.N. Hushalini. "Software Project Management using Machine Learning Technique—A Review." **Applied Sciences** 11, 2021, 5183.

¹² Y. Zhu, S.G. Gurjot, M.A. Fahad, A. Roobaea, & M. Mehedi. "Application of Data Mining Technology in Detecting Network Intrusion and Security Maintenance." **Journal of Intelligent Systems** 30, 2021, no. 1 664-676.

¹³ D. Yang, Y. Zeyu, Y. Hongqiang, & C. Yanrong. "An Improved Genetic Algorithm and its Application in Neural Network Adversarial Attack." **Plos one** 17, 2022, no. 5 e0267970.

¹⁴ S.Cheng, L. Hui, L. Xiujuan, & S. Yuhui. "A Quarter Century of Particle Swarm Optimization." **Complex & Intelligent Systems** 4, 2018, 227-239.

¹⁵ S.S. Kareem, R.M. Reham, A. H. Fatma, & M.E-B. Hazem M. "An Effective Feature Selection Model using Hybrid Metaheuristic Algorithms for IOT Intrusion Detection." **Sensors** 22, 2022, no. 4: 1396.

¹⁶ A.K. Balyan, A, Sachin, K.L. Umesh, K.S. Sanjeev, M. Poongodi, D.A. Abeer, E. Hela, & R. Kamran. "A Hybrid Intrusion Detection Model using EGA-PSO and Improved Random Forest Method." **Sensors** 22, 2022, no. 16: 5986.

¹⁷ L. Xu, B. Song, & M. Cao, M. "An Improved Particle Swarm Optimization Algorithm with Adaptive Weighted Delay Velocity." **Systems Science & Control Engineering**, 9(1), 2021, 188-197.

Chapter Two

Literature Review

2.1 Intrusion Detection

In today's interconnected and digitized world, computer systems and networks' security is paramount. Intrusion detection plays a vital role in ensuring information systems' integrity, confidentiality, and availability by identifying and mitigating potential security breaches. Over the years, numerous studies have been conducted to develop effective intrusion detection techniques and algorithms. This literature review aims to provide an overview of the existing research in the field of intrusion detection, focusing on various approaches, methodologies, and advancements.

Traditional IDSs typically rely on signature-based or rule-based approaches to identify known attacks. Signature-based IDS uses predefined attack patterns, or signatures, to match against network traffic and detect specific attacks. However, this approach suffers from the limitation of being ineffective against zero-day attacks or unknown threats. On the other hand, rule-based IDS utilizes predefined rules that describe specific patterns of network activities indicative of attacks. While rule-based IDS can detect previously unknown attacks, it requires constantly updating rules to keep up with the evolving threat landscape.

Anomaly detection-based intrusion detection systems aim to identify deviations from standard behavior patterns. These systems learn the expected behavior of a network or system and raise the alarm when observed activities deviate significantly. Machine learning techniques, such as neural networks, decision trees, and support vector machines, have been widely employed for anomaly detection. These approaches offer the advantage of adaptability to new and emerging

threats, as they do not rely on predefined signatures or rules. However, they can suffer from false positives and require substantial training data to model normal behavior accurately.

Researchers have proposed hybrid intrusion detection systems that combine multiple techniques to overcome the limitations of signature-based and anomaly detection approaches. For example, hybrid systems may utilize signature-based detection for known attacks and anomaly detection for identifying novel threats. This fusion of techniques aims to enhance the detection accuracy and reduce false positives and negatives. Moreover, the integration of machine learning algorithms and expert systems has shown promising results in improving the performance of intrusion detection systems.

The rapid growth of data generated by modern networks necessitates using big data analytics for intrusion detection. Big data analytics techniques, such as data mining, pattern recognition, and statistical analysis, enable the identification of hidden patterns and correlations within large-scale network datasets. By leveraging the power of big data, intrusion detection systems can effectively analyze massive amounts of network traffic in real time, enabling timely detection and response to security threats.

With the increasing adoption of cloud computing, intrusion detection in cloud environments has gained significant attention. Cloud-based intrusion detection systems face unique challenges like multi-tenancy, virtualization, and scalability. Several studies have proposed intrusion detection mechanisms specifically designed for cloud infrastructures, including behavior-based anomaly detection, log analysis, and machine learning-based approaches. The goal is to provide robust security measures that protect the confidentiality and integrity of cloud-based services and data.

Intrusion detection is a critical component of cybersecurity, aiming to safeguard computer systems and networks from malicious activities. This literature review has provided an overview of various intrusion detection approaches, including traditional IDS, anomaly detection, hybrid systems, big data analytics, and cloud-based detection. The research in this field continues to evolve, driven by the ever-growing threat landscape and technological advancements. Future directions for research may involve exploring the application of artificial intelligence, deep learning, and blockchain technologies further to enhance the capabilities and effectiveness of intrusion detection systems.

2.2 Steganography

By 1970, digital steganography had been introduced. In the initial audit, evidence found on the suspect's computer confirms the financial wrongdoing. The first person for a computer crime was brought to justice in 1996. If technology is both the catalyst for and an enabler of criminal activity, we say that corruption has been committed "through the computer." In 1996, the first case of a computer crime being prosecuted was filed in Texas, USA, leading to a 5-year sentence. The prevalence of personal use of the Internet and computers in the 1990s led to increased crimes committed using these technologies. The field of computer networks developed into its discipline in the late 1990s and into the 2000s. According to CSI studies, nearly 46% of responders have been victimized by internet crime. In 2010, 11% of American adults were targets of technology or Internet-related offenses in their houses, per the Survey. This proportion is up by 6-8% compared to the previous seven years. This is what the "Australian Company Latest Study" says about it. In 2006, fraud and data breaches cost Australian businesses A\$ 2 million. Company Crime Survey estimates that financial theft and data breaches cost the industry \$2,000,000 in revenue lost yearly. The phrase "digital investigation" has become more commonplace as more investigations use digital equipment¹.

Cases can be strengthened by a digital network, which involves identifying and retrieving data from digital devices and then archiving and analyzing that material for later use in court. Depending on the nature of the investigation, steganography can refer to various subfields, such as disk investigations, computer crime scene investigation, network security, Cyberspace ballistics, mobile information security, database e-discovery, or crypto criminal experts^{2,3}.

Cybercrime is rising at a shocking rate in this digital age. To combat cybercrime, law enforcement agencies must take appropriate action, which brings us to steganography. When the FBI began programs in their laboratories in 1984 to investigate crimes committed using computers, digital forensics was born. Different standards and definitions and steganography investigation models have changed with the advancement and development of digital forensics. Many digital forensic investigation models share some stages but are designed to investigate different types of crimes in various scenarios^{4,5}.

Crime analysis is "the adaptation of empirically generated and validated techniques to the retention, recognition, classification, Evaluation, perception, verification, and digital evidence gathered from electronic inputs to assist or further the restoration of offending behavior or to predict malicious access that destabilizes planning performance." In 2001, the initial DFWR (Digital Forensic Research Workshop) developed the DFRWS investigative Paradigm, a digital investigation model. It comprises six phases: recognition, protection, collecting, investigation, and Evaluation. The first stage, known as Identification, is used to determine the primary source of the crime. The preservation stage is the second phase/stage. Maintaining a chain of custody establishes a proper case during this phase. At this point, it is also ensured that no temptation has been introduced into the data collection process. The third phase is the collection phase, in which digital forensic investigators use numerous tools and methods to collect the essential data identified in the previous stage. The following two steps,

Examination and analysis, are critical. Evidence has been traced and collected from previously collected data using data mining and other techniques to prove the crime in a court of law. The final phase is the presentation phase, which documents all of the evidence/reports/data from the beginning of the investigation until the presentation in court⁶.

2.2.1 Investigation of Networks

Network forensics is a new and emerging field that detects and collects evidence to present in court for malicious activities that use the internet or networks as a source, such as DDOS attacks and data theft. Because data and information packets are sent from one device to another, the evidence's in-network forensics life span is concise. Many tools and techniques for network forensics have evolved over the years to find proof. Honeypots and intrusion detection systems are well-known developments in network forensic tools^{7,8}.

2.2.2 Network Techniques for Botnet Investigation

Network investigations is a new and developing branch of digital investigation that investigates network-related proofs in the form of logs, network flows, and packets. This section discusses Network Forensic methodologies for detecting and analyzing Botnets and their malicious activities. Botnets are investigated using the methods listed below^{9,10}.

Honeypots: Honeypots are simulated devices that perform their functions in a controlled virtual environment. Attackers and malware-infected bots are fascinated by honeypots. There are two kinds of honeypots, Low interface honeypots and High interaction honeypots are used to gather information by imitating the entire system (for example, Windows and Linux), etc. The main benefit of honeypots is that all attacks, malware infections, and unauthorized access attempts are automatically logged by honeypot operators, allowing them to create prediction patterns for future use. Pham and Dacier proposed low-interaction honeypot systems to

develop a plan that automatically detects Botnet attacks in large datasets. A completely automated honeynet architecture that can also reconfigure itself. Mittal and Singh created a methodology for determining the Intrusion Detection System rules based on data from Honeypots^{11,12}.

Deep Packet Inspection (DPI): this packet sniffing/inspection method provides a detailed inspection of data packets sent over the network to identify malicious content in the box. When attempting to analyze encrypted data, the main issue it raises is data confidentiality. Consequently, it is unable to detect zero-day malware. Deep packet inspection is still used today to gather more information about data packets by scanning the packet content¹³.

2.2.3 Machine Learning Role in Botnet Investigations

One of the most essential and well-known botnet detection methods is using a machine-learning algorithm to distinguish botnet traffic. Because botnets generate distinguishable traffic, machine learning procedures/approaches determine botnet traffic patterns that other techniques would not detect. ML approaches have advantages over traditional anomaly- and signature-based techniques¹⁴.

2.2.4 Frameworks for IoT and Smart Systems

Digital evidence is a set of techniques created and improved by the research world. Its task is to gather traces from a criminal investigation in a safe way, look at them, analyze them, and produce evidence that can assist in finding the bad guys, their methods, and why they did what they did during an investigative process of a security breach, which mainly gives rise to litigation. Crime analysis has become more specialized over time and with the help of new technology. There are now several subgroups, such as cloud environment, penetration testing,

Internet of Things (IoT) evidence collection, mobile crime scene investigation, memory crime scene investigation, and data ballistics¹⁵.

Security analysis pathways look into security problems that happen in networks. They often use logs and packets that have been collected to find attackers and people doing bad things. Network forensics experts use various tools to gather and maintain substantiation appropriately so that it can be looked at later. But there is no best way to explore, so there are a lot of different network investigation architectures. In its most basic form, a network investigation structure tells experts what steps to take when in an investigative process. It breaks the research into separate, independent actions, with its collection of tools and methods¹⁶.

2.2.5 The NIST Digital Investigative Procedure

Even though numerous data recovery methods are already in place, they use the NIST Special Publishing 800-86 as their guide because it is widely used and can combine various methodologies with disaster management. IoT crime scene investigators, like some of the other categories of digital evidence collection, may cross borders and necessitate specific requirements and prerequisites, such as how to collect substantiation and whether it can be used in court or not. We think that live investigative preparation is also significant even though IoT devices may be used in areas of critical systems where carrying them down for maintenance for evidence collection is not possible¹⁷.

Collection: Particularly in real-time forensics, it is essential to quickly identify prospective information resources in (integrated) IoT ecosystems. Unfortunately, this can be challenging to execute manually in IoT systems because of the changing nature of data relationships. This means we may leverage ML approaches, such as classification algorithms, to automate the

collection process (e.g., Naive Bayes, Nearest Neighbor, and Support Vector Machines classifiers). A careful balancing act between false-negative and false-positive outcomes is required¹⁸. Digital information obtained from evolving connection components may be pre-processed, the recommended initial (including such encryption algorithm and cryptographic hash automated system for use) chosen, and techniques available selected for analyzing the information (as an example, using logistic regression to assess the data gathered thus far to establish any correlations among items of interest and factors)¹⁹.

Analysis: If the test goes well, we can confidently choose our methods and equipment. Which way, such as K-Nearest Neighbors or Decision Trees, should be used? The k-nearest-neighbors algorithm can identify the nearest element in an environment by calculating the magnitude of the difference between them. In contrast, decision trees can split slightly gathered data into lesser groups while simultaneously generating a corresponding decision tree. Actual investigations and in-depth investigations will then be performed on the data²⁰.

Reporting: The report will detail the results of the study, the methods utilized, the thinking behind those methods, and any caveats or restrictions placed on those methods (if any). For instance, where do classification methods like Neural Networks fall short? Would any information have been lost when using these categorization methods in real-time investigations? Web servers, such as database management systems, webpages, and the Internet of Things, are prominent targets for application assaults²¹.

Injection of Code: Integrating information into a web software application to run or understand dangerous material unpredictably is a common attack vector, and remote code execution attempts are a subclass of this broader class of assaults. As per OWASP's top ten susceptibilities (<https://owasp.org/www-project/top-ten/>), SQL injections are among the most

prevalent and efficient in Cyberspace because of their versatility, attack surface area, and the difficulty with which they are occasionally protected. Injection problems arise whenever an interpretation is given untrustworthy data as a command or query. Because of the suspect's adversarial data, the interpreters may carry out undesired instructions or acquire forbidden information²².

Code injection is a particularly pernicious form of attack. The server, incorrectly assuming that the user's query is genuine and attractive, leaves itself vulnerable to assaults of this type. When spotting attacks, the repercussions of erroneous data injection might be devastating for conveying system malware to proliferate. During an attack, malicious code is entered as user input into the Web application to steal sensitive data. SQL and NoSQL queries, OS functions, XML scripts, SMTP protocols, and other programs are all vulnerable to malicious code injected into them. The purpose is to introduce hacking tools that disrupt the intended functionality of the software and cause it to execute unexpectedly. Websites become infected with malware when intruders incorporate or integrate harmful code. Malware of this sort can interfere with the proper functioning of online resources. The availability of online resources may also be jeopardized²³.

Cross-Site Scripting: An attacker may exploit a flaw in web services by inserting code (like JavaScript) to obtain an advantage against users in cross-site scripting (XSS). Utilize it on accessible public pages like the header or a page whereby users could offer feedback. The exploit can only happen if the website has a search bar or actual article that the adversary can use to access the system. Web applications are vulnerable to cross-site hacking attacks, which occur when malicious scripts are inserted into a website to steal data from another computer. More than half of all web server assaults in 2009 were of this sort²⁴.

Kernel Attacks: This kind of damage flaws in kernel modules to escalate the user's privileges. Skilled attackers have made it their go-to weapon, which is widely considered among the most serious weaknesses. Derusbi, RobbinHood, Sauron, Uroburos, and GrayFish are just a few examples of malware that used kernel module flaws to elevate their permissions and turn off security, thereby attacking devices and networks. Recollection degradation, policy violations, denial of operation, and knowledge leaking are the four primary forms of kernel attacks, and most undertakings are implemented in unsafe technologies. Because of this flaw, an intruder might potentially learn the content and structure of the kernel's storage, permitting him to bypass kernel-level encryption and access private kernel information²⁵.

Microarchitectural Assaults: Using consequentialism, this modified concept of vulnerability leaks private data through prefetching pathways and other means along wrongly stated paths. Owing to their capacity to manipulate the execution of the path flow information, these attacks can compromise privacy protection and cause critical hardware and software protection safeguards to be ignored. Accessibility is gained to the perpetrator's shared memory space via the installation of a process on their workstation²⁶.

Information Flow Attack: Data that should be kept private is stolen in this manner. When an adversary can discern confidential details by studying the connection between personal and accessible outputs and inputs, we discuss a data flow attack. Since it links data streams to stealthy assault possibilities, this technique finds application in many situations²⁷.

Verification: An authorization effort will be made to gain access to assets without the victim's identification. Increasing one's privileges is the usual means of achieving this. The term "session hijacking" refers to discovering a situation in which a client or user has obtained

access to information that is typically off-limits to them due to a flaw in the code, a security hole, or a misconfiguration²⁸.

Malicious Insiders: Malevolent individuals can be current or former personnel and strategic partners who abuse their access to information and data for malicious purposes. Employees who accidentally compromise the company's protection or release information because of a human mistake, a deficiency of technological skills, or incompetence are not included in this threat. Organizational insider threats are hazardous since the perpetrators already breached the company's defenses, have essential information, and know system vulnerabilities²⁹. An individual's virus/malware deployment to steal data is a major security risk. Although insider assaults pose a significant threat, most organizations overlook them to protect themselves from potential hackers. Until a company suffers an insider assault, it doesn't give much thought to its associates and does little to improve its network security. In addition, harmful corporate cyberattacks are often preplanned³⁰.

Session Hijacking: Ethical hacking occurs when an adversary discovers a previously defined relationship between a client and a provider and gets possession of it. The user-host can be established via any protected service, including a web application or other Protocol communication. To spy on the patient's activity and perform specific assaults, intruders insert malware between the client and the hosts. A connection hijacked enables an intruder to take on the authorized user's identity, analyze information, and infiltrate or gather targeted items. When in control of the victim's connection, the attacker takes on their identity and can carry out any operations ordinarily possible for the victim³¹.

Qualification: Vulnerability scanner software is ordinarily used in authentication challenges, where an outside entity tries to acquire a system's access by guessing or guessing at an

authenticator. Identities can be broken into several ways, the most common of which are phrase attacks and brute-force account reading. In none of the information we looked at, did we find evidence of this specific attack type³².

Use of Force: To crack a passcode and acquire the target's identity or computer, the adversary has several combinations of that password. Among the many variations of brute force, assaults are identity loading and, the opposite, the counter brute-force attack. Passcodes that are easy to guess or are insecure are more vulnerable to brute-force attempts³³. This type of attacker aims to get into a secure resource by exploiting a vulnerability in the authentication process, namely an insufficient username and critical grouping. Brute-force cyberattacks often utilize a secret database to try every possible combination of an authenticator. This user conduct makes brute-force attacks and identity takeovers easier to pull off³⁴.

Keylogger: Keyloggers are a form of security device that tracks and documents all input from the keyboard on a personal computer to steal sensitive information such as login credentials. A device or software-based keylogger is an acceptable means of IT surveillance for both private and business use. In contrast perspective, a keylogger has nefarious applications. Creating such a log is a common side effect of malware infections, which harvest personal data like credit card details and passwords for illegal purposes³⁵.

Man-in-the-Middle: This form of cyberattack happens when an attacker reads a user's information by sending and pretends to be the recipient of that information, like on a banking web, a user account message, or a social networking site. An adversary can manipulate or stop data being exchanged between involved individuals without understanding either party. One typical scenario involves hackers intercepting a user's communications with his account to

gain access to his account details. The intruder can then exploit this data to their advantage. Encrypted links are recommended as they raise the difficulty for any would-be attackers³⁶.

Phishing: Phishing is the fraudulent online solicitation of private information (e.g., passcode, payment card details) via electronic means. A sufferer may get a message via email or text from someone pretending to be an acquaintance, a legitimate business, or even the authorities. The recipient is tricked into visiting a phony website after opening the e-mail or receiving the SMS message. The victim is then confident enough to submit his login credentials. This enables identity theft, checking account takeovers, and the sale of private information on the dark web by the criminals who get the sent access details. Phishing is a thought control technique that targets victims by impersonating reputable businesses to steal their private and financial information³⁷.

Network Attacks Unauthorized: Cyberattacks are any activities taken to cause damage to a structure or steal information. Network protocol threats affect the monitoring and screening of essential news, which can compromise the business's and its customers' safety. At the same time, active assaults involve the actual modification or deletion of information³⁸.

Botnets: A chatbot is a piece of code that can be replicated continuously by taking advantage of loopholes in an operating system of the computer. Simply put, botnets are groups of systems a bot has hijacked. Hundreds, if not thousands, of computers make up these connections. An offender in control of a botnet can utilize it to increase the effectiveness of their operations, including generating hundreds of misleading or phishing electronic mail or initiating generalized denial-of-service threats. Botnets are complex attack vectors that aim to compromise systems and machines. To begin an attack, hackers surreptitiously enter a

network and run code through a Trojan. A wealth of publicly available information about botnets has emerged as one of the most studied forms of attack³⁹.

DNS Attacks: If an intruder can tamper with hostname (DNS) servers, they can trick users into visiting malicious websites by providing false answers to their authentication and authorization inquiries. When a victim's computer requests a webpage domain, a misconfigured DNS server may return the Internet address instead for a corrupt webpage duplicate. The attack tricks the victim into visiting malicious websites where personal information can be stolen by fraud or direct debit theft. The attacker can make money by tricking people into visiting malicious advertising websites or installing malicious software. A cyberattack of this kind⁴⁰ can jeopardize the integrity of the underpinning infrastructure.

2.2.6 Denial of Service (DoS) and Distributed (DDoS) (DoS and DDoS)

In all cases, the attacker aims to prevent authorized users from retrieving the targeted service networks. Attackers often attain this by overwhelming the traffic targets or submitting malicious packets that lead the target to malfunction or fail. There are two main types of DoS attacks: those that target specific users and those that aim to make networks or databases unavailable to everyone. These episodes can persist for a few minutes to several hours or even a few days. If an organization is targeted and lacks enough protection, it could suffer devastating financial damage⁴¹.

The intruders flood the computer with packet transfers until it cannot process anymore. Because of this, no one can access the information stored in computers, as the systems are down and unable to process any requests. One central server can control Many botnets in collections throughout a DDoS attack. Consequently, all zombies coordinate their actions and repeatedly demand the same thing from the victim. This is because more people are trying to

access the website simultaneously than the server can handle. The server may need to restart or become unresponsive if an attack on a critical resource occurs. 2018 saw a rise of 40% in this kind of attack, with more than 400 thousand occurring per month. This type of attack can be spotted by monitoring the data transferred between the attacker's IP addresses and the victim. It's a significant concern because it's the most popular form of assault and the first line of defense against securing a connection from intruder^{42,43}

Reused IP Address: When a user disconnects from a network but another user is assigned the same IP address, this creates a security hole. It's because the new subscribers may utilize the old IP address to gain access to the senior user's data^{44,45}.

Network Sniffing: This is a technique a hacker can use to intercept and examine every packet of data carrying confidential details that travels through a specific network. Both hardware and software sniffers are available for integration into a given system. This program captures and analyzes packets received by the device's local network. Sniffing is performed to gain unauthorized entry or steal sensitive data. Sniffers are particularly handy for cybercrime because various network programs (HTTP, FTP, SMTP, POP3, etc.) convey data in textual format (for example, user names and passwords). Users commonly re-use the same login and password across various services, making it easy for hackers to access all of them^{46,47}.

Physical: Threats that include using physical force to gain entry to a target's structure, devices, or network connections are known as "direct attacks." Any facility or piece of machinery is vulnerable to this sort of attack. In none of the data we looked at we found evidence of this specific attack type^{48,49}.

Equipment Theft: Theft or equipment damage was the leading cause of data leak incidents in 63% of small and mid-sized businesses in 2019. Since the patient's components are also set up

correctly, thieves who steal a device can immediately reach the computer networks. Most consumers don't realize that doing something as simple as utilizing a pen drive might lead to fraudsters stealing sensitive company information. Confidential firm information can be jeopardized, for instance, if an unprotected private note is lost or stolen^{50,51}. Thievery of devices leads to problems, like loss of hours at work, wasted proprietary information, lost client information, disrupted operations, subversion, financial loss, and even negative publicity. A data loss incident might be classified as "theft and damage" if the resource disappears without a trace, whether by accident or malice. Insiders committed Three-fourths of attacks, seventy-one percent of intrusions were economically successful, and fifty-six percent of compromises were not uncovered for three months or more, all based on the same poll⁵².

Unauthorized Accesses: An insider attack occurs when a malicious outsider gains access to a system, network, or data belonging to a company. It is typical for an attacker to perform a simulation before an actual incident to Validate the channel's protection and locate weak spots⁵³. Unauthorized entry or attempted entry into a system with the intent to cause disruption, destroy, or commit fraud is what this type of assault is referring to. There is an upsurge in the number of international adversaries and the rate of official information misusing their rights for personal gain as automation and Online services in businesses increase. The confidentiality of sensitive information must be maintained at all costs. Confidentiality, user access, and cryptosystem are security measures companies can take. Researchers have focused a lot of attention on methods for finding breaches in security⁵⁴.

Virtualization: A virtual box creates an abstracted version of a physical system, either a software application, which other programs can execute. Virtualization is targeted because it

controls all VMs in a data center or service. Virtualized setups do not have a central repository of secure and convenient^{55,56}.

Hypervisor Rootkit: A rootkit is a series of nasty programs enabling intruders to hide their tracks while opening a computer's most sensitive files and systems. This allows the virtualization rootkit to eavesdrop on devices and host OS operations and conversations.

The proliferation of rootkit technologies has made virtualization a potential entry point for malicious actors. Malicious users can access the virtual computer infrastructure using this attack, which has the possibility of infecting the entire virtual network. In this way, the adversaries may take control of all virtual machines^{57,58}.

VM Exit: This assault makes use of a vulnerability in hypervisor security. This flaw allows an attacker to run arbitrary code within a VM, which might then be used to compromise additional VMs or even the host operating system. This means it can use the host computer's RAM, CPU, and disk space as if it were the actual VM^{59,60}.

2.2.7 Discussion of the Different Types of Attacks

Application malicious activities leverage well-known entry points to the server to carry out malicious or otherwise inappropriate tasks. These assaults count on the fact that a conventional server will acknowledge legitimate client requests. To a greater extent than usual, software used in consumer technology is distributed despite meeting the standards of the existing security existence^{61,62}. By passing the authentication process, hackers can access a company's network and steal sensitive information. Attackers employ loopholes in the authentication techniques and procedures to break into systems and obtain access. Today, many Internet users still employ the bare minimum of security by using only their email address and a random passcode to access their accounts, making them vulnerable to these

kinds of attacks. Attackers can also find openings in your defenses if you don't have reasonable internal control, regularly update and maintain your devices, and have a comprehensive data security strategy in place^{63,64}.

Identification attacks encompass a wide range of techniques to gain unauthorized access to a system that requires a username and password. Numerous software developers still cling to the ineffectiveness of using single words as passwords, even though such attacks are overwhelmingly unsuccessful with computers that require multiple characters and a range of elite and uppercase and lowercase numbers and symbols. Brute force attacks, while sometimes necessary, can be time-consuming and frustrating. User training is a powerful weapon against psychological manipulation^{65,66}.

Attacks on a network usually aim to compromise the system or make users' access to network resources unavailable. The potential for abuse increases with the proliferation of online SaaS products. The industrial base or a way to determine it can be attacked by hackers using botnets or DNS intrusions. Since most Internet users hide their identities, it's hard to pinpoint who's responsible for an assault. Also, this is the most studied type of assault in the field, and the studies in the collection show that most major firms have implemented multiple safeguards to ward off such attempts⁶⁷.

Physical assets are subject to a wide range of losses and thefts, which pose serious security concerns. Another possible situation is that an adversary has gained unlawful access to sensitive corporate information. Corporations' efforts to execute regulations are becoming more sophisticated, but adversaries are always thinking of new methods for circumventing them. Misinterpretation of confidentiality; (ii) a complete absence of rigorous control environment about the facility authentication scheme; (iii) managers who are using corporate

funds confidentially and genuinely; (iv) the deployment of cloud computing that further makes it possible for individuals to carry confidential information on computers, mobile phones, and smart operators that may be reported stolen, impeding the data⁶⁸.

Lastly, in virtualization, attackers aim to take over the running of host VMs before migrating onto additional VMs or some other virtualization. Generally speaking, automated platforms are more susceptible to attacks than their non-virtualized counterparts. Because of this, the security of the overall virtualized setting may be compromised. An intruder can compromise a virtual network and get the administration to all VMs within it⁶⁹.

2.3 Intrusion Detection Systems (IDS)

It is essential for communication and sharing information that computers can connect. On the other hand, computers can be vulnerable to attacks from the outside world and require constant surveillance and detection. These dangers can potentially compromise all three pillars of computer security: confidentiality, integrity, and availability (CIA). An attempt to breach the CIA or to circumvent the security mechanisms of a computer or a network is referred to as an incursion. The process of monitoring and analyzing the traffic in a network or a computer in search of indications of an intrusion is called intrusion detection. The primary goals of an intrusion detection system (IDS) can be summed up as follows: monitoring hosts and networks, analyzing the behaviors of computer networks, producing warnings, and reacting to suspicious behaviors⁷⁰.

The research literature has a significant number of IDS systems. Nevertheless, many IDS are plagued by two issues: A high false alarm rate is one in which alarms are triggered from infractions that do not pose a hazard. At the same time, numerous more significant violations go unnoticed. Because new attacks are difficult to identify, there has been an increased focus

on applying contemporary machine learning techniques to the problem of identifying intrusions⁷¹.

In general, there are two different ways that intrusion detection systems (IDS) can be categorized: classification based on the technique and type based on the monitored platform. In the following sections, we will go through these classifications further. Anomaly detection and misuse-based IDS are the categories that can classify IDS according to their methodologies. Methods of anomaly identification begin by modeling the user's behavior to construct a regular (or standard) behavior. This method observes how a user behaves over time and then tries to build a model as similar as possible to the user's typical behavior. The intrusion detection system (IDS) can detect suspicious behavior and sound an alarm whenever an event occurs that deviates from the typical behavior of the system. This characteristic makes it excellent for detecting zero-day assaults. Two primary drawbacks are associated with this model: one is an inability to deal with the ongoing evolution of regular user behavior, and the other is a high rate of false positives⁷².

On the other hand, whether utilizing the signature-based method or the misuse detection method, the events that have been captured are compared with known assaults or threats to identify any potential intrusions. Signatures are a common term that is used to refer to known attacks. Compared to the method for detecting anomalies, this approach is more precise and produces fewer false positives. On the other hand, it does not help identify newly launched assaults. Host-based intrusion detection systems (HIDS) and network-based intrusion detection systems (NIDS) are the two forms of IDS that can be categorized according to the monitored platform (NIDS). HIDS can monitor and analyze the actions taking place on the system (host) on which it is installed. It can monitor certain aspects of the dynamic behavior as well as the current state of the system. The Network Intrusion Detection System (NIDS),

on the other hand, is responsible for monitoring network traffic to identify remote attacks, which are carried out over a network connection. It is a gadget dispersed around networks to monitor the traffic that passes through the devices it sits on. It could be a hardware-based system or a software-based one. NIDS typically features two network interfaces; one is used to listen in on talks on the network, while the other is utilized for control and reporting purposes⁷³.

The purpose of intrusion detection systems is to audit computer systems and network operations for signs of malicious activity and network policy violations. In the 1980s, the IDS model was first described in its entirety. IDSs can be categorized in several different ways depending on the method used. The first two types are network-based and host-based, and their distinction is in the location from which the invasive behavior was noticed. Monitoring and analyzing network traffic is the primary emphasis of network-based intrusion detection systems (IDSs). Host-based intrusion detection systems determine whether or not a malicious action has been taken by monitoring the processes and events of the software environment associated with a specific computer⁷⁴.

Another way to categorize IDSs is according to the data analytics methods implemented, which can be broken down into signature-based (misuse-based), anomaly-based, and hybrid. The signature-based approach examines network packets or data from a specific system (such as logs) to locate signatures, which are patterns typical of intrusive conduct. Because it uses data from databases that have already been labeled, this method is substantially more effective than others regarding known assaults. Although it is a straightforward strategy that gets the job done, it cannot identify previously undiscovered threats and calls for frequent database updates⁷⁵.

The data are analyzed using an " anomaly-based " method to recognize abnormal events distinct from the expected behaviors of networks and systems. This capability might be accomplished by using data provided in the past and putting it through its paces to train a specific algorithm. The strategy that has been presented holds promise since, in contrast to the technique that has come before it, it permits the discovery of zero-day assaults. Additionally, it enables more sophisticated customization for the specific system or network in question. Because these techniques are not only based on labeled data but are also taught to recognize anomalies based on data provided in the past, they may end up finding situations that are not necessarily cyber security attacks. This is a significant drawback that this case presents, and it is because a high level of false positive alarms⁷⁶ characterizes these kinds of techniques.

Signature intrusion detection systems, known as Knowledge-based or Misuse Detection, are based on pattern-matching techniques to identify known attacks. SIDS is another name for these types of detection systems. When conducting SIDS, matching procedures are utilized to locate a previous incursion. In other words, an alarm signal is triggered if an intrusion signature coincides with the signature of an earlier intrusion already in the signature database. This allows the database to track and prevent future intrusions⁷⁷ more effectively. To implement SIDS, the host logs are investigated to detect sequences of commands or actions that have already been categorized as malicious. In the body of academic research, Sudden Infant Death Syndrome has also been referred to as Knowledge-Based Detection and Misuse Detection⁷⁸.

The primary objective is to compile a database of intrusion signatures, which will then be compared to the existing signatures, and an alert will be triggered if a match is discovered between the current set of activities and any of the signatures. For instance, a rule that takes the form "if: antecedent -then: consequent" might result in the statement "if (source IP address=destination IP address), then mark as an attack." In most cases, SIDS provides a high

degree of detection accuracy for intruders who have already been identified. However, SIDS has trouble remembering zero-day attacks because, until the signature of the new assault is extracted and saved, there is no matching signature in the database. This makes it difficult for SIDS to detect zero-day attacks. SIDS is utilized in a wide variety of everyday tools⁷⁹.

The conventional techniques for SIDS involve inspecting network packets and attempting to match them against a signatures database. However, these methods are unable to detect assaults that include many packets at the same time. Extracting signature information across numerous packets may be necessary due to the increased sophistication of current viruses. To accomplish this, the IDS must remember the contents of prior packets. When developing a signature for SIDS, there have been a variety of approaches, the most common of which involve developing signatures in the form of state machines, formal language string patterns, or semantic requirements⁸⁰.

SIDS approaches are becoming less effective as the number of zero-day attacks continues to rise. This is because no previous signature exists for any attack that uses zero days. The old model's validity may be further called into question by the emergence of the malicious software's polymorphic forms and an increasing number of targeted attacks. The application of AIDS methods, which determine what constitutes normal behavior rather than what constitutes abnormal conduct, could prove to be a practical approach to resolving this issue⁸¹.

Anomaly-based intrusion detection system (AIDS) AIDS has piqued the interest of a significant number of academics as a result of its capacity to circumvent the limitations presented by SIDS. Methods that are statistically based or knowledge-based, as well as machine learning, are used in AIDS to construct a standard model of the behavior of a computer system. A behavior that significantly deviates from the model is viewed as an

anomaly, and this difference in behavior might be understood as an intrusion if it is significant enough⁸².

The presumption underlying this collection of tactics is that harmful behavior is distinct from the usual conduct of a user. Intrusions are characterized by the fact that deviant users' behaviors differ from those considered typical. The first step of the development of AIDS is the training phase, and the second phase is the testing phase. During the training phase, the average traffic profile is used to learn a model of normal behavior. Then, during the testing phase, a new data set is used to establish the system's capacity to generalize to previously unseen intrusions. This is done so the system can better protect itself from future threats. Depending on the approach that was taken during training, there are a few different classifications that can be applied to AIDS. These categories include statistical, knowledge-based, and machine learning-based based⁸³.

The capacity to detect zero-day assaults is the primary benefit of AIDS. This is possible because AIDS does not depend on a signature database to recognize abnormal user behavior. When an individual's behavior is observed to deviate from their typical pattern, AIDS sends out a warning signal. In addition, there are a few upsides of having AIDS. To begin, they have the power to uncover any unscrupulous goings-on within the organization. An alert will be triggered if an unauthorized user begins doing transactions in a stolen account that are not consistent with the regular activity of the account's owner. Second, because the system is built from unique user profiles, it is exceedingly challenging for cybercriminals to determine a typical user activity that does not trigger an alarm⁸⁴.

When compared to AIDS, SIDS can only recognize well-known intrusions, but AIDS can identify zero-day attacks. However, AIDS might result in a high false positive rate because

abnormalities may be new normal activities rather than actual incursions. This is because AIDS can cause further everyday actions to be misinterpreted as abnormalities. Due to the absence of a taxonomy for anomaly-based intrusion detection systems, we have classified these systems into five subclasses based on the characteristics they share, which are as follows: Based on statistics, patterns, rules, states, and heuristics respectively⁸⁵.

IDS was segmented into categories according to the methodologies employed to identify intrusions. IDS can also be categorized according to the input data sources that are utilized in the process of identifying anomalous activity. Host-based intrusion detection systems (HIDS) and network-based intrusion detection systems are the two primary categories of intrusion detection systems (IDS) in terms of the data sources they monitor (NIDS)⁸⁶.

The data that HIDS examines originates from the host system and audit sources, such as the operating system, window server logs, firewall logs, application system audits, and database logs. HIDS can detect insider threats even if there is no network traffic involved. The network traffic retrieved from a network via packet capture, NetFlow, and other network data sources is what the Network Intrusion Detection System (NIDS) watches over. A network-based intrusion detection system can monitor multiple computers at once when connected to the same network. NIDS can monitor the external harmful activities that could have begun from an external threat at an earlier phase before the dangers spread to another computer system. This allows NIDS to prevent the hazards from spreading to other computer systems. On the other hand, due to the sheer amount of information that travels via contemporary high-speed communication networks, NIDSs can only inspect a portion of the data included within a high bandwidth network. When implemented in conjunction with host intrusion detection systems (HIDS) and firewalls, network intrusion prevention systems (NIPS) have the potential to offer

a practical, robust, and multi-tiered defense against attacks from both the outside and the inside of a network⁸⁷.

HIDS methodology uses discontinuous system call patterns, aiming to increase detection rates while simultaneously reducing the number of false alarms. The fundamental concept is to provide a semantic structure to kernel-level system calls to comprehend unusual behavior in computer programs. AIDS systems have also been included in Network Intrusion Detection Systems (NIDS) and Host Intrusion Detection Systems (HIDS) to improve detection performance by implementing machine learning, knowledge-based, and statistical techniques. System calls, application program interfaces, log files, and data packets gathered from well-known assaults are the data sources. These data sources may be helpful when differentiating between abnormal and intrusive behaviors⁸⁸.

Three primary categories can be used to classify AIDS treatments based on information, statistics, and computerized methods of instruction and learning. The way that relies on statistics entails compiling and analyzing each data record associated with a collection of things and developing a statistical model of typical user behavior. Machine learning methods, on the other hand, acquire complex pattern-matching capabilities from training data. On the other hand, knowledge-based methods attempt to determine the requested actions based on data already present in the system, such as protocol specifications and examples of network traffic⁸⁹.

2.3.1 Assaults against Cyber Security

An IDS system can be set up to detect a wide variety of different types of attacks. The objective of an attack, its legal classification, the degree of severity involved, the extent of the attack, and the types of networks it targets are all valid categories for classifying cyber security attacks. Reconnoitering attacks, access attacks, and denial of service assaults are

examples of attacks carried out purposefully. A reconnaissance assault is a particularly treacherous form of cyberattack since it requires the attacker to trick their victims into thinking they are friends with them so that they can obtain necessary information from them. The use of packet sniffers, scanning the port, and inquiring about various aspects of the internet are all examples of these kinds of assaults. The intruder can access the device they are attacking in access attacks⁹⁰.

Phishing, social engineering, man-in-the-middle assaults, and attacks on secret code are some of the various attacks that fall under this category. A denial of service attack (also known as a DoS attack) is the third kind of attack that falls under this category. The perpetrator of a denial of service attack uses the fact that the victim site is connected to the internet to turn off the services that the victim site provides. This is accomplished by overwhelming the victim site with many requests. It can be a single-source attack or a multi-source attack, in which case it would be known as a distributed denial of service attack (DDoS). Smurf, SYN flood, and DNS attacks are distributed denial of service attacks⁹¹.

The legal classification of attacks is the second form of categorization, and it includes types of attacks such as cybercrime, cyber espionage, cyber terrorism, and cyberwar. An illustration of a cybercrime attack is identity theft, which refers to using a person's online account without that person's authorization. Cyber espionage, also known as cyberspying, is another type of activity that falls under this category. Cyber espionage, also known as cyberspying, involves using computer networks to gain unauthorized access to confidential information, particularly that associated with governments. Attacks in the form of cyber terrorism are carried out by extremists utilizing cyberspace. Cyber wars use the internet and other forms of digital communication to wage war between states. Attacks can be divided into two distinct sorts, active attacks, and passive attacks, according to the degree of involvement they require from

the target. This is the third form of classification. Simply put, the difference between these two types of attacks is that, in the first type, the attacker intends to change the system's resources or modify its operation. In contrast, the attacker uses the information in the second type without changing the resources or processes. Spoofing, man-in-the-middle attacks, buffer overflow, and several other types of assaults are examples of active attacks. Examples of passive assaults include monitoring every keystroke and installing backdoors⁹².

Cyber-attacks can be divided into two subcategories: malicious attacks and non-malicious attacks. To attack to cause damage, malicious attacks use many kinds of software. These forms of software include viruses, worms, Trojan horses, spyware, adware, botnets, and other types. On the other hand, non-malicious assaults are attacks carried out unintentionally by staff members who are not adequately trained, and they may result in a small amount of data loss. Cyber security attack classification is based on the types of networks in which assaults are categorized according to the kinds of networks, such as mobile ad hoc networks (MANET) and wireless sensor networks (WSN). Attacks on MANET can take various forms, such as the black hole attack, the flood rushing attack, and the Byzantine attack. Attacks on the application, network, and even higher layers are further examples of what might happen on WSN⁹³.

2.3.2 Identifying an Identity Clone Attack

The idea of conducting a security Evaluation of emerging technologies, such as the infrastructures of IoT and WSNs, using the Security-in-Depth paradigm introduces the concept of minimizing any risk by maximizing the effectiveness of various security layers. This is done so that any potential vulnerabilities can be eliminated. In this discussion, security layers refer to the combination of physical and logical control methods that reinforce the ability to prevent, detect, delay, or respond to any attempt to disrupt the system's security. In this way, intrusion detection systems (IDS), intrusion prevention systems (IPS), security

information and event management (SIEM), and firewall appliances are combined, installed at various network layers, and made to function following anomaly- and behavior-based signatures. When a harmful pattern is matched and detected, the intruder may be prevented from attempting additional incursions or eliminated from the system⁹⁴.

Corrective controls have been developed to accomplish a specific level of clone node attack prevention and mitigation across Internet of Things and Wireless Sensor Network protocols. It is essential to emphasize that although the presented methods have the potential to alleviate various weaknesses, either entirely or experimentally, there are no ML, DL, RF, or statistical implementations to compare them to. Even more crucially, the improvements are not mainly geared toward RPL, which creates a gap in research on more intelligent solutions. It is essential to point out that the adaptations, trials, specifications, and tests carried out all significantly impact the stated complexity's degree of variation, which is exponential. This has an immediate effect on the data regarding the memory cost because it is a measurement that indicates the difficulty of the number of operations that the solution needs to conduct to observe the dynamics of each node to identify a suspicious movement⁹⁵.

However, it is true that RPL, a component of many IoT and WSN routing protocols, may already have documented detection signatures for its typical adversarial patterns. This is a possibility. A preliminary study found that extending IDS anomaly-based signatures on specific network segments, where physical nodes provide a first authentication on the DODAG tree, can help reduce the number of identity theft and counterfeiting attacks. At this point, it is possible to ascertain geographical information and prohibit nodes that are considered suspicious. On the other hand, standard intrusion prevention systems (IPS), intrusion detection systems (firewalls), and other comparable defensive systems are prone to produce a high proportion of false positives. Because of this, using similarities as a basis for comparison is ineffective when correctly understanding the peculiarities of stealthy attacks

like the Clone ID attack and taking informed steps in response to them. To detect node ranking attacks on RPL, a host-based intrusion detection system (IDS) outfitted with a machine learning (ML) engine and a statistical learning model has been developed. This development is essential to reducing more sophisticated attacks directed at the DODAG organization. In any event, the Clone ID attack surfaces and defenses in RPL networks remain undiscovered, as with many recently discovered dangers related to the Internet of Things⁹⁶.

An assault using a cloned ID on the RPL protocol During the process of sensing RPL-based connections, it is possible for the DIS and DIO control messages to be manipulated or created to commit identity counterfeiting, more specifically, the Clone ID attack. This is an attack in which malicious actors learn about the configuration of DODAG trees and modify the information necessary to add a malicious node. An attack such as this can succeed because the default security and self-healing features in the RPL standard version are not meant to be implemented. In point of fact, the IEFT gives its unequivocal endorsement to implementing data confidentiality and integrity configurations that use pre-shared keys. These qualities can potentially improve the dependability of the resources. Still, their implementation has a significant negative impact, particularly on the nodes' mobility and the network's overall performance. As a result of the complexity of the self-configuration and organization between nodes, which makes critical management and peering a significant challenge, it has also been demonstrated that a malicious actor operating within the network can easily circumvent security controls. This problem is because it allows the malicious actor to avoid detection easily. Based on this, nodes can join freely without enforcing any authorization or authentication rules, which opens up a fantastic chance to duplicate many nodes in various locations. A hostile actor detects a network built on RPL and targets it; next, the DODAG configuration is read, and a malicious node Z learns from its siblings' identities to mimic a specific one, E. At this point, Z continues to function as a sibling node, but it is no longer in

direct competition with the node being attacked. The malicious node, Z, continues to send forged DIS control messages that contain the cloned identity of E, another sibling node until access is granted to the network. Therefore, when the malicious node, Z, is accepted as a member of the graph, all other nodes in the network send a DIO control message to reconstruct the DODAG topology⁹⁷.

Communication with the malicious node, Z, can commence in a manner that is not obvious to the user. It is essential to point out that the legitimate node, E, may send another DIS message to restore communication with the network. This will result in continuous competition between the malicious node, Z, and the legitimate node, E, to gain trustworthiness among the graph members. The winner will be the one who can maintain DIS messages longer than the other competitor. This is attainable due to two primary factors: According to the IETF, the RPL protocol uses a mode termed Unsecured as its default setting. In this mode, the fundamental messages DIS, DIO, DAO, and DAO-ACK are susceptible to easy modification and do not undergo any integrity or authenticity check. The ability of the attacker to mimic a node within the network confers a significant advantage to them. RPL enables any node to join the DODAG tree at any moment, using DIS and DIO messages to include itself and rearrange the DODAG tree, as was discussed earlier. This is possible because any node can join the tree at any time. An illegitimate node can take advantage of the Trickle Timer mechanism of the protocol, which generates persistence within the DODAG tree by sending DIO messages at regular intervals to check the DODAG tree's stability. This will allow the illegitimate node to take advantage of the protocol⁹⁸.

An intrusion detection system based on statistics first constructs a distribution model for an everyday behavior profile, then detects occurrences with a low probability and marks them as potential intrusions. In its most basic form, statistical AIDS involves considering statistical

metrics such as the packets' median, mean, mode, and standard deviation. In other words, rather than analyzing the data traffic, each box, which represents the flow's fingerprint, is watched. This provides a more accurate picture of the flow. Statistical analysis is used to determine whether there are any significant deviations in the present behavior compared to the normal behavior⁹⁹.

Univariate: Since "uni" is short for "one," it indicates that just a single variable is present in the data. When a statistical average profile needs to be constructed for only one measure of behaviors in computer systems, this method is the one that is utilized. Univariate IDS examines each metric to search for anomalies on its own¹⁰⁰.

Multivariate: It is predicated on the relationships between two or more measures to comprehend the connections between the variables. If experimental results demonstrate that improved categorization may be achieved from combinations of correlated measurements rather than by analyzing each step independently, then this model could be helpful. Investigate using a multivariate quality control strategy to recognize incursions by constructing a comprehensive profile of typical actions across time. The fact that it is challenging to estimate distributions for high-dimensional data presents the most significant obstacle for multivariate statistical identifiers¹⁰¹.

Time series model: A time series is an observational string compiled throughout a specific period. Suppose the likelihood of a new observation occurring is extremely low. In that case, the word is considered abnormal, and proposed an approach for detecting network faults by analyzing the sudden variation seen in time series data as part of their research¹⁰². This method's practicability was demonstrated and confirmed using simulated experiments—strategies based on acquired knowledge. The term "expert system method" can also refer to

this collection of approaches. For this strategy, you must construct a knowledge base that accurately depicts the profile of valid traffic. In this standard profile, all actions deviating from the norm are considered incursions. The typical profile model, in contrast to the other types of AIDS, is typically developed based on human understanding in the form of a set of criteria that attempt to characterize normal system activity¹⁰³.

Because the system knows all the usual activities, the primary advantage of using knowledge-based procedures is the power to limit the number of false-positive alerts generated. However, in a computing environment that is constantly evolving, this kind of IDS needs frequent updates on knowledge for the expected normal behavior. This process takes a lot of time because it is tough to obtain information about all usual behaviors¹⁰⁴.

A finite state machine, often known as an FSM, is a type of computation model that can both represent and control the flow of execution. It is possible to build a model of an intrusion detection system by applying this model to the field of intrusion detection. In most cases, the model is depicted using states, transitions, and activities as its constituent parts. A state examines the data from the history. For example, any variations in the input are noticed, and a transition takes place based on the variances that have been discovered. Any behavior deviating from the FSM, representing legitimate system behavior, is considered an assault. An FSM can represent legitimate system behavior¹⁰⁵.

Description Language: Description language defines the syntax of rules that can specify the qualities of an attack that have already been set. Languages for describing software, such as N-grammars and UML, could be used to construct rules.

Expert System: An expert system comprises several rules describing attacks. Typically, a knowledge engineer will manually develop the powers of an expert system in conjunction with a domain expert. This process takes place within an expert system¹⁰⁶.

In intrusion detection systems (IDS), signature analysis was one of the first methods used. It is based on the straightforward concept of string matching. In the string-matching process, every word of an incoming packet is examined alongside its unique signature. An alarm will go out if a signature is found to match. If this is not the case, the information included within the traffic is compared to the following signature in the signature database¹⁰⁷.

Machine learning models consist of a collection of rules, strategies, or sophisticated "transfer functions" that can be utilized to identify or anticipate behavior and locate noteworthy data patterns¹⁰⁸. AIDS research has significantly benefited from the application of machine learning techniques. Clustering, neural networks, association rules, decision trees, genetic algorithms, and nearest-neighbor methods are only some algorithms and techniques utilized to discover knowledge from intrusion datasets. There is some historical study that has investigated the usage of a variety of ways to construct AIDS. The performance of two feature selection algorithms utilizing Bayesian networks (BN) and Classification Regression Trees (CRC) was analyzed, and the researchers merged these two methods to get a greater level of accuracy¹⁰⁹.

The principles of machine learning have been applied to the development of various anti-AIDS drugs. The creation of enhanced IDS that require less human knowledge and have improved accuracy is the target of the use of techniques from the field of machine learning. Throughout the past few years, there has been a rise in the number of applications that use machine learning strategies. Identifying patterns and constructing intrusion detection systems

based on the dataset are two of the primary focuses of IDS research driven by machine learning. In machine learning, there are often two categories of approaches: supervised and unsupervised. In the intrusion detection system, supervised learning was used¹¹⁰.

IDS methods that are based on supervised learning detect intrusions by making use of training data that has been labeled. Training and examination are the typical components of the two stages that make up a supervised learning approach. During the training phase, relevant characteristics and classes are uncovered, and the algorithm subsequently gains knowledge from the data samples it has been given. Each record in supervised learning IDS is a pair consisting of a network or host data source and an associated output value (i.e., label), denoting either an intrusion or a normal condition. This label can be either an intrusion or a normal state. The next step is to utilize feature selection to get rid of features that aren't necessary. A supervised learning technique is then used to train a classifier to learn the inherent relationship between the input data and the labeled output value. The training data for selected features are used in conjunction with the supervised learning technique. In the body of research that has been done, numerous approaches to supervised learning have been investigated, each of which has both positive and negative aspects. During the testing phase, the trained model is utilized to classify the unknown data as either belonging to the standard class or the intrusion class¹¹¹.

After this, the resultant classifier is transformed into a model that, given a collection of feature values, can make an educated guess as to the category to which the input data may be assigned. Several different metrics are used to Validate the effectiveness of a classifier in terms of its ability to predict an item's category correctly. There are many different approaches to classification, including Decision Trees, Rule-based Systems, Neural Networks, Support Vector Machines, Naive Bayes, and k K-nearest-neighbor methods. Each strategy

employs a different learning approach to construct a classification model. Nevertheless, a way of classification that is appropriate¹¹².

2.3.3 Detection Systems That Are Based on Intrusion Detection Systems

Monitoring and analyzing active networks and traffic on those networks is required for intrusion detection, which helps identify potential computer hacking attempts. The IDS system is a compilation of many approaches and techniques that can be utilized for this objective. Most intrusion detection systems (IDSs) generally come equipped with conventional security features. Gathering information from witnessed incidents is the first step in an IDS. It performs in-depth logging and compares the activities being performed with event-related data from various networks. Depending on the circumstances, the most critical component of an IDS is the detector, which uses multiple methodologies and related techniques. Additionally, there would be the capacity for mitigation. The procedure of determining an intrusion and taking preventative measures against it is called an Intrusion Detection and Prevention System in this context¹¹³.

2.3.4 A Strategy for the Detection of Anomalies

Various diverse patterns may be utilized in anomaly-based recognition techniques, including generative, discriminating, and hybrid structures. If typical and until behaviors are to be differentiated, clustering should be binary in situations when the assault is assigned to one sort of attack. Hierarchical structures can be further subdivided into multiple classes after they have been divided into various subclasses. Any studies that make use of hierarchical datasets have previously been carried out. In many cases, the detector can support real-time applications, whether it is operating online or offline. Numerous research articles on intrusion detection methods have been published throughout the past few years. Most of these studies

concentrate on detecting network intrusions by applying various machine learning and deep learning strategies¹¹⁴.

2.4 Artificial Intelligence (AI)

Understanding the similarities and differences among Artificial intelligence (AI), Machine learning (ML), and Deep Learning (DL) approaches that can be used to solve DF problems is essential¹¹⁵.

2.4.1 Artificial intelligence (AI)

This is the study of giving machines intelligence, or the capacity to mimic human performance in certain situations (such as object processing, natural language processing, etc.). AI is not the same as learning algorithms or smart devices. Artificial intelligence (AI) is any tool that mimics or enhances human intelligence. There has been a rise in criminal actions because of the increasing prevalence of AI technology^{116,117}. Intelligent bots are computer programs that mimic the capabilities of a human brain. Intelligent agents are employed to engage with the world around them. Using this method, the agent can recognize its surroundings through its sensors, at which point it can alter the environment's state.

For the functions within the agents to carry out these effects, the most crucial parts of AI systems are how instruments are utilized to gather information and how it is translated to controllers. The end game of artificial intelligence is to program a machine to perform human-like tasks. Only learning algorithms, whose purpose is to simulate the human brain's educational process, are necessary to accomplish this task. The future of AI technology seems bright, and it provides significant advantages today. However, these tools cannot be avoided when committing heinous acts that endanger society at large^{118,119}.

2.4.2 Machine Learning (ML)

ML is an AI technique that makes use of a self-learning system. It is used for artificial intelligence reasons like emulating human behavior, but it also requires minimizing the period and effort necessary to do even the simplest activities. ML can be conceived as a system that learns without being explicitly programmed. As a result, ML describes a system constantly adapting and deciding what to do next based on data rather than code. Machine learning (ML) is an emerging technique in numerous fields (including business, research, and technology) to give computers new capabilities. Medical research, robotics, and engineering are just a few sectors that can benefit from the plethora of independent strategies based on ML^{120,121}.

Machine learning can be divided into two primary categories: supervised and unsupervised. The success of supervised learning depends on the availability of relevant information in labeled data. Classification is the most common task in supervised learning (and is also used most frequently in IDS), yet manually classifying data is expensive and time-consuming. Classification is the most common task in supervised learning. As a consequence, the primary obstacle hindering the progress of supervised learning is an insufficient amount of labeled data. On the other hand, unsupervised learning involves gleaning useful feature information from data that has not been tagged. This makes it significantly more straightforward to gather data for training purposes. However, the detection performance of unsupervised learning methods is typically lower than supervised learning methods. This is because unsupervised learning methods lack direct human supervision¹²².

2.4.2.1 Intensive Education

When used to learning algorithms, transfer learning enables the system to discover patterns within sequences, such as in picture detection. The system identifies the Angles, framework,

object category, and item entity before the item is placed. The two primary ways to define ML forensics are the inductive approach and the logical approach^{123,124}:

2.4.2.2 Inductive Learning

The basis of inductive reasoning is an overarching comprehension of particular data. The information is novel and does not help to preserve the truth. In other words, fresh information can disprove previously held beliefs. No theory has sufficient evidence to back it up. Numerous objectives can be pursued in this domain, such as learning about broad categories using only a few examples. This "experience" is what we'll be drawing on as examples. The idea is based on the concept of comparing and contrasting various instances. Specifically, inductive learning techniques are used^{125,126}.

2.4.2.3 Learning through Deduction

Knowledge is gleaned through established procedures using deductive reasoning, often called logic. In deductive reasoning, one uses pre-existing knowledge to conclude. This is not brand-new info. However, this is taken for granted in the introductory context. As a matter of mathematical logic, new information cannot overturn the previous facts^{127,128}.

2.4.3 Machine Learning's Importance in Intrusion Investigations

ML is an AI offshoot used for processing large datasets, analyzing them for signs of criminality and risk, and dividing them into subsets for further investigation. The absence of intelligence in a system renders it incapable of genuine learning. Capitalizing on AI's potential as a leading security solution capacity, DFI via ML is the next trend. ML behavioral analytics application in medicine, industry, marketing, and security has increased recently. ML Detect uses online and cloud resources and wireless and wired networks to uncover illegal activity. The mission of the MLF is to enhance its performance by acquiring fresh information and developing a systematic framework for arranging that information¹²⁹.

Artificial intelligence (AI)--based ML algorithms can scan large datasets for risk patterns, data segmentation, and criminal activity. With the help of ML algorithms, researchers can query massive, geographically scattered data sets housed in social and wired networks and the web or cloud. ML systems employ pattern recognition software to predict behavior from vast amounts of data. ML algorithms attempt to glean useful information from past events to predict future actions better. To predict when and where crimes will happen in the future based on past data, ML systems can identify patterns of criminal behavior. Break-ins, money laundering, and intrusion attempts are just some of the potentially dangerous events that could be in the collected data collection. Data from servers, unsure devices, wireless devices, and the Internet, among other sources, can be formally analyzed for visualization, link connotation, subdivision, and prediction of illegal behaviors, allowing this goal to be achieved. Cyber dangers today are increasingly complex and challenging to identify with the methods used in the past. There has been an increase in the complexity of attacks as hackers have developed more sophisticated techniques. Every single one of these assaults would slip through the cracks of the system administrator's radar. The industry suffers from slow incident incidence speed, more significant delays in detecting and preventing cyber threats, and the requirement for more innovative skills to eliminate these cyber dangers because human knowledge and competencies have some parameters. Therefore, it may be beneficial to construct more sophisticated machine learning models to avert and guard against these virtual risks. Many automated software exists today to help humans carry out complicated and scientific activities. These computerized tools will need AI and ML methods in the future^{130,131}.

2.4.4 Approaches to Supervised Machine Learning

Using supervised ML algorithms in real-time forensics has benefits, including the capacity to anticipate future events based on historical data. So, let's peek at several supervised ML

techniques that might work here. Examples of machine learning procedures include the SVM, kNN, and Random Forests (RF)¹³².

2.4.4.1 k-Nearest Neighbour

Forensically gathered digital data can be used with kNN to help reveal previously unknown connections. Its non-parametric learning methods make it helpful in identifying similarities between dataset samples and classifying them accordingly. In most cases, the instances that originate from or are kept in memory are essential in defining the output of kNN. In addition, the vast majority of kNN neighbors must reach a verdict based on continuous variables. Specifically, the KNN employs the Manhattan, Euclidean, and Minkowski distance metrics. The technique measures the gap between samples using a K equivalent to the square root of the tuple's digits. Ascending order is then applied, making it simple to pick the closest neighbor. The nearest-points Euclidean distance, Manhattan distance, and the Minkowski distance function all designate the distance metric^{133,134}.

One of the methods for machine learning was called K-Nearest Neighbor, and it was used to categorize objects based on datasets that served as learning data. It was expected that whichever came closest to the actual thing would be used. K-Nearest Neighbor can remember feature vectors and learn to classify data because it is a machine-learning technique. In the classification stage, the same features were generated for the training and testing datasets (whose classification is unidentified). The distance between this new vector and the vector used for learning was calculated. When this occurs, we shall use the closest possible K value. The newly classified points are anticipated to be included in most of these categories. Euclidean Distance techniques will determine the distances between each nearby and faraway location. Using text-based categorization, we may calculate cosine similarity using Euclidean Distance. A more considerable Euclidean Distance value indicates that the training and test

datasets are geographically and statistically distinct. The lesser the Euclidean Distance score, the closer the test data is to the training data. In text classification, however, a more excellent cosine similarity value indicates a closer relationship between the test and learning data. In contrast, a lower value means less of a connection between the two sets of data. Here is how you solve for distances in Euclidean space¹³⁵:

$$Cap P1 \text{ dan } P2 = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Algorithm 2.1: KNN Algorithm

Input a large amount of data and a set of test samples that should be classified (data space has n dimension)

Output the estimated class of each test sample

1. Begin
 2. Divide the large amount of data to m separate clusters using k-means algorithm
 3. For each cluster $i=1 \dots m$
 4. Calculate the size of cluster (number of data samples)
 5. Calculate d_j^i ($j=1 \dots n$)
 6. For each test sample
 7. Choose the proper cluster of data by using Algorithm 1.
 8. Employ the KNN algorithm on the selected cluster to find the estimated class of the test sample
 9. End
-

KNN Algorithm^{136,137}

2.4.4.2 Support Vector Machine (SVM)

SVMs learn by classifying objects. SVM, derived from statistical training, may be efficiently used for the detailed investigation of computer information since it can generate a hyper-plane that maximizes class margins. SVM employs an approach that provides a training set to be the primary underlying nodes of a tree, which may subsequently be segmented utilizing user knowledge¹³⁸. SVM is a two-classification strategy that segments training datasets using hyperplanes. Separation maximizes the period (soft or hard) and turns it into a combinatorial

optimization challenge. If the learning algorithm is sequentially time-sharing, optimize the challenging frequency with a linear SVM; if it is linearly with an increased flow, enhance the soft frequency and choose the proper parameter with a multidimensional SVM; if it is incrementally non-timesharing, optimize the smooth frequency and choose the appropriate parameter with a multidimensional SVM. Here's a collection of SVMs.

2.4.4.2.1 Machine with Linear Separable Support Vectors

The core knowledge behind linear separable SVM is to find an appropriate partition hyperplane in a sample space training set denoted as M , separating the models of different groups. Linearly separable data models are those that a linear function can separate. So, what exactly is a linear function? If spatial dimensions are ignored, a linear function is noted as a straight line in two-dimensional space, a level in three-dimensional space, and so on; such a linear function is typically referred to as a hyperplane. We can see a simple example of two-dimensional space in a two-dimensional space. Positive classes are denoted by "O," whereas antagonistic classes are represented by "X." Samples are linearly separable; from a graphical standpoint, it is evident that not just this conventional line but also a plethora of other lines can be used to separate samples. Lines that can accurately partition data and have the biggest intervals correspond to the linear discrete support vector machine¹³⁹.

The staging point period must all be determined to get the biggest period. To illustrate the partition of the support vectors in the subspace, we use the linear model: where W is the hyperplane's standard vectors and b is its distance from its source. Assuming the hyperplane can appropriately identify the data for training¹⁴⁰: The greatest interval hypothesis is defined by the formula above. It denotes a positive sample that has been stated as a negative sample. In actuality, the rate of 1 is simply for the sake of control, and it could theoretically be any constant.

2.4.4.2.2 Nonlinear Support Vector Machines (NSVMs)

These are a form of multidimensional support. The basic idea besides variational support vector machines correlation is to utilize non - linear transformation to visualize information to a high-dimensional subspace (Hilbert space) and then perform linear interpolation in that space so that the stochastic challenge in the low-dimensional space is the same as the high-dimensional attribute control scheme of the regression model. This is the reliable process: The regression input variables method comprises the type of activation functions, the parameter specifications, the penalties component D, and the increased sensitivity coefficient. These specifications have a powerful connection with how well the correlational process is performed¹⁴¹.

The greatest learning procedure for classification is the SVM, which is a machine learning technology. It is the most used supervised machine learning method. The SVM is a classification and regression algorithm pattern that uses several different kernel functions. SVMs have primarily been utilized for binary classification. The goal is to identify a line or hyperplane that connects two classes and is as far away from the contiguous instances of each class as possible. The margin is a space that separates instances of various groups. Support vectors are the nearest instances to the hyperplane. The margin must be as large as feasible; this is necessary to improve the effectiveness of the classification of newly additional cases. The biggest margin's boundary function can be calculated as follows¹⁴²:

$$\text{Minimize } W(\alpha) \frac{1}{2} \sum_i \alpha_i = \frac{1}{2N} \sum_j \alpha_j = \frac{1}{2N} \sum_i \alpha_i \quad k(y_i, y_j) - \sum_i \alpha_i$$

$k(y_i, y_j)$ represents the kernel function of SVMs, where α is a vector of N variables, C is the soft border constraint, $C > 0$, and $C > 0$. In SVMs, kernel functions are used to categorize dataset instances into distinct categories. Here are the four primary types of kernel functions:

linear kernel: $k(y_i, y_j) = y_i^T y_j$;

polynomial kernel: $k(y_i, y_j) = (y_i^T y_j + r)^d, > 0;$

linear kernel: $k(y_i, y_j) = y_i^T y_j;$

polynomial kernel: $k(y_i, y_j) = (y_i^T y_j + r)^d, > 0;$

radial basis function (RBF) kernel: $k(y_i, y_j) = \exp(-\gamma \|y_i - y_j\|^2), > 0;$

sigmoid kernel: $y_i^T y_j = \tanh(\gamma y_i^T y_j + r)$

This technique has been used to identify infiltration in information security. The fundamental benefit of utilizing an SVM as an IDS is its speediness, which allows it to identify intrusions in real-time. Due to their respectable rounded nature and capability to overcome the curse of dimensionality, SVMs have become a popular tool for identifying anomalous infiltration. They can oversimplify well-using kernel methods even in high-dimensional spaces under small-training-sample situations, they're also beneficial for discovering universal minimal definite risk utilizing operational risk drop. Not relying on typical investigational risks like neural networks and learning a variety of patterns that can increase, the SVM can decide the suitable configuration parameters. When a new pattern emerges during classification, support vector machines can dynamically update training patterns¹⁴³.

Algorithm 2.2: SVM Algorithm

Input: $D=[X,Y]$; X (array of input with m features), Y (array of class labels)

$Y=\text{array}(C)$ // Class label

Output: Find the performance of the system

function train_svm(X,Y , number_of_runs)

initialize: learning_rate=Math.random();

for learning_rate in number of runs

 error=0;

for i in X

if $(Y[i] * (X[i]*w)) < 1$ **then**

 update : $w=w + \text{learning_rate} * ((X[i]*Y[i])*(-2*(1/\text{number of runs})*w))$

else

```

        update:  $w = w - \text{learning\_rate} * (-2 * (1 / \text{number\_of\_runs}) * w)$ 
    end if
end
end
end

```

SVM Algorithm¹⁴⁴

2.4.5 Naive Bayes Algorithm

Can anticipate class event probabilities. The authors stress that Naive Bayes is self-contained and doesn't need additional features. Naive Bayes classifies based on standard deviation and mean. It also permits training and testing data grouping. This permits Naive Bayes to handle outlier data and irrelevant properties¹⁴⁵. The Naive Bayes algorithm is a basic probabilistic classification algorithm that obtains its probability value by calculating frequency and value combinations from the linked collection. All attributes are assumed to be independent in this algorithm. To establish the class of the data to be Validated, the Nave Bayes classification procedure requires various hints or directives.

Algorithm 2.3: Naïve Bayes Algorithm

Input:

$D = \{d_1, d_2, \dots, d_n\}$ //set of n data items.

k // Number of desired cluster

Output:

A set of k clusters.

Step:

1. Arbitrarily choose k data-items from D as
initial centroids;

2. Repeat

Assign each item d_i to the cluster which
has the closest centroid;

Calculate new mean for each cluster;

Until convergence criteria is met.

Naïve Bayes Algorithm¹⁴⁶

2.4.6 Decision Tree

A decision tree (DT) uses a series of decisions to conduct classification and regression on a dataset (rules). The model has nodes, divisions, and trees. Each node signifies a characteristic. The subdivision symbolizes a decision or rule, while each tree indicates a conceivable consequence or class name. To minimize over-fitting, the DT method automatically selects ideal tree properties and prunes superfluous branches. Most DTs are C4.5, CART, and ID3. Random Forest (RF) and XGBoost60 use several decision trees.

Decision trees (DT) are a popular data mining classification tool. ID3, C4.5, C5, and CART are decision tree algorithms. DT splits data iteratively to construct a tree to increase prediction accuracy. Using mathematical techniques (such as the Gini index, information gain, and Chi-squared test), they partition the input data into two or more subsections. This technique was repetitive at the respective tree nodes until the entire tree was constructed. Because of their hierarchical structure, decision trees are simple to build and understand. They can simulate complex functions. There are two phases to the decision tree classifier: growth and build. ii. The pruning stage. The tree is formed in the first step by recursively partitioning the training dataset using the best criterion until all or nearly all of the data in each partition has the same class label. At this point, data overfitting is a possibility. During the pruning phase, the number of consecutive branches is reduced so that the tree may effectively generalize the model. Pruning entails traversing the decision tree from the bottom-up or top-down, deleting noisy and outlier nodes, and improving specific criteria in the process. Popular pruning procedures include cost-complexity, decreased error, minimum error, minimum descriptive length, minimum message length, and crucial value. The pruning phase of the decision tree resolves the problem of overfitting the data. As a consequence, during the pruning stage, classification accuracy improves. Only the fully developed tree is accessible during the pruning phase. In the growth phase, numerous passes over the training data are required.

When compared to the time complexity required to develop the decision tree, pruning the decision tree takes less time¹⁴⁷.

Algorithm 2.4: Decision Tree Algorithm

GenDecTree(Sample S, Features F)

Steps:

1. **If** stopping_condition(S, F) = true then
 - a. Leaf = createNode()
 - b. leafLabel = classify(s)
 - c. **return** leaf
 2. root = createNode()
 3. root.test_condition = findBestSpilt(S, F)
 4. V = {v / v a possible outcome of root.test_condition}
 5. **Foreach** value v ∈ V:
 - a. S_v = {s / root.test condition(s) = v and s ∈ S }:
 - b. Child = TreeGrowth (S_v, F);
 - c. Add child as descent of root and label the edge {root → child} as v
 6. **return** root
-

Decision Tree Algorithm¹⁴⁸

2.4.7 Random Forests

The live forensic analysis must use a random forest classifier for supervised learning due to the abundance of data from networked contexts. Random forest lets a single classifier give a machine-learning model for parameterization and over-fitting. This uses independently tested ensemble decision trees. This allows randomizing a dataset. Given class c, the random forest can estimate the sample's probability of c. Comparable to the decision tree classifier, a random forest classifier was constructed with 300 estimators and various hyperparameters¹⁴⁹. A random forest is a collection of random Decision Tree classifiers that combine the predictions of individual trees to make predictions. Incorporating randomization into the decision tree construction method can be done in a variety of ways. A random forest

can be used to create predictions about nominal or numeric target properties (Regression).

One of the most effective forecasting models is random forests.

Algorithm 2.5: Random Forest Algorithm

```

1: for  $k \leftarrow 1$  to  $M$  do
2:  $D_k \leftarrow \text{Bootstrap}(D)$ 
3:  $h_k \leftarrow \text{Decision Tree With Random Feature Selection}(D_k, F)$ 
4: end for
5: return  $U_{h_k}$ 

```

Random Forest Algorithm¹⁵⁰

2.4.8 AdaBoost Classifier

A random forest of three estimators was created using an AdaBoost classifier and its base estimator. This classifier had 200 estimators in total. "SAMME" is an algorithm¹⁷¹. AdaBoost is an extension of the boosting procedure. This ensemble technique has been used in a variety of applications, with speech recognition, Diseases detection, and image-based categorization, among others. This ensemble technique has not been extensively applied for noise and intrusion classification. For the overall classification, the majority of the investigators have advanced only robust discrete learners. AdaBoost is capable of generating a hypothesis from a list of candidate labels. The prediction error for the weak hypothesis would be smaller than 0.5 during training. The goal of the distribution is to pick the 'hard' training data for the following iteration and sample back. Through weighted common voting of the classes, AdaBoost develops a set of hypotheses and ensemble from individual hypothesis prediction¹⁵¹.

Algorithm 2.6: Adaboost Algorithm

Given: $(x_1; y_1) \dots, (x_m; y_m)$, $x_i \in X$ $y_i \in Y = \{1, -1\}$.

Initialize $D_1(i) = 1/m$.

For $t=1..T$:

1. Train weak classifier using distribution D_t .
2. Get weak hypothesis $h_t : X \rightarrow \{1, -1\}$ with error $\epsilon_t = E_{D_t}(h_t(x_i) \neq y_i)$

3. Choose $\alpha_t = 1/2 \log \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right)$

4. Update:

if instance i is correctly classified

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} = \begin{cases} e^{-\alpha_t} & \text{if instance } i \text{ is correctly classified} \\ e^{\alpha_t} & \text{if instance } i \text{ is not correctly classified} \end{cases}$$

if instance i is not correctly classified

where Z_t is a normalization factor (chosen so that $\sum_{i=1}^m D_{t+1}(i) = 1$).

Output the final hypothesis: $H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(x) \right)$.

Adaboost Algorithm ¹⁵²

2.4.9 Logistic Regression

Logistic regression gives probability estimates, not predictions. This paradigm suits binary classification. In this model, event probability is a linear function of input features. Logistic regression determines p for a linear combination of independent parameters to determine class labels. To avoid overfitting, fit a logistic regression model using the non-linear conjugate gradient approach with an LR penalty on weights.¹⁵³

Algorithm 2.7: Logistic Regression Algorithm

Input Dataset $D = D^S \cup D^T$ with feature representation x_i

Output Dataset $D = D^S \cup D^T$ with feature representation f_i

1. Train the LLP classifier with D
2. Predict the pseudo-label values for target data
3. Noisy labels dataset: $P = \{x_i, y_i\}_{i=1}^{ns+nr}$
4. **for** $r=1$ to R **do**
5. Training subset: $P^r = \{m \text{ random samples}\}$
6. Train projection functions $f^r(\cdot)$ with P^r using eqn. (4)
7. **end**
8. **for** $i=1$ to $n_s + n_r$ **do**
9. **for** $r=1$ to R **do**
10. Obtain projection vector: $V_i^r = f^r(x_i)$
11. **end**
12. $v_i = ((V_i^1)', \dots, (V_i^R)')'$

13. End

Logistic Regression Algorithm¹⁵⁴

2.4.10 Convolutional Neural Network

An input layer, a convolutional layer, a pooling layer, a completely associated layer, and an output layer make up the CNN structure¹⁵⁵. The number of convolution layers and pooling layers in different CNN structures varies. Adopt the CNN's input distinctive feature plot of layer I ($M_0 = X$) of layer i. Finally, we can express the convolution process as $M_i = f(M_{i-1} W_i + b_i)$.

Here, the i-layer convolution kernel weight vector is W_i , the convolution operation symbol is " $*$ ", the i-layer offset vector is b_i , and the activation function is " f " (x). Different window values are specified in the convolutional layer, which then excerpts diverse feature data from the data matrix M_{i-1} and excerpts diverse features M_i from the information using various convolution kernels. To reduce the number of constraints in the neural network as a whole, the convolution kernel uses the "parameter sharing" principle, which means it uses the same weights and offsets for each convolution. After the convolutional layer, the feature map is typically sampled using a variety of selection instructions by the assembling layer. If M_i is the input to the pooling layer and M_{i+1} is the output of the pooling layer, then the pooling layer can be stated as $M_{i+1} = \text{subsampling (or something similar) } (M_i)$. The window region's maximum or mean value is typically selected as the sampling criterion's selection threshold. As a result of the pooling layer, redundant features are less likely to affect the model.

Convolutional neural networks (CNN) are widely used deep learning (DL) models because they combine CNN to extract the image's features with fully connected neural networks (FCNs). Using a pooling layer on top of the convolution layer, it is possible to lessen the magnitude of the feature data while still preserving the image's I/O and spatial information.

The following equation is used to process an image: $L_0 = L - K + 2PS + 1$. It's important to note that L is an abbreviation for the image's length. It is the kernel size (K) as well as the level of dimension (P) at both ends that fill in the zero. The stride of the kernel on a convolution layer is denoted by the letter S . A convolutional layer's number and performance aren't always proportional to the number of convolutional layers it contains. It is necessary to experiment with various designs and learn from them to regulate the optimum amount of convolutional layers and in what way it performs well. If an image is grayscale or RGB, the number of convolutional layers and kernel size will determine how many weights will be used to design a hidden layer in a convolution layer¹⁵⁶.

Algorithm 2.8: CNN Algorithm

Initialize w_k using randomly draw n weights

$i = 1$

while $i \leq \text{maximum_iter}$ **do**

 Do feedforward pass

 Update weights forever through Adam and backpropagation errors

 Set the $\mu(L)$ = mean of central points of all filters of layer L

 Update the weights of layer L using the $w_k^{(L)}(x, y) = ((w_k^{(L)} * \mu^{(L)}) / \sum w_k^{(L)})$

 Update the weights of central points of & filters of layer L using the $w_k^{(L)}(c_x, c_y) = -\mu^{(L)}$

$i = i + 1$

If training accuracy converges **Then**

 Exit

End.

CNN Algorithm¹⁵⁷

2.4.11 Recurrent Neural Networks (RNN)

To model sequence data, RNNs build on the traditional feed-forward neural network's capabilities. An RNN comprises an input, hidden, and output components, where the memory elements are the hidden units. Each RNN unit decides based on its existing input and the

previous input's output. Speech dispensation, human action identification, writing prediction, and semantic understanding are just a few of the many applications of RNN. When it comes to supervised classification and feature extraction in an IDS, an RNN can be utilized. Short-term memory is a problem for RNNs because they can only handle sequences of a certain length. There are several RNN variants, such as LSTM and GRU, that can address these issues¹⁵⁸. Elman's simple structure for RNNs is one of many possible structures. An RNN is a variation of a feed-forward neural network that has been improved upon. There is a one-way flow of data from the input layer to the hidden layer and then to the output layer in the traditional neural network model. The RNN, on the other hand, is unique in that it can remember the information it has already processed ($t+1, t+1, \dots$). As a result, the output of a hidden layer includes not only that from an upper layer but also that from the same layer at the previous time point¹⁵⁹.

A complete network can be constructed by unfolding an RNN over time. One way to represent the input and output sets is to use the abbreviations " $\{o_0, o_1, \dots, o_{t-1}, o_t, o_{t+1}, \dots\}$ " respectively. It is possible to denote the input set as " $\{i_0, i_1, \dots, i_{t-1}, i_t, i_{t+1}, \dots\}$ " and the output set as " $\{h_0, h_1, \dots, h_{t-1}, h_t, h_{t+1}, \dots\}$ " for each unseen layer. There are weight matrices from the input to the hidden layer, from the hidden layer to the output layer, and within each of those layers. The most crucial task is completed by the hidden units in an RNN. However, the hidden layer's nodes are self-connected and interconnected to fully exchange information with the input layer's information¹⁶⁰.

2.5 Preparation Stages for Data

This section offers tips for formulating live networking information for machine learning investigations¹⁶¹.

Feature Engineering: In this context select subsets of exclusive features from ECO's live data. This provides significant or selective features for classification. Feature engineering identifies many classification-helping traits. When completely executed in an IoT setting to create a digital criminal tool, the suggested framework would leverage Machine learning algorithms to automate feature identification to eliminate duplicates through feature selection and deletion. This study increases awareness of the relevance of supervised procedures during live forensics¹⁶².

Feature Selection: To increase the accuracy with which a specified classified model can predict the result of the real-time investigation, feature selection makes an effort to discriminatively pick from a subset of features X, a set of Y features. The ultimate purpose of feature selection in this setting is to facilitate the learning model's capability to excerpt relevant and significant subsets from the feature distribution and then map those subsets to the original class distribution for use in forensic analysis¹⁶³.

Feature Elimination: Feature elimination highlights the fact that each dataset can be utilized to ascertain whether or not the limited features found in the dataset are beneficial/pertinent. The terms "relevant" and "helpful" above indicate whether or not certain characteristics can be disabled. It's important to keep in mind that several things could play a role in its extinction. For instance, in the ECO context, new devices are frequently configured and re-configured on the go, resulting in large amounts of data. Furthermore, shifts in technical and technological components could slow down the ability to determine which features need to be eliminated. Because it's possible to get rid of something that could come in handy, you should be cautious while performing this procedure. Following a forensic examination, numerous methods of forensic profiling of opponents were revealed when performing feature elimination. Since shifts in behavior are so common, this is the case¹⁶⁴.

Feature Normalization: Particularly since data taken from different sources typically consist of different features, it is important to normalize the features of a given data set during real-time forensics so that each feature can be normalized individually based on some given range. Assigning different weights to these extracted data can also be done using different feature distance metrics, such as Euclidean distances, Manhattan distances, and so on. With this feature in mind, normalization takes on greater significance as it can equalize the variation in various features via the computation of similarity. In most cases, this method requires applying a statistical transformation to the feature components to ensure that the resulting values accurately represent the features, or at least provide a good approximation of them. A uniform random variable, ranks, or scaling method could be used to modify the characteristics of the IoT data¹⁶⁵.

Representation of Characteristics: One of the defining features of a dynamic environment like the ECO is the ability to combine data from several sources into a single operational framework. Consequently, when several features from different sources are combined throughout a specific analytical range, it is required to provide an inimitable setup for the instances in the individual feature vectors. For more flexibility, a feature space data structure was outlined to handle disparate data types. The feature representation step is where the data format will be determined to carry out such a procedure¹⁶⁶.

Dimensionality Reduction: This is the procedure of transforming a large amount of raw information from a high into a low-dimensional space where the innovative input dimension retains the majority of the fresh information's attributes¹⁶⁷. For data visualization, high-dimensionality and dimensionality reduction approaches are frequently used. These strategies, however, can be used in machine learning to help explain the information and better build a

prediction model. Furthermore, the curse of dimensionality states that adding more input features makes predictive modeling more difficult. As a result, the more features there are, the more difficult it is to picture the training set and subsequently work on it. In high-dimensional spaces can also be inconvenient for a variety of reasons: raw data is frequently sparse, and most of these features are associated and thus redundant; as a result, processing the data is typically computationally limited. Dimensionality reduction methods are useful in this situation. Unassuming models that oversimplify effectually, with few input data variables, are chosen. Hence, reducing the number of input features is frequently beneficial¹⁶⁸. Dimensionality reduction has two fundamental components: Feature selection is the procedure of utilizing scoring or statistical methods to find and select relevant features from input variables. Feature extraction is the process of extracting new data with fewer dimensions from high-dimensional input data¹⁶⁹.

Uniform Manifold Approximation and Projection (UMAP)

¹⁷⁰ devised UMAP, a unique multiple-knowledge approach for dimension lessening. One of the limitations of UMAP is that it can only be used for dimensionality reduction, which means it is not suitable for all types of data analysis tasks. For example, it cannot be used for classification or regression tasks, since it does not provide a predictive model. The UMAP approach, with its superior efficiency and lack of computational constraints on embedding dimensions, is thought to maintain more of the global structure. UMAP is one of the most rapid manifold learning applications accessible. It is divided into two stages¹⁹¹: Using the binary exploration and a fixed number of nearest neighbors, create a high-dimensional graph and calculate the bandwidth of the exponential probability. Stochastic Gradient Descent (SGD) is utilized to improve the calculation rapidity by optimizing the low-dimensional representation. The exponential probability distribution in high magnitudes is calculated by UMAP as $p_{i|j} = \frac{e^{-d(x_i, x_j)}}{\sum_k e^{-d(x_i, x_k)}}$, where p is the distance between each i th data point and its

first nearest neighbor. Furthermore, UMAP employs the number of closest neighbors k as follows: $k = 2\sum ip_{ij}$

Also, $p_{ij} = p_{i|j} + p_{j|i} - p_{i|j}p_{j|i}$

Algorithm 2.9: UMAP Algorithm

```
function UMAP(X, n, d, min-dist, n-epochs)
# Construct the relevant weighted graph
For all  $x \in X$  do
    fs-set[]  $\leftarrow$  LocalFuzzySimplicialSET(X, x, n)
top-rep  $\leftarrow U_{x \in X}^{\text{fs-set}[x]}$  # We recommend the probabilistic t-conorm
# Perform optimization of the graph layout
Y  $\leftarrow$  SPECTRALEMBEDDING(top-rep, d)
Y  $\leftarrow$  OPTIMIZEEMBEDDING(top-rep, Y', min-dist, n-epochs)
return Y
```

UMAP Algorithm¹⁷¹

2.5.1 Particle Swarm Optimization (PSO)

An example of an optimization algorithm is the particle swarm algorithm. PSO, or Particle Swarm Optimization, is an algorithm for optimizing a problem that was created by Eberhart and Kennedy in 1995. The algorithm is based on a workable answer by iteratively searching and rating candidates against some quality metric. The PSO method is called metaheuristic since it is used to quickly and efficiently find solutions that, if not optimal, are at least "good enough." Since PSO is often used to compute a variable's worth, its search space is all the potential values for that variable. The PSO method generates a swarm of particles, each of which is characterized by its present location in the search space, its best-observed position thus far, the best global position gotten by some particle in the swarm, and its relative velocity. After being set in motion, the particles randomly explore the search space, with each new

position being Validated by an objective function to determine whether or not it should be utilized to update the local best or the swarm's best solution^{172,173}.

Algorithm 2.10: PSO Algorithm

1 Initialize population

2 for r= 1: maximum generation

3 for i = 1 : population size

4 if $f(x_{i,d}(t)) < f(p_i(t))$ **then** $p_i(t) = x_{i,d}(t)$

5 $f(p_g(t)) = \min_t (f(p_t(t)))$

6 end

7 for d =1 : dimension

8 $v_{i,d}(t+1) = wv_{i,d}(t) + c_1r_1(p_t - x_{i,d}(t)) + c_2r_2(p_g - x_{i,d}(t))$

9 $x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1)$

10 if $v_{i,d}(t+1) > v_{\max}$ **then** $v_{i,d}(t+1) = v_{\max}$

11 else if $v_{i,d}(t+1) > v_{\min}$ **then** $v_{i,d}(t+1) = v_{\min}$

12 end

13 if $x_{i,d}(t+1) > x_{\max}$ **then** $x_{i,d}(t+1) = x_{\max}$

14 else if $x_{i,d}(t+1) < x_{\min}$ **then** $x_{i,d}(t+1) = x_{\min}$

15 end

16 end

17 end

18 end

PSO Algorithm¹⁷⁴

2.5.2 Chi-Square (Chi²)

The Chi-Squared test (also known as Pearson's Chi-Squared test) is a statistical theory test used to determine whether or not two variables are independent. Chi-square analysis is also applied to feature selection, where the chi-square statistics are calculated for each feature and the target variable to determine if there is a correlation between the two. It is possible to get rid of a feature variable if it does not affect the dependent variable. If the two variables are intertwined, the feature variable takes on a critical role. To that end, the following formula is used to determine Chi-Squared statistics: $\chi^2_c = \sum (O_i - E_i)^2 / E_i$, where O represents the

observed value and E represents the anticipated value. If the two numbers are unrelated, then O and E will be near, but if they are related, then the Chi-squared value will be large¹⁷⁵.

2.5.3 Mutual Information (MI)

If you want to know how much you can learn about one random variable by observing another, one way to accomplish it is by calculating the mutual information between the two. What's more, in probability and information theory, the mutual information (MI) of two random variables is a measure of the mutual dependency between the two variables. Because of this, a high mutual information value indicates a significant decrease in uncertainty, whereas a low value indicates only a modest reduction. It is agreed that two random variables are unrelated if and only if their mutual information is zero. Furthermore, Shannon and Weaver define $I(X; Y)$ as the mutual information between two variables X| AND Y¹⁷⁶.

$$I(X; Y) = \sum_{x, y} P_{XY}(x, y) \log \frac{P_{XY}(x, y)}{P_X(x)P_Y(y)} = E P_{XY} \log \frac{P_{XY}}{P_X P_Y}$$

The marginals are $P_X(x)$ and $P_Y(y)$: $P_X(x) = \sum_y P_{XY}(x, y)$

Algorithm 2.11: Mutual Information Algorithm¹⁷⁷

Input: Random variable $X = \{x_1, \dots, x_M\}$, number of bins R, B-Spline order k

Output: W (X)

Foreach i, $1 \leq i \leq M$ **do**

 Calculate the normalized variable z_i , ($i=1, \dots, M$) using Eqn. (9);

Foreach j, $1 \leq j \leq R$ **do**

 Calculate the weighting coefficient $B_j, k(z_i)$ using Eqn. (7) and (8) with the normalized value

Z_i ;

end

end

Output $W(X)_{i,j} = B_j, k(z_i)$

Mutual Information Algorithm¹⁷⁷

2.5.4 Genetic Algorithm

Genetic algorithms (GA) are a class of mathematical models based on the ideas of natural selection and evolution. This heuristic is frequently employed to find practical answers to optimization and search challenges. Select problems are transformed utilizing chromosome-like data, and chromosomes are developed using selection factors, recombination, and mutations. Holland proposed the genetic algorithm (GA), which is a metaheuristic algorithm. This algorithm mimics the process of natural selection, in which fitter individuals have a better chance of surviving in a competitive environment than others. It's a worldwide search technique that evolves with each passing generation. In GA, the first step involves creating a random population of the chromosome, each of which represents a distinct strategy for approaching the issue. The fitness value of a chromosome indicates how well it fits the challenge at work. Using evolutionary operators like selection, crossover, and mutation, the population progresses toward improved fitness with each generation. This method is repeated until a solution is found or until the maximum number of iterations has been reached.

A random set of chromosomes is frequently used to start the genetic algorithm procedure. These chromosomes reflect a problem that needs to be solved. The positions of each chromosome are encoded in numbers, letters, or bits depending on the features of an issue. Genes are the names for these places, which are randomly modified within a range during development. During the evolution stage, a population is a collection of chromosomes. Each chromosome's GOODNESS is calculated using the Evaluation function. During the Evaluation, there are two main things to consider. Crossover is a technique for simulating a species' metamorphosis and natural reproduction. Chromosomal selection favors the fittest chromosomes for survival and synthesis. In the form of partial functions or by using multiple chromosome coding schemes specifically tailored for a specific task, genetic algorithms may represent nearly any type of limitation¹⁷⁸.

Algorithm 2.12: Genetic Algorithm

Input: instance Π ,

size α of population,

rate β of elitism,

rate γ of mutation,

number δ of iterations

Output: solution X

// Initialization

1 generate α feasible solutions randomly;

2 save them in the population Pop;

// Loop until the terminal condition

3 for $i=1$ to α do

//Elitism based selection

4 number of elitism $ne = \alpha \cdot \beta$;

5 select the best ne solutions in Pop and save them in Pop₁;

// Crossover

6 number of crossover $nc = (\alpha - ne)/2$;

7 for $j=1$ to nc do

8 randomly select two solutions X_A and X_B from Pop;

9 generate X_C and X_D by one-point crossover to X_A and X_B ;

10 save X_C and X_D to Pop₂;

11 endfor

// Mutation

12 for $j=1$ to nc do

13 select a solution X_j from Pop₂;

14 mutate each bit of X_j under the rate γ and generate a new solution X_j' ;

15 if X_j' is unfeasible

16 update X_j with a feasible solution by repairing X_j' ;

17 endif

18 update X_j with X_j' in Pop₂;

19 endfor

// Updating

20 update Pop = Pop₁ + Pop₂;

```

21 endfor
// Returning the best solution
22 return the best solution X in Pop ;

```

Genetic Algorithm¹⁷⁹

2.5.4.1 Levy Flight Algorithm

The Levy Flight algorithm is a random search optimization technique inspired by the behaviour of animals during foraging. It introduces heavy-tailed Levy flights into the search space to explore and exploit solutions. Levy flights are random steps that exhibit occasional long jumps, allowing the algorithm to explore a large portion of the search space efficiently.

2.13 Pseudocode for the Levy Flight Algorithm

1. Initialize the population of candidate solutions.
 2. Set the step size, s , and the scaling factor, ' λ '.
 3. Validate the fitness of each candidate solution.
 4. Repeat until a termination criterion is met:
 5. For each candidate solution:
 6. Generate a random step size, ' r ', from a Levy distribution.
 7. Generate a random step direction, ' d ', from a uniform distribution.
 8. Update the candidate solution's position: ' $\text{new_position} = \text{current_position} + \lambda * (s * r * d)$ '.
 9. Validate the fitness of the new candidate solution.
 10. If the new candidate solution is better, replace the current solution with the new one.
- Return the best candidate solution found.

Pseudocode for the Levy Flight Algorithm¹³³

2.5.4.2 Other Enhancements for Evolutionary Algorithms

Differential Evolution (DE): DE enhances evolutionary algorithms by introducing differential mutation and crossover operations. It creates new candidate solutions by adding the weighted difference between two randomly selected solutions to a third solution. DE improves exploration and exploitation capabilities.

2.14 Pseudocode for Differential Evolution (DE)

1. Initialize the population of candidate solutions.
 2. Set the mutation rate, 'F', and the crossover rate, 'CR'.
 3. Validate the fitness of each candidate solution.
 4. Repeat until a termination criterion is met:
 5. For each candidate solution:
 6. Select three distinct solutions randomly.
 7. Generate a mutant vector: '**mutant_vector** = **solution1** + **F** * (**solution2** - **solution3**)'.
 8. Perform binomial crossover between the mutant vector and the current solution:
'**new_solution** = **crossover**(**mutant_vector**, **current_solution**, **CR**)'.
 9. Validate the fitness of the new solution.
 10. If the new solution is better, replace the current solution with the new one.
- Return the best candidate solution found.

Pseudocode for Differential Evolution (DE)¹⁴⁵

2.5.4.3 Particle Swarm Optimization (PSO)

PSO simulates the behavior of a swarm of particles in a search space. Each particle represents a candidate solution, and the swarm collaboratively explores the space by adjusting its position based on its previous best position and the best position found by any particle in the swarm.

2.15 Pseudocode for Particle Swarm Optimization (PSO)

1. Initialize the population of particles with random positions and velocities.
2. Set the cognitive and social learning rates, '**c1 and c2**,' and the inertia weight, **w**.
3. Validate the fitness of each particle.
4. Initialize the best positions of each particle to their current positions.
5. Initialize the best global position to the particle's position with the best fitness.
6. Repeat until a termination criterion is met:
7. For each particle:
8. Update the velocity: '**new_velocity** = **w** * **old_velocity** + **c1** * **rand()** * (**best_particle_position** - **current_position**) + **c2** * **rand()** * (**best_global_position** - **current_position**)'.
9. Update the position: '**new_position** = **current_position** + **new_velocity**'.
10. Validate the fitness of the new position.

11. If the new position is better, update the particle's best position.
 12. If the new position is better than the best global position, update the best global position.
- Return the best global position found.

Pseudocode for Particle Swarm Optimization (PSO)¹⁶⁵

These are just a few examples of techniques that can enhance evolutionary algorithms. There are many other methods and variations available based on specific problem domains and requirements.

2.5.5 Principal Components Analysis

PCA is the most widely used and most popular technique for reducing dimensions because of its flexibility and reversible approach to calculation. PCA reduces the number of variables in high-dimensional space, which is computationally expensive and projects the appropriate features into a low-dimensional subspace. When there are "m" observations and "n" measurements, the matrix (A) can be represented as follows¹⁸⁰:

Equation $A_{m \times n} = [A_1, A_2, A_3, \dots, A_n]$ shows that each column is a vector.

Equation $\mu = \frac{1}{n} \sum_{i=1}^n A_{ij}$ is used to calculate the average mean () of each vector's values.

The deviation is a metric used to measure the difference between a vector and its mean. Using the equation $\varphi = A_i - \mu$, the deviation is calculated (3)

Measurement of the correlation between the two variables can be done using covariance. Any value other than zero indicates no correlation between the two variables, while any matter other than positive suggests that the data are negatively correlated. The covariance matrix tells us about the data's spread. Using the formula $B_{m \times n} = 1/n-1 \sum_{i=1}^n \varphi (X_i - t_i) (X_i + t)^t$, the covariance matrix is built over the square matrix with the number of classes as the dimension. Singular Value Decomposition is commonly used to calculate Eigen-values and Eigenvectors for PCA on the covariance matrix (SVD). The covariance matrix 'B' has m eigenvalue

eigenvector pairs. The highest P eigenvalues are then selected, as they are more helpful in classifying data, while the lower $m-P$ values are discarded because they are of low significance.

Algorithm 2.16: PCA

Input: Training Network Set $D = \{b1, c1, c2, c13, f1, f2, \dots, f9\}$

Output: New training data matrix $D' = \{v1, v2, \dots, vd'\}$

1. $X' = D - \text{mean}$ Remove the average value
 2. Find the covariance $X' X'^T$ of data matrix X'
 3. Finding the eigenvalues λ and eigenvectors of the covariance matrix $X' X'^T$
 4. Arrange feature three values $\lambda_1 \geq \lambda_3 \geq \dots \geq \lambda_d$
 5. $\omega = (\omega_1, \omega_2, \dots, \omega_d)$
 6. $\Omega' = (k\omega_1, k\omega_2, k\omega_3, \dots, k\omega_n, \dots, \omega_d')(n \leq d')$
 7. $D' = \{v1, v2, \dots, vd'\} = X * \omega'$
-

PCA Pseudocode ¹⁸²

2.5.6 Deep Learning

Different scholars have come up with various definitions for deep learning. Despite this, they all share characteristics and terms, some of which include "complex architectural data model," "unsupervised machine learning," "learning multiple layers," and "nonlinear data transformations." All of these essential concepts are intimately connected to neural networks and pattern recognition in one way or another. Deep learning generally does not require pre-selected features, which allows it to circumvent the difficulty of feature selection. This is possible because deep learning automatically extracts significant features from raw input to solve the issue. Deep learning models typically include several processing layers with many abstraction levels. This gives the system the ability to learn a variety of data attributes. The presence of numerous tiers provides the network with the ability to recall individual characteristics. Deep learning is an approach that has gained widespread recognition as one that generates promising results in various fields, including image recognition, speech recognition, face recognition, language translation, topic classification, sentiment analysis,

signal processing, and natural language processing, to name a few. Face recognition is one of the most challenging tasks in computer vision, as it requires the ability to distinguish between. In addition, numerous distinct architectures for deep learning have been developed, such as DBNs, RNNs, and CNNs, amongst others¹⁸³.

A convolutional neural network (CNN) is a deep learning model based on the organization of the animal visual cortex. CNNs are used to process data such as photographs. Its primary purpose is to learn the spatial hierarchies of features in low- to high-level patterns autonomously and adaptively. It has been demonstrated to be effective in various tasks, such as face recognition, object recognition, and the detection of traffic signs, most notably in robots and self-driving automobiles. Keeping the number of parameters in an ANN to a minimum is the most critical aspect of a CNN. As a result, programmers and researchers have been driven to concentrate their efforts on developing larger models capable of tackling complex problems, which is impossible with traditional ANNs¹⁸⁴.

Learning the essential characteristics of the data it is fed is the primary objective of a CNN. The first few layers of this procedure consist of a collection of convolutional feature extractors that are then sent through a series of learnable filters. The data that is being input is passed through a sliding window that is formed by the filters that are being applied. In this scenario, the overlapping distance is referred to as the stride, and the maps that result from the process are called feature maps. Each CNN layer comprises convolutional kernels and contributes to the production of various feature maps. In the feature map of the layer above, adjacent neuron regions are linked to one another to form a neuron. The kernel must be distributed uniformly over the input's various spatial positions to generate a feature map. Following the construction of the convolutional and pooling layers, one or more fully connected layers may be deployed to complete the classification. The presence of numerous

tiers provides the network with the ability to recall individual characteristics. Deep learning is an approach that has gained widespread recognition as one that generates promising results in various fields, including image recognition, speech recognition, face recognition, language translation, topic classification, sentiment analysis, signal processing, and natural language processing, to name a few. Face recognition is one of the most challenging tasks in computer vision, as it requires the ability to distinguish between. In addition, numerous distinct architectures for deep learning have been developed, such as DBNs, RNNs, and CNNs, amongst others¹⁸⁵.

A convolutional neural network (CNN) is a deep learning model based on the organization of the animal visual cortex. CNNs are used to process data such as photographs. Its primary purpose is to learn the spatial hierarchies of features in low- to high-level patterns autonomously and adaptively. It has been demonstrated to be effective in various tasks, such as face recognition, object recognition, and the detection of traffic signs, most notably in robots and self-driving automobiles. Keeping the number of parameters in an ANN to a minimum is the most critical aspect of a CNN. As a result, programmers and researchers have been driven to concentrate their efforts on developing larger models capable of tackling complex problems, which is impossible with traditional ANNs¹⁸⁶.

Learning the essential characteristics of the data it is fed is the primary objective of a CNN. The first few layers of this procedure consist of a collection of convolutional feature extractors that are then sent through a series of learnable filters. The data that is being input is passed through a sliding window that is formed by the filters that are being applied. In this scenario, the overlapping distance is referred to as the stride, and the maps that result from the process are called feature maps. Each CNN layer comprises convolutional kernels and contributes to the production of various feature maps. In the feature map of the layer above,

adjacent neuron regions are linked to one another to form a neuron. The kernel must be distributed uniformly over the input's various spatial positions to generate a feature map.

Following the construction of the convolutional and pooling layers, one or more fully connected layers may be deployed to complete the classification¹⁸⁷. The concept of convolution processes inspired the creation of the term CNN. The convolutional layer is a vital component because it is the part of the CNN architecture responsible for carrying out feature extraction. A convolution operation and an activation function are examples of the linear and nonlinear processes typically included in this layer. In addition, convolution procedures use learning characteristics to apply a filter, also known as a kernel, to the data to preserve the spatial relationship between the pixels. When referring to the data that is generated as a result of a convolution process, the terms "activation map," "convolved feature," and "feature map" are frequently utilized. The values of the kernel are automatically adjusted whenever a CNN is training to learn the characteristics of a new input efficiently. The size of the feature map is determined by the dimensional values of depth (which refers to the number of filters that are applied), stride (which refers to the number of pixels by which the filter slides), and zero padding (which relates to padding data with zeros at the border)¹⁸⁸.

Consequently, a more practical approach is required, and it is possible that the previously described method is not as effective as recognizing local portions in the image rather than the full image. Regarding the convolutional layer, the hidden neurons in the following layer can only accept inputs from the area of the layer that comes before it that corresponds to them (the neighbor regions).

Layers That Are Pooling

In most cases, the down-sampling operation that is carried out by the pooling layer results in a reduction of the in-plane dimensionality of the feature maps. This, in turn, results in the introduction of translation invariance to any slight movements or distortions. Additionally, the number of following learnable parameters is decreased as a result. It is interesting to note that none of the pooling layers have any parameters that can be learned. Despite this, filter size, stride, and padding are all considered hyperparameters, much like the convolutional layer. After the convolutional layer, many features still need to be accounted for, resulting in a highly complex network. Therefore, the pooling layer needs to be able to under-sample the features to simplify the network and prevent it from overfitting. Both the Maximum Pooling and the Average pooling methods used most frequently in CNN systems are the pooling methods¹⁸⁹. Following the convolutional layer is where you'll typically find the pooling layer. Its primary objective is to reduce the size of the feature matrix, increase training speed, and minimize parameters to circumvent problems associated with overfitting. The data sent into the pooling layer is first segmented into several separate areas according to size. In Max-Pooling, the value with the highest total in each region is selected to represent the region as a whole. In contrast, in Average Pooling, the deal with the highest mean total in each area is selected to represent the region as a whole¹⁹⁰.

Layers That Are Completely Linked

When referring to a network as "completely connected" (FC), it is assumed that every neuron in the layer below has a connection to a neuron in the layer above it. The output feature map for the final convolutional or pooling layer is typically flattened, presenting it as a one-dimensional set of numbers (or vectors). These numbers or vectors are connected to one or more fully connected layers, sometimes referred to as dense layers. In these layers, every input is connected to an output based on a learnable weight, and the consequences can be adjusted. Following the extraction of the features in the convolutional layers and the

subsequent downsampling of the features in the pooling layer, the features may be mapped utilizing the fully connected layers of the neural network. Because of this, it is possible to ascertain the final outputs of the network, which are the probabilities associated with each class. The completely connected layer is often followed by a SoftMax classifier (or another type of classifier), where a SoftMax classifier typically contains a total of n neurons, where n represents the number of classes. In other words, a fully connected layer is typically followed by a SoftMax classifier¹⁹¹.

Neural Networks With Recurrent Connections

Recurrent neural networks, often called RNNs, are more advanced feed-forward neural networks. Unlike feed-forward neural networks, RNNs can memorize data at each step for subsequent outputs. An RNN is a recurrent neural network, and its neurons have their outputs connected to the inputs of both other neurons and themselves. Because of this, RNNs can model data sequences and time series utilizing the memory that is included inside them. RNNs are typically used in IDSs to extract temporal correlations between security threats and harmful actions (also known as temporal features), whereas CNNs extract spatial characteristics. RNNs can be implemented in various architectures, one of which is the long short-term memory (LSTM) model, frequently utilized in intrusion detection systems (IDSs)¹⁹².

2.5.7 Chi-Square

Chi-square is a popular statistical analysis method used frequently for feature selection. It does this by determining the relative significance of each attribute about the result class. The value of the chi-square statistic is utilized to determine how dependent the features are on the result class. To put it another way, a feature with a high chi-square value indicates that the feature is acceptable for classification because it is more dependent on the result class^{193,194}.

2.5.8 ANOVA (Analysis of Variance)

The analysis of variance and covariance (ANOVA) is a technique for selecting univariate features that rank the features based on their variance score. The significance of a feature's influence on the response class is determined by its variance score. A low variance between the attributes of several courses indicates that the classification might be improved, whereas a high conflict suggests that the classification could be worse¹⁹⁵.

2.5.9 Principal Component Analysis (PCA)

The principal component analysis (PCA) is a well-known method that can reduce the amount of data. The principal component analysis (PCA) uses linear algebra to reduce the data's dimensionality while preserving the information's essential nature and practical qualities. When PCA is done for feature reduction, there is less amount of information that is lost. In addition to that, it has a lower sensitivity to noisy data¹⁹².

2.6 A Classification Model Based on Deep Neural Networks

Following the data reduction and cleaning processes, the deep neural network is utilized for classification purposes. A significant number of intrusion and anomaly-based applications make extensive use of deep neural networks. Input, hidden, and output layers are the components that make up a DNN model. During training, the DNN adjusts the settings to prevent classification errors. DNN models benefit from complex hidden layer structures, which allow them to be more accurate and versatile when dealing with massive datasets. Each layer adds its unique degree of complexity to all of the aspects. The presented deep neural network (DNN) model has three hidden layers, and the activation function of the hidden layer is the rectified linear unit (ReLU), while the activation function of the output layer is sigmoid¹⁷¹.

Because the output of multilayer neural networks known as "autoencoders" is comparable to the input and has a small number of reduced variances, these types of neural networks provide the same output as their inputs with a little reconstruction error. The autoencoder employs an unsupervised learning methodology to decode or reassemble the encoded work. When utilizing an autoencoder, data can have their dimensions decreased, features can be retrieved, images can be compressed, and noise can be minimized. To keep things as straightforward as possible, we will discuss the fundamental architecture of an autoencoder without going into further detail¹⁴⁵.

The encoder, the bottleneck, the decoder, and the reconstruction loss are the four primary elements of a general autoencoder. An encoder is used to do additional compression on the data coming from the input. This helps to lower the total amount of characteristics that the model needs to take into account. The layer of the input data that has the most compressed data and the fewest features is the one that is causing the bottleneck. A model can decode the encoded representation by utilizing a decoder, which enables the model to verify that the output and the input are identical. When analyzing the effectiveness of a decoder, the difference between the output of the decoder and the initial input is referred to as the "reconstruction loss," and this word describes the discrepancy. Backpropagation is utilized for training purposes and to reduce further the amount of data lost during reconstruction. The objective of AE is to minimize losses to their absolute minimum. The input value x can be compressed using the encoder function E to produce the desired result. The decoder D will work toward recreating the input as best it can. In this scenario, the reconstruction loss is represented by the difference between the encoded and decoded vectors¹⁹⁴. The approach known as mean squared error, or MSE, can be utilized to determine the amount of reconstruction loss.

Variational Autoencoders, often VAEs, can determine reconstruction loss by utilizing Kullback–Leibler (KL) divergence. The KL divergence is a measurement that compares the probability distributions of data already in the latent space and data projected into the latent space. This positive value represents the degree to which the two distributions are dissimilar. Denoising, variational, convolutional, and sparse autoencoders are examples of the many available available^{160,161}.

2.7 Deep Neural Network

Since biological neural network characteristics affect it, we suggested using an MLP model technique. Inputs that can be transmitted from one node to another utilizing a loop in the system are used to describe the multilayer perceptron (MLP), also known as a feedback neural network. To speak in terms more familiar with mathematics, each layer of the MLP model has a sizeable number of neurons or units. This model is comprised of at least three levels, each of which contains one or more hidden layers, and it also includes an output layer. A hyperparameter selection approach can be utilized to ascertain the number of hidden layers present in a model. Information can travel from one layer to the next thanks to neural connections between the layers. In mathematics, the MLP is defined as $O: \mathbb{R}^m \rightarrow \mathbb{R}^N$ where m refers to the size of the input vector, and N refers to the length of the output $O(x)$ vector, a function of x . In other words, the input vector's size determines the output vector's size. Each of the layers can be computed as follows: where the variable denotes the size of the input, and the nonlinear activation function is represented by the variable f , which can be either a sigmoid (with values in the range $[0, 1]$) or a tangent part. The sigmoid function is the more common of the two (values in the range $[-1, 1]$)¹¹².

An artificial neural network (ANN) is intended to function in a manner that is analogous to that of a human brain. An artificial neural network (ANN) has numerous hidden layers in

addition to an input and output layer. All of the connections between the units in neighboring layers have been made. Because an ANN is composed of many teams and theoretically capable of approximating any arbitrary function, it possesses a powerful ability to fit, particularly for nonlinear functions. Training ANNs take a lot of time because the structure of the model is so complicated. It is essential to highlight that the backpropagation technique used to train ANN models cannot be employed in preparing deep neural networks. Therefore, an ANN is considered part of the shallow models instead of the deep learning models¹⁰³.

In **Support Vector Machines (SVMs)**, the goal of the technique is to locate an n-dimensional feature space hyperplane with the maximum separation margin. Because the separation hyperplane is only controlled by a limited number of support vectors, SVMs can produce satisfying results even when working with minimal training sets. SVMs, on the other hand, are extremely sensitive to noise in the vicinity of the hyperplane. SVMs are capable of doing well when it comes to solving linear problems. Kernel functions are typically applied when dealing with nonlinear data. A kernel function converts the initial space to a new area, allowing the linear and nonlinear data to be extracted from the original data set. Tricks using the kernel are commonplace in support vector machines (SVMs) and other machine learning techniques¹²⁴.

K -Nearest Neighbor (KNN). The manifold hypothesis serves as the inspiration for the fundamental concept behind KNN. If most of a sample's neighbors belong to the same class, then there is a high probability that the sample belongs to the same class. Consequently, the categorization result is only relevant to the top k nearest neighbors. The value of the parameter k profoundly impacts the effectiveness of KNN models. When k is less, the model becomes more complicated, which also increases the likelihood that it will be overfit. On the

other hand, the opposite is true: the larger k is, the simpler the model is, and the lower its capacity to fit¹⁶⁵.

Naive Bayes. The concept of attribute independence is the foundation for the Naive Bayes method based on conditional probability. The Naive Bayes classifier works backward from each sample to get the conditional probabilities for each possible class. The sample was placed in the category with the highest possible probability. The Tree of Decisions The data is categorized based on criteria using the decision tree method. The structure of the model is similar to a tree, which enables one to interpret it. The decision tree method can automatically eliminate features that are irrelevant or redundant. The learning process involves picking features to include, creating trees, and then pruning those trees. During the process of training a decision tree model, the algorithm chooses the features that are the most appropriate at a time and then creates child nodes from the parent node. The decision tree is an elementary kind of classification. Multiple decision trees are the building blocks of more complex algorithms like the random forest and the extreme gradient boosting (XGBoost), both examples.

Clustering. The principle of similarity is the foundation for clustering, which entails placing remarkably similar data inside the same clusters and placing less similar data within other clusters. Clustering is a subcategory of unsupervised learning and should not be confused with classification. Clustering algorithms do not require prior knowledge or labeled data; as a result, the requirements for the data set are not very stringent. However, to detect assaults using clustering techniques, it is essential to refer to external information⁹⁶.

The K-means algorithm is a common type of clustering algorithm. In this algorithm, K refers to the total number of clusters, while the mean refers to the average of the attributes. The K-

means algorithm employs the criterion of distance as its primary similarity measure. The smaller the gap that separates two data objects, the higher the probability that those data objects will be included in the same cluster. The K-means technique works very well with linear data. However, it does not produce optimum results when used with nonconvex data. The K-means algorithm is also sensitive to the initialization circumstance and the parameter K. As a consequence, numerous iterations of tests need to be carried out to determine the appropriate value for the parameter.

Ensembles and Hybrid Musical Structures Every single classification criterion has advantages and disadvantages. Combining several different weak classifiers into one robust classifier is a natural technique that might be used. Ensemble approaches involve training several classifiers, which are then used to cast votes to determine the final findings. The development of hybrid approaches involves the creation of numerous phases, with each stage employing a different classification model. An increasing number of academics are beginning to focus their attention on ensemble and hybrid classifiers. This is because ensemble and hybrid classifiers typically perform better than single classifiers—the essential considerations concern which classifiers to combine and how this is done.

2.8 Datasets

2.8.1 CICIDS2017 Dataset

The network intrusion detection system needs updated data to train the model to work effectively in the case of an anonymous intrusion. DARPA98/99's KDDCUP99 open-access dataset restricts the addition of new attacks. Real-world attacks (in the form of CSVs) are included in the CICIDS 2017 dataset¹⁰⁶. CIC Flow Meter's network traffic analysis results in labeled flows based on timestamps, protocols, ports, source IPs, and destination IPs. There is a wide range of attack types not covered by previous datasets, such as brute force, DoS, and

DDoS attacks, as well as web-based attack types like brute force, DoS and DDoS intrusion, and Bot and Scan. In addition, it has 84 features divided into four categories. Complete network formation, complete traffic, a labeled dataset, and complete interaction are among the 11 criteria that must be met to produce a reliable dataset. Complete capture and readily available protocols, Attack heterogeneity, a wide range of attack features, and metadata.

Background traffic in the CICIDS2017 dataset is based on 25 users' abstracted network activity. Precise protocols like HTTP, FTP, HTTPS, SSH, and email protocols were found in the user profiles. The developers used statistical metrics like maximum, minimum, mean, and standard deviation to group the network events into the following features: The packet size distribution, packets flowed in a flow, The payload's weight, The distribution of the protocols' response times, Payload patterns to be aware of. As a result, CICIDS2017 covers a wide range of common attack scenarios. Brute Force Attacks, HeartBleed Attacks, Botnet, DoS Attacks, Distributed DoS (DDoS) Attacks, Web Attacks, and Infiltration Attacks are just a few of the assaults that can be used against a computer system. The authors have made the dataset available in Packet Capture (PCAP) format packet payloads. Machine and deep learning can be helpful in the corresponding profiles and labeled flows in CSV files¹²⁷.

2.8.2 NSL/KDD

IDS system's efficacy and practicality have been tested using the NSL-KDD dataset. This dataset is a condensed variety of the original KDD 99 dataset created by¹⁰⁸. One class attribute and 41 features make up the NSL-KDD dataset. Twenty-one attacks are included in the training dataset, while 37 attacks are included in the test dataset. DoS, U2R, Probe, and R2L are the types of attacks.

2.8.3 DARPA (Lincoln Laboratory 1998,1999)

Network security analysis was the primary purpose of this dataset¹⁰⁹. Researchers slammed this dataset because of the artificially inserted attacks and non-harmful traffic. These activities include e-mail transmission and receipt, web browsing, file transmission and receipt via FTP, remote computer access via telnet and work, Internet Relay Chat (IRC) transmission and receipt, and router monitoring via SNMP. This data collection contains information about various attacks, including denial-of-service, guess-password, DOS attack, remote FTP, syn flood, Nmap, and rootkit. This dataset does not understand actual network traffic. It features abnormalities like a lack of false positives, which prevents an Evaluation of IDSs on information systems with existing attacks and network equipment. It is also outdated. Furthermore, there are no actual attack data records in the database.

2.8.4 DEFCON-8

Port scanning and buffer run-off attacks were included in DEFCON-8, created in 2000, while DEFCON-10 was created in 2002 using bad packets, organizational treat, and FTP attacks by telnet¹¹⁰. As a result, network traffic generated during the "Capture the Flag (CTF)" competition consists of intrusive activity rather than regular background traffic. To test alert correlation methods, this dataset is used to gather data.

2.8.5 Center for Applied Internet Data Analysis (2002/2016)

CAIDA is comprised of the following three distinct varieties of datasets: Members of CAIDA, DARPA, NSF, and DHS, as well as Cisco, collect a variety of data kinds via an OC48 link in San Jose; CAIDA DDOS attack dataset, which contains one hour's worth of DDoS attack traffic divided into 5-minute pcap files; 3) CAIDA Internet trace 2016, which contains inactive network tracing from CAIDA's equinix-Chicago monitor on High-speed Internet connection. Most of CAIDA's datasets are related to a single instance or operation, and all the

data concerning the message, protocols, and destinations have been anonymized. This dataset suffers from several deficiencies, which render it unsuitable for use as a measurement sample¹¹¹.

2.8.6 Lawrence Berkeley National Laboratory (LBNL)

There are full-header network traffic records of the total network traffic at the medium-sized site of Lawrence Berkeley National Laboratory (LBNL) in 2004/2005. This dataset has been heavily annotated without a payload to remove any identifying information¹¹².

2.8.7 CDX

Using the CDX dataset, one can generate a modern-day labeled dataset from the network warfare competitions represented in the CDX dataset¹¹³. Attackers employ Nikto, Nessus, and WebScarab for reconnaissance and attack automation. "Benign traffic" comprises a network, electronic mail, and DNS lookups. IDS alert rules can be tested on CDX, but the lack of traffic variety and volume makes it ineffective.

2.8.8 Kyoto 2009

The Kyoto (Kyoto University - 2009): No manual labeling or anonymization process was used to create this dataset, but it has a limited view of the network traffic because only honeypot attacks can be viewed¹¹⁴. This dataset provides ten extra features for NIDS investigation and Evaluation, including IDS, malware, and Ashula detection. Regular traffic has been reproduced frequently during attacks to lessen alarms, although only DNS and mail traffic data are produced, not real-world regular traffic.

2.8.9 Twente

Dutch for "Twente" Netflow collected data from a honeypot network with OpenSSH, Apache, and Proftpd on port 113. Auth/ident, ICMP, and IRC side-effect traffic are neither benign nor

malicious. Unknown warning traffic is also present. This genuinely labeled dataset lacks assaults, though.

2.8.10 UMASS (University of Massachusetts - 2011)

Network packets and some wireless application traces are included in the UMASS (University of Massachusetts - 2011) dataset¹¹⁵. This was generated using a single TCP-based transfer attack situation. The dataset lacks traffic and attack variation, making it unsuitable for IDS and IPS testing.

2.8.11 ISCX2012

ISCX2012 (UNB, 2012) uses a dynamic approach to create relevant and realistic IDS Evaluation datasets¹¹⁶. It has two parts. First, the Alpha profile used multi-stage attempts to stream the aberrant dataset. Beta profile provided accurate network traffic and related noise. Real traces are Validated to generate traffic profiles for HTTP, SMTP, SSH, IMAP, POP3, and FTP. This approach generates datasets with entire packet payloads and profiles. This dataset doesn't include HTTPS traces because it accounts for 70% of network traffic. Simulated attacks aren't based on real-world facts.

2.8.12 ADFA

The ADFA dataset was supported by a web server, a Database server, remote access, and an FTP server, all provided by Apache, MySQL, and Tikiwiki. This dataset includes FTP, SSH, and Java-based Meterpreter attack vectors, adding a new Superuser, Linux Meterpreter payload, and C100 Web shell. Each training and evaluation vector has ten assaults. Because this dataset lacks assault diversity and variation, some attack behaviors are challenging to identify from non-adversaries.

2.8.13 AWS: CSE-CIC-IDS2018

There is a systematic approach in the CSE-CIC-IDS2018 dataset to generate datasets that contain detailed descriptions of intrusions and abstractions of the distribution models for applications and protocols¹¹⁷. Using these profiles to generate network events, either by agents or humans, is possible. We can use the generated profiles with a wide range of network protocols and topologies because of their abstract nature. Profiles can be combined to create custom datasets.

2.8.14 2019 Dataset of the CIC-DDoS2019

Common DDoS attacks that are harmless and up-to-date are included in CICDDoS2019 (PCAPs)^{118,119}. This study includes CICFlowMeter-V3 network traffic analysis results labeled by timestamp, basis and endpoint IPs, source and terminus ports, procedures, and assaults (CSV files).

2.8.15 MQTT-IOT-IDS2020

Datasets include the normal. pcap and Sparta files and scan A and scan sU files for brute force scanning¹²⁰. It's possible to record one scenario at a time, such as the regular operation, the Sparta SSH brute force, the aggressive scan, the MQTT brute force, and the UDP scan. The attack pcap files include all of the normal operations in the background. We need to know the attacker's IP address to identify the attacker. CSV files with the same pcap file names contain basic packet features extracted from the pcap files. A few examples of these attributes are flags, length, and various MQTT message parameters. After that, the features that can be extracted in only one direction or both directions are extracted. Some features (denoted by an asterisk *) have two values for bidirectional flows: one for forwarding flow and the other for regressive flow. The prefix "fwd_" for forward and "bwd_" for backward is used to record and distinguish the two features¹²¹.

2.9 Related Works

Word embedding models were used for intrusion detection in network traffic¹²². Today's businesses need network traffic analysis to prevent cyberattacks. Machine learning can detect hazardous network session tendencies. They can't be used on raw network traffic data. Active research focuses on designing and implementing feature extraction strategies to transform network traffic raw data into a form machine learning algorithms can assess. The suggested method extracts semantic features from contextualized raw data. Experiments on three datasets showed that a feature extraction strategy based on word embedding models can improve the classification performance of classic machine learning algorithms used in intrusion detection. This study shows word embedding models can accurately detect intrusions. Word embedding models require longer to compute than simpler algorithms, but they provide higher accuracy for identifying sophisticated attacks.

It was suggested that feature extraction be used for machine learning-based intrusion detection in IoT networks¹²³. The amount of IoT network security breaches proves the unreliability of contemporary NIDS (NIDS). Due to network disruptions and data loss, researchers are working to improve NIDS systems. An examination of related papers found that most researchers used unproven combinations of FR and ML on NIDS datasets to enhance categorization. Feature collections, attack types, and network design vary amongst these datasets. This project aims to see if these tactics work with different datasets. DFF, CNN, RNN, DT, LR, and Naive Bayes are ML models (NB). Using three benchmark datasets, UNSW-NB15, ToN-IoT, and CSE-CIC-IDS2018, the detection accuracy of three FE approaches, PCA, AE, and LDA, is investigated. Despite the frequent use of PCA and AE, the correct number of retrieved dimensions is ignored. No single FE technique or machine learning model provides the optimum results across all datasets. LDA decreases the

performance of ML models on two datasets. The variance tests LDA and PCA extracted dimensions. The datasets employed significantly impact the performance of the methodologies used, and a universal (benchmark) feature set is essential for continued growth and success in this discipline.

For intrusion detection, a novel feature extraction algorithm based on Siamese convolutional neural networks was developed¹²⁴. Intrusion detection systems can identify potential security risks or malicious attacks that are made to steal information or alter network protocols (IDS). The unpredictable and convoluted nature of cyber-attacks has led to the development of cutting-edge intelligence techniques, which have resulted in the creation of robust and automated intrusion detection systems (IDS) that rely on the most recent advances in machine learning (ML) and deep learning (DL). The vast bulk of the effort that has been published has been put towards constructing intricate ML/DL architectures that take a brute-force approach to optimize their detection capabilities. On the other hand, only a few researchers have concentrated their efforts on identifying or extracting user-friendly risk indicators that security experts may easily apply. Many articles have been written about the various ways of dimensionality reduction; yet, to accurately detect attacks, a tremendous number of selected attributes are still required, which humans cannot immediately or intuitively comprehend. This work adds to the transformation of the accessible data obtained by IDS into a single risk signal that is both easily understood and can be acted upon. This transformation aims to boost user confidence and comprehension of data without compromising its accuracy. To accomplish this, the following components were added to a novel feature extraction pipeline: I a novel modality transformation mechanism for converting feature vectors to images (Vec2im); (ii) a dimensionality reduction module that employs Siamese convolutional neural networks to reduce the input data dimensionality to a 1-d feature space; and (iii) a fuzzy allocation scheme that converts raw data into fuzzy class memberships. This was done to

accomplish the On the NSL-KDD dataset; a comprehensive comparison study showed that the proposed method was superior to several well-known feature selection (FS) and extraction techniques regarding detection accuracy, dimensionality reduction performance, and execution time. The proposed method achieved an accuracy of 86.64 percent using only one feature. While the generated images might be used and integrated as a visual analytics tool in existing IDS, security professionals could use the output of the suggested feature extraction process as a signal of malicious activity.

The Deep Belief Network was used to suggest an optimization method for an intrusion detection classification model¹²⁵. The network's quick growth and popularity have resulted in several security issues. A standard method for protecting a network, intrusion detection is becoming increasingly prevalent. Deep Belief Networks, often known as DBNs, are well-known deep learning models that perform admirably in classification tasks and are frequently utilized in intrusion detection. On the other hand, the majority of what makes up the network topology of DBN is decided by practice. This study presents a new joint optimization strategy for optimizing the network topology of DBN to address the challenge of maximizing the effectiveness of a DBN-based intrusion detection classification model (DBN-IDS). They started by developing an algorithm called PSO (Particle Swarm Optimization) based on an adjustable inertia weight and a learning component. In the second step of the process, we optimize the PSO by using the clustering, foraging, and other properties of fish swarms to find the initial optimization solution. After that, they used the initial optimization solution to search for the global optimization solution by utilizing genetic operators with self-adjusting crossover probability and mutation probability to optimize the PSO. This allowed them to seek the global optimization solution more effectively. In the final step, the network topology of the intrusion detection classification model is constructed with the help of the global optimization solution that was produced by the joint optimization technique that was

previously explained. Compared to other DBN-IDS optimization algorithms, their algorithm reduces average detection time by at least 24.69 percent while increasing average training time by 6.9 percent. Compared to the tested classification algorithms, their DBN-IDS improves average classification accuracy by at least 1.3 percent and up to 14.80 percent in the 5-category classification, proven to be an efficient DBN-IDS optimization method. The experimental results show that this is an efficient method for DBN-IDS optimization.

The Butterfly Optimization Algorithm was proposed for improving the accuracy of Network Intrusion Detection Systems in Medical IoT Systems¹²⁶. Network intrusion detection systems monitor traffic in a medical IoT system to identify any abnormal behaviors that may be occurring. Machine learning and artificial intelligence (AI) methods are frequently used in building intrusion detection systems. This prevents assaults on medical Internet of Things (IoT) devices. In this article, an artificial neural network is utilized to analyze the activity of a medical IoT system to identify any unusual patterns. The detection accuracy is greatly influenced by the features input into the artificial neural network. Because it has such a significant bearing on the overall learning process, determining the essential and differentiating features of network traffic is a topic that is both vital and difficult to understand. The butterfly optimization algorithm, a meta-heuristic optimization algorithm, is utilized in the strategy suggested to select the most valuable characteristics for the learning process in an artificial neural network. The fact that the butterfly optimization algorithm achieved an accuracy of 93.27 percent suggests that it can identify distinguishing characteristics of the data associated with network traffic. The decision tree, support vector machine, and ant colony optimization methods proposed in prior studies were all surpassed by the recommended approach for the identical job.

It was proposed to use a Deep Recurrent Neural Network to optimize a Border Collie Cat for an intrusion detection system in a healthcare IoT network¹²⁷. Attacks are a significant concern

for the applications and communication networks of the Internet of Things (IoT). Unidentified intruders can hurt the system's availability for end users and lead to an increase in the number of data breaches and instances of identity theft. As a consequence of this, it is essential to be able to identify attacks within IoT systems to guarantee proper defense and security. The Border Collie Cat Optimization-based Deep Recurrent Neural Network is presented as a method for identifying infiltration in IoT networks within the context of this study. The method known as Border Collie Cat Optimization is developed by fusing two different optimization strategies: Border Collie Optimization and Cat Swarm Optimization. During the authentication process, the hashing and encryption functions are utilized to perform the initial authentication of the communications. Following the completion of the authentication process, the communication phase is used to set up connectivity between the user and the server. This allows the Internet of Things device to participate in data movement throughout the network. The suggested Border Collie Optimization approach is employed during the training process of the Deep Recurrent Neural Network classifier, which is then used to detect intruders in the IoT network. The proposed method surpassed the competition regarding detection rate, sensitivity, specificity, and accuracy with values of 0.9375, 0.9539, 0.8791, and 0.9263, respectively.

The research was carried out on Intrusion Detection for IoT Using an Improved Genetic Algorithm and a Deep Belief Network¹²⁸. As a result of the proliferation of the Internet of Things (IoT), the security of the network layer is taking on an increasingly vital role. Traditional intrusion detection systems are useless in the complex Internet context of the IoT. In the case of a deep learning intrusion detection technique, a neural network structure may have good detection accuracy for one type of attack but poor detection accuracy for others. This might happen because different types of attacks target different parts of the network. As a consequence of this, it is essential to develop a model that is self-adaptive and able to

change the topology of the network according to the type of attack. This research presents an updated genetic algorithm (GA) and a deep belief network-based intrusion detection model developed for this research (DBN). The appropriate number of hidden layers and the number of neurons in each layer are adaptively created through numerous iterations of the GA. The result is an intrusion detection model based on the DBN with a high detection rate and a compact structure. The intrusion detection model was created in response to various forms of attacks. Ultimately, simulations using the NSL-KDD dataset tested the model and methods. The findings of the experiments imply that combining an improved version of an existing intrusion detection model with DBN can increase the number of successful identifications of intrusion attacks while simultaneously reducing the complexity of the neural network architecture.

It was proposed to develop an intrusion detection system for the Internet of Things based on blockchain and multi-agent systems¹²⁹. Network security has emerged as a primary concern in recent years due to the proliferation of the Internet of Things (IoT) technology. When applied to an IoT network, typical intrusion detection systems have limitations that arise from the network's complexity and resource limits. The primary topics of this work are the creation, implementation, and testing of a hybrid intrusion detection system that uses a multi-agent system, blockchain, and deep learning techniques. The system can be broken down into data gathering, administration, analysis, and reaction. The NSL-KDD dataset, used for knowledge discovery and data mining at the National Security Laboratory, is utilized in the testing of the system. The results of this study indicate that deep learning strategies are beneficial when spotting attacks on the transport layer. According to the experiment's findings, deep learning strategies appear to be appropriate for use in intrusion detection in an Internet of Things (IoT) network environment.

A Novel Intrusion Detection Approach for IoT Environments Using Machine Learning Ensemble was proposed¹⁴⁰. Because of the numerous fields in which it can be utilized, the Internet of Things (IoT) has recently gained much attention. Another reason for the growing significance of the Internet of Things is that it uses an adaptable and scalable architecture. The vast and varied applications of the Internet of Things in recent years have drawn the attention of cybercriminals. They take advantage of the vulnerabilities generated by the lack of rigorous and uniform security measures present in the open-source IoT framework. These vulnerabilities have the effect of discouraging current and potential stakeholders. The authors present a binary classifier strategy that was developed from a machine learning ensemble method to filter and dump undesirable traffic. This prevents hostile actors from accessing the IoT network and its peripherals. A binary classifier is trained to use pre-processed recorded data packets using the gradient boosting machine (GBM) ensemble method to identify anomalies and shield Internet of Things networks from zero-day attacks. The robust class performance measures of the model resulted in an accuracy of 98.27 percent, a precision of 96.40 percent, and a recall of 95.70 percent. The simulation results reveal that the proposed method is successful against cyberattacks, making it suitable for essential applications, including the Internet of Things (IoT).

It was proposed to design and develop a deep learning-based model for anomaly detection in IoT networks¹⁴¹. As the number of Internet of Things (IoT) devices continues to grow, dishonest individuals have access to a larger attack surface from which they can launch potentially more damaging cyberattacks. As a direct result, the security industry has witnessed an exponential rise in cyberattacks. The perpetrators of cyber-attacks frequently utilize novel and creative strategies to carry them out, which is one of the reasons why many of these attacks have been effective in attaining their malicious goals. An anomaly-based IDS (Intrusion Detection System) employs the methods of machine learning to identify and

categorize attacks on IoT networks. Traditional machine-learning algorithms become ineffective when confronted with unanticipated network technology and many different infiltration strategies. In various study fields, deep learning strategies have proved that they can effectively discover abnormalities. Convolutional neural networks are a good solution for anomaly detection and classification because of their ability to automatically categorize primary qualities in input data and their effectiveness in completing speedier calculations. This makes them an ideal candidate for use in the field. In this research, the authors conceived and constructed a one-of-a-kind anomaly-based intrusion detection model for IoT networks. Convolutional neural network models are used to develop a multiclass classification system as the first step in the process. Convolutional neural networks are then used to bring the proposed model to life in one, two, and three dimensions. The proposed convolutional neural network model is Validated using the BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, and IoT-23 intrusion detection datasets. Transfer learning is used to accomplish binary and multiclass classification, and a convolutional neural network multiclass pre-trained model is deployed for this purpose. Compared to previously developed implementations of deep learning, their suggested binary and multiclass classification models demonstrated high accuracy, precision, recall, and F1 scores.

It was tested as a deep learning strategy for intrusion detection in an IoT multi-cloud context¹³². The Internet of Things (IoT) allows users to connect billions of intelligent end devices and provide various services. On the other hand, when the Internet of Things devices are connected to wireless networks, these devices have uncontrolled access, causing security holes. Integrating cloud computing with the Internet of Things networks is gaining popularity because it reduces the resource limits imposed on sensor nodes. On the other hand, Internet of Things networks are susceptible to attacks because of the intricacy of their networks and the open broadcast features they have. Intrusion Detection Systems, often known as IDS, are

utilized regularly to ensure network security and maintain reliable operations. IDS can locate anomalies and assure network security even in complex network environments. Traditional neural network-based intrusion detection systems have a high barrier to entry in terms of learning, and their classification accuracy is subpar. Recently, deep learning strategies have gained traction in the image and signal processing industry and security applications. This research study presents a deep learning-based intrusion detection system for multi-cloud IoT scenarios to overcome the limitations of neural network-based intrusion detection models. The proposed model for intrusion detection improves detection accuracy by making training more effective. Hence, the model should be used. In an experimental Evaluation using the NSL-KDD dataset, the recommended model outperformed typical techniques. It had a detection rate of 97.51 percent, a detection accuracy of 96.28 percent, and a precision of 94.41 percent.

A Deep Learning-based IoT Intrusion Detection System with Enhanced Transient Search Optimization was proposed¹³³. Significant security concerns have arisen due to recent advances in cloud computing, the Internet of Things (IoT), and online communication. Because of these improvements, the number of cyberattacks is on the rise, and the security measures now in place cannot give effective cures. Recent research has led to the development of several artificial intelligence (AI)-based solutions for various security-related problems, including detecting intrusions. In this piece of research, the authors presented an efficient artificial intelligence-based mechanism for intrusion detection systems (IDS) in IoT systems. They tackled the challenging technical issues by utilizing deep learning and metaheuristics (MH) algorithms, both effective in similar situations. They presented a method for extracting features based on convolutional neural networks to extract important properties (CNNs). In addition to this, they developed an innovative method for feature selection that they referred to as TSOE. This method uses the operators of the differential evolution (DE)

algorithm and is founded on a new variant of the transient search optimization (TSO) algorithm. In the proposed TSOE, the DE will be utilized to facilitate an improvement in the balancing procedure that will take place between the exploration and exploitation stages. In addition, they examined the performance of the newly developed method by using three public datasets, namely KDDCup-99, NSL-KDD, BoT-IoT, and CICIDS-2017. The results showed that the newly developed method attained a greater accuracy level than previous approaches already in use.

A combination of machine and deep learning for feature extraction in the Industrial Internet of Things was investigated¹³⁴. An approach to feature extraction for the Industrial Internet of Things that uses machine learning and deep learning techniques (IIoT). To train the model with the most useful feature set, they utilized machine learning techniques and approaches for extracting features from deep learning. To validate their findings, they employed network models based on deep learning to Validate these qualities. They believed that error metrics and accuracy were the most significant variables to consider when combining machine learning with deep learning techniques. The mean square error was used to calculate the error rates, indicating that the subset had a lower error rate than the model Validated on the complete dataset. In addition, the performance of the suggested model is examined using mean square error, accuracy rate, and false value statistics. According to the comparison, the performance of their approach is 25% better than that of the competitors when measured against the IIoT dataset and other existing methodologies.

Intrusion Detection in the Network It was investigated using a novel feature selection model and various recurrent neural networks¹³⁵. The current uptick in hacking and attacks on computer networks worldwide has brought to light the pressing requirement for enhanced intrusion detection and prevention systems. Because of their increased magnitude and scope,

network attacks have become more difficult to detect without the assistance of an intrusion detection system, also known as an IDS. On the other side, the majority of anomaly-based intrusion detection systems have a high percentage of false positives in their detection rates. In addition, the Remote-to-Local (R2L) and User-to-Root (U2R) threats have poor expected accuracy scores in modern IDS techniques. Consequently, the authors of this paper present an original IDS paradigm to solve these IDS problems. The suggested structure contains three distinct elements altogether. The first thing that needs to be done is to construct the SFSDT model, also known as the feature selection model. The purpose of the SFSDT is to locate the most advantageous feature subset from all available feature sets. The Sequence Forward Selection (SFS) and the Decision Tree (DT) algorithms have been incorporated into this model. The second step is to develop several IDS models, each of which will be trained using the most optimal feature subset. There are a few distinct varieties of Recurrent Neural Networks (RNN), the most common of which are the Traditional RNN, the Long Short-Term Memory (LSTM), and the Gated Recurrent Unit (GRU). Experiments form the basis for the trained models based on two datasets from IDS: NSL-KDD from 2010 and ISCX from 2012. The last thing that must be done is to Validate the suggested model by comparing it to various alternative IDS models. According to the findings of the experiments, the proposed models significantly improve both the accuracy of the detection rate and the classification of the various types of attacks. In addition, by evaluating RAM profilers, this method can reduce the time spent on calculations.

The Taxonomy, Reference Architecture, and Analyses for IoT Intrusion Detection were examined¹⁶⁶. This work aims to identify gaps, vulnerabilities, and a neutral reference architecture for deep learning (DL) approaches for intrusion-detection systems (IDSs) in the Internet of Things. This work aims to identify gaps, vulnerabilities, and a neutral reference architecture for deep learning (DL) approaches (IoT). A comparison of several intrusion

detection systems (IDS) and a discussion of anomaly-based IDSs on DL methodologies are presented. These techniques include supervised, unsupervised, and hybrid methods. In the Internet of Things (IoT) context, each of the techniques contained within these three categories has been chiefly implemented. To this day, anomaly-based IDS for the Internet of Things has only been implemented in a select few. For each anomaly-based IDS, an analysis was conducted on implementing the four categories of feature(s) extraction, classification, prediction, and regression. They looked at essential performance parameters, such as benchmark detection rates and critical performance parameters, and the required efficiency of the various methodologies. To classify the data, we examined and compared four different machine learning strategies: logistic regression (LR), support vector machine (SVM), decision tree (DT), and an artificial neural network (ANN). Consequently, the Receiver Operating Characteristic (ROC) curve was utilized to make the comparison. The study model demonstrates good results across the board for different assaults. They researched threats to the IoT ecosystem using factually based and simulation-generated datasets (namely the Bot-IoT and the IoTID20 datasets).

For the Internet of Things, a new ensemble-based intrusion detection system was proposed¹⁶⁷. The Internet of Things (IoT) has shown tremendous endurance over the past several years by fundamentally altering human lives by automating basic daily tasks. This has allowed IoT to display its true potential. This is achieved by connecting different kinds of physical devices, each of which has unique functionality. The proliferation of the Internet of Things networks has led to increased cyber attacks, which puts the data's integrity and stability in danger. Several different intrusion detection systems, abbreviated as "IDSs," have been proposed to protect data against unanticipated efforts and inappropriate usage by identifying potentially destructive behaviors based on established attack patterns. The current intrusion detection systems (IDS) need to be upgraded because there has been a sharp uptick in attacks of this

kind. Machine learning to enhance intrusion detection systems has become an increasingly important focus in recent years. Within the scope of this work is the proposal of an ensemble-based intrusion detection model. After evaluating the model's performance with several well-known methods considered to be state-of-the-art at the moment, the suggested model was modified to include logistic regression, naive Bayes, and decision trees, in addition to a voting classifier. In addition, the CICIDS2017 dataset was utilized to conduct an efficacy analysis of the proposed model. Compared to previously developed models, the findings demonstrate a significant improvement in accuracy in classification scenarios involving binary and multi-class categories.

A review of IoT-based intrusion detection systems was conducted¹³⁸. The Internet of Things (IoT) is a modern concept that aims to connect the physical objects used in various industries with the Internet. These industries include manufacturing, business automation, human health and fitness, and environmental monitoring. It increases the use of things connected to the internet in everyday tasks, raising issues about safety and utility. Since their introduction, Intrusion Detection Systems (IDS) have proven to be an effective tool for safeguarding computer networks and information systems. On the other hand, implementing traditional IDS techniques on IoT is unlikely to be successful due to several unique factors—for instance, specific protocol stacks, strained-asset gadgets, and measurements. The findings of IDS investigations into the Internet of Things are analyzed in this research. You will need to recognize significant biases, general disadvantages, and investigation rules to accomplish the goal. The proposed IDSs that comprise the state-of-the-art were organized into categories according to the following characteristics: discernment technique, IDS installation procedure, security risk, and confirmation procedure. The paper also launched an attack differentiating process for IoT threats that IDSs could address. Additionally, the paper investigated every

flaw in an attribute, looked at various material perspectives, and made recommendations for specific IDS procedures for Internet of Things (IoT) threats.

It was suggested that an intrusion detection system be developed to progress the deep learning algorithms based on the Internet of Things infrastructure. As a result of the Internet of Things (IoT), which uses sensor devices to collect data from an environment containing a smart grid, smart grids have become popular targets for penetration. These data are transferred to the cloud, a massive network of supercomputers that provides various services to intelligent infrastructures such as smart homes and buildings. These infrastructures include smart cities and transportation systems. These can provide cybercriminals with many leeways to maneuver, allowing them to launch destructive cyberattacks. The innovative component of this suggested line of research is establishing a solid framework for intrusion detection based on the environment of IoT devices¹⁹¹. The approach that was suggested was developed by the use of an IoTID20 dataset assault, which is a dataset that was recently developed from the IoT infrastructure. In this framework, the classification of the incursion was accomplished by utilizing three different advanced deep-learning algorithms. These algorithms were a convolution neural network (CNN), a long short-term memory (LSTM), and a hybrid convolution neural network with the long short-term memory (CNN-LSTM) model. To enhance the proposed system, the dimensionality of the network dataset was scaled down, and the particle swarm optimization method (PSO) was applied to extract essential characteristics from the network dataset. To process the features that were obtained, deep learning algorithms were utilized. According to the results of the experiments, the proposed systems achieved the following levels of accuracy: CNN = 96.60 percent, LSTM = 99.82 percent, and CNN-LSTM = 98.80 percent. The proposed framework obtained the needed results when applied to a new variable dataset, and the system will be implemented in the Internet of Things environment at the university. The proposed framework was compared to other systems already in place, and

the results showed that the suggested system improved the security of the IoT environment against threats more efficiently and effectively. The findings of the experiments demonstrated that the proposed architecture for an intrusion detection system based on deep learning algorithms can efficiently detect real-world threats and enhance the level of security provided by the IoT environment.

This paper presents a comprehensive analysis of existing IoT intrusion detection systems (IDS) and an overview of machine learning approaches, deployment tactics, evaluation procedures, and datasets frequently utilized in the creation of IDS. This study investigates how existing IoT intrusion detection systems (IDS) identify invasive attacks and ensure secure IoT connections. It also identifies future research problems that need to be addressed to combat risks posed by the Internet of Things and make the IoT more secure, as well as the categorization of IoT attacks. Researchers working on the security of IoT devices will benefit from these goals because they will facilitate the consolidation, comparison, and organization of previously separate research initiatives. As a result of this research, a novel IoT intrusion detection system taxonomy has been developed. This taxonomy elucidates IoT intrusion detection system methodologies, their advantages and disadvantages, IoT attacks that target IoT communication systems, and the associated enhanced IDS and detection capabilities to identify IoT intrusions.

It was suggested that there be a mechanism for identifying and categorizing network breaches. In contrast to most of the preceding research, which utilized the KDD99 dataset, this study utilized the UNSW-NB 15 dataset, which is both more recent and comprehensive. This data collection is comprised of network packets that were transmitted and received by hosts from one another. It possesses 49 attributes, some of which are DoS, Fuzzers, Exploit, Worms, shellcode, reconnaissance, generic, and analysis Backdoor. Other attributes include these. In

this study, SVM and MLP were utilized to compare each other. Finally, a decision tree classifier called AdaBoost was constructed to differentiate typical activity from potential threats. The study analyzed network traffic and classified it as either a threat or not a threat based on its characteristics. Using the UNSW-NB15 dataset, the experimental results demonstrated that our proposed method accurately detects multiple kinds of network intrusions on computer networks, with a detection rate of 99.3 percent. Applications in research, as well as network security, will benefit from the approach that has been provided¹⁴⁰.

An automated digital forensic technique in conjunction with intrusion detection systems was proposed¹⁴¹. An Intrusion Detection System was utilized, and it was able to identify the breach (IDS). The primary function of this system is to detect intrusions and react swiftly to them. Put another way, the function of the IDS is limited to detection and reaction¹⁴². When an intrusion is identified, the intrusion detection system is unable to record the current state of the system²⁴³. As a direct consequence, the evidence against the attack is not kept in the form in which it was initially presented. A novel security strategy is urgently required to guarantee the authenticity and completeness of the evidence that will be used in later investigations. An automated Digital Forensic Technique, along with an Intrusion Detection System, is proposed throughout this research. When an intrusion is discovered by an intrusion detection system (IDS), the IDS first notifies the administrator by sending them an alarm message. Then, it utilizes a digital forensic tool to take a snapshot of the system's present state. If legal action is taken, the photograph taken may be used as evidence of the harm.

A hybrid model was proposed. It is motivated by deep belief criteria and meta-heuristics. Using Search-and-Rescue—BrainStorm Optimization (SAR-BSO), a hybrid feature selection (FS) and deep belief network classifier is used to localize and detect anomalies for streaming data logs. The significant contribution of the research lies in FS, which is carried out using

SAR-BSO, which increases the detection power of the model as it selects the most significant variables by minimizing redundant features.

A bio-inspired optimized technique is adapted such that feature selection is more effective. The feature vector established using informative data is employed for anomaly detection using the proposed SAR-BSO-based DBN classifier that declares the normal and abnormal events from the data logs. The proposed algorithm also addresses the challenges in the performance fluctuation and complexities in adapting the new techniques.

2.9.1 Summary of Literature Reviewed

During the extensive literature review in this study, GA and other evolutionary algorithms were prominent among the best existing models for optimization. This is because it deploys the cause of nature in its decision-making processes. It is equally adjudged to be good at producing optimal solutions. Existing methods for improving hybridized algorithm models for intrusion detection have employed various techniques. These include incorporating machine learning algorithms such as Genetic Algorithms, Support Vector Machines, Artificial Neural Networks, and Random Forests to enhance detection accuracy and robustness. Feature selection and dimensionality reduction methods like Principal Component Analysis and Genetic Programming have been utilized to improve the efficiency and effectiveness of the hybrid models. Moreover, ensemble learning approaches, such as AdaBoost and Bagging, have been employed to combine the outputs of multiple algorithms for improved detection performance. These methods contribute to developing advanced and reliable hybridized algorithm models for effective intrusion detection systems. The researchers at various times suggested combinations of existing and improvements on existing Models to combine the failings of these existing models. The failings are mainly due to the evolution of threats, so researchers must keep up the pace of improving on models for Intrusion detection. These suggestions birthed this study of improving and hybridizing the feature selection process of

Machine learning-based IDS development. Where a hybrid of DR is used to decipher what the text on unclear scenery pictures is saying. A Hybrid and Improved Genetic Algorithm with Local Linear Embedding Model for Classifying Intrusion Detection Systems.

Endnotes

¹ J-P, Yaacoub, N. Hassan, S. Ola, & C. Ali. "Security Analysis of Drones Systems: Attacks, Limitations, and Recommendations." **Internet of Things 11**, 2020, 100218.

² M.I, Alghamdi. "Digital Forensics in Cyber Security—Recent Trends, Threats, and Opportunities". **Cybersecurity Threats with New Perspectives**, 2021

³ S. Das, & T. Nayak. "Impact of Cybercrime: Issues and Challenges". **International Journal of Engineering Sciences & Emerging Technologies**, 6(2), 2013, 142-153.

⁴ M. Abulaish, N.A.H. Haldar, & J. Jahiruddin. "P2DF: A Privacy-Preserving Digital Forensics Framework". **International Journal of Digital Crime and Forensics (IJDCF)**, 13(6), 2021, 1-15.

⁵ O.L. Al-Sanjary, A. A. Ahmed, M.N. Mohammed, & K.L.T. Aik. "Challenges on Digital Cyber-Security and Network Forensics: A Survey". In **International Conference of Reliable Information and Communication Technology Cham: Springer International Publishing**. 2021, pp. 524-537.

⁶ G. George, & C. Uppin. "A proactive approach to network forensics intrusion (denial of service flood attack) using dynamic features, selection and convolution neural network". **Open Journal of Physical Science**, 2021, (ISSN: 2734-2123), 2(2), 01-09.

⁷ S. Lee, A. Abdullah, & N.Z. Jhanjhi. "A Review on Honeypot-Based Botnet Detection Models for Smart Factory". **International Journal of Advanced Computer Science and Applications**, 2020.

⁸ R. Rivera, L. Pazmiño, F. Becerra, & J. Barriga. "An Analysis of Cyber Espionage Process. In *Developments and Advances in Defense and Security*": **Proceedings of MICRADS Springer Singapore**, 2021, pp. 3-14.

⁹ D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani & D. Garant. "Botnet Detection Based on Traffic Behavior Analysis and Flow Intervals". **Computers & Security**, 39, 2013, 2-16.

¹⁰ F. Casino, T.K. Dasaklis, G.P. Spathoulas, M. Anagnostopoulos, A. Ghosal, I. Borocz, & C. Patsakis. "Research Trends, Challenges, and Emerging Topics in Digital Forensics: A Review of Reviews". **IEEE Access**, 10, 2022, 25464-25493.

- ¹¹ V. Ndatinya, Z. Xiao, V.R. Manepalli, K. Meng, & Y. Xiao, "Network Forensics Analysis using Wireshark". **International Journal of Security and Networks**, 10(2), 2015, 91-106.
- ¹² M.F. Alex, & R. Kishore. "Forensics Framework for Cloud Computing". **Computers & Electrical Engineering**, 60, 2017, 193-205.
- ¹³ A. Khraisat, & A. Alazab. "A Critical Review of Intrusion Detection Systems in the Internet of Things: Techniques, Deployment Strategy, Evaluation Strategy, Attacks, Public Datasets and Challenges". **Cybersecurity**, 4, 2021, 1-27.
- ¹⁴ K. Lyko, M. Nitzschke, & A.C. Ngonga Ngomo. "Big Data Acquisition. New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe", 2016, 39-61.
- ¹⁵ J.F. Pinto da Costa, & M. Cabral. "Statistical Methods with Applications in Data Mining: A Review of the Most Recent Works". **Mathematics**, 10(6), 2022, 993.
- ¹⁶ E.G. Dada, J.S. Bassi, H. Chiroma, A.O. Adetunmbi, & O.E. Ajibuwa. "Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems". **Heliyon**, 5(6), 2019.
- ¹⁷ Y. Pan, F. Sun, Z. Teng, J. White, D.C. Schmidt, J. Staples, & L. Krause. "Detecting Web Attacks with End-to-End Deep Learning". **Journal of Internet Services and Applications**, 10(1), 2019, 1-22.
- ¹⁸ A. Ukil, J. Sen, & S. Koilakonda. "Embedded Security for Internet of Things". In **2011 2nd National Conference on Emerging Trends and Applications in Computer Science IEEE**. 2011, pp. 1-6
- ¹⁹ G. A. Di Lucca, A. R. Fasolino, M. Mastroianni & P. Tramontana, "Identifying Cross Site Scripting Vulnerabilities in Web Applications," **Proceedings. Sixth IEEE International Workshop on Web Site Evolution, Chicago, IL, USA**, 2004, pp. 71-80.
- ²⁰ Cai, M., Huang, H., & Huang, J. "Understanding Security Vulnerabilities in File Systems". In **Proceedings of the 10th ACM SIGOPS Asia-Pacific Workshop on Systems**, 2019, pp. 8-15.
- ²¹ McIlroy, J. Sevcik, T. Tebbi, B.L. Titzer, & T. Verwaest. "Spectre is Here to Stay: An Analysis of Side-Channels and Speculative Execution". **arXiv preprint arXiv:1902.05178**, 2019.
- ²² J. Jang-Jaccard, & S. Nepal. "A Survey of Emerging Threats in Cybersecurity". **Journal of Computer and System Sciences**, 80(5), 2014, 973-993..
- ²³ N. Rakotondravony, B. Taubmann, W. Mandarawi, E. Weishäupl, P. Xu, B. Kolosnjaji, P. Mykolai, D.M. Hermann & H.P. Reiser. "Classifying Malware Attacks in IaaS Cloud Environments". **Journal of Cloud Computing**, 6, 2017, 1-12.
- ²⁴ D.S. Wall. "Enemies Within: Redefining the Insider Threat in Organizational Security Policy". **Security Journal**, 26, 2013, 107-124.
- ²⁵ J. Wiles, T. Gudaitis, J. Jabbusch, R. Rogers, & S. Lowther. "Low Tech Hacking": Street Smarts for Security Professionals. **Elsevier**, 2011.

- ²⁶ C. Cerrudo, & E. M. Fayo. “*Hacking Databases for Owning Your Data*”. 2007
- ²⁷ S. Sadehpour, & N. Vljajic. “*Click Fraud in Digital Advertising: A Comprehensive Survey*”. **Computers**, 10(12), 2021, 164.
- ²⁸ L. Han. “*Password Cracking and Countermeasures in Computer Security: A Survey*”. **arXiv preprint arXiv**: 2014.
- ²⁹ J. Address, & S. Winterfield. “*Non-State Actors in Computer Network Operations*”. **Cyber Warfare: Techniques, Tactics and Tools for Security Practitioners**, 2014, 207-219.
- ³¹ A. Mallik. “*Man-In-The-Middle-Attack: Understanding in Simple Words*”. **Cyberspace: Jurnal Pendidikan Teknologi Informasi**, 2(2), 2019, 109-134.
- ³² A. Ashiru. “*Identifying Phishing as a form of cybercrime in Nigeria*”. **Nnamdi Azikiwe University Journal of International Law and Jurisprudence**, 12(2), 2021, 176-186.
- ³³ L. Fang, H. Wu, K. Qian, W. Wang, & L. Han. “*A Comprehensive Analysis of DDoS Attacks Based on DNS*”. In **Journal of Physics: Conference Series**, Vol. 2024, 2021, No. 1, p. 012027.
- ³⁴ S. Sivamohan, R. Veeramani, K. Liza, S. Krishnaveni, & B. Jothi. “*Data Mining Technique for DDoS Attack in Cloud Computing*”. **International Journal of Computer Technology and Applications**, 9, 2016, 149-156.
- ³⁵ N. Zlatanov. “*Computer Security and Mobile Security Challenges*”. **Conference: Tech Security Conference At**, 2015.
- ³⁶ M.A. Qadeer, M. A. Iqbal, M. Zahid, & M.R. Siddiqui. “*Network Traffic Analysis and Intrusion Detection Using Packet Sniffer*”. **2010 Second International Conference on Communication Software and Networks IEEE**. 2010, pp. 313-317.
- ³⁷ L. Cazorla, C. Alcaraz & J. Lopez, “*Cyber Stealth Attacks in Critical Information Infrastructures*,” in **IEEE Systems Journal**, June, vol. 12, 2018, no. 2, pp. 1778-1792,
- ³⁸ J.B. Ulven, & G. Wangen. “*A Systematic Review of Cybersecurity Risks in Higher Education*”. **Future Internet**, 13(2), 2021, 39.
- ³⁹ R. Ávila, R. Khoury, R. Khoury, & F. Petrillo. “*Use of Security Logs for Data Leak detection: a systematic literature review*”. **Security and Communication Networks**, 2021, 1-29.
- ⁴⁰ J. Jang-Jaccard, & S. Nepal. “*A Survey of Emerging Threats in Cybersecurity*”. **Journal of Computer and System Sciences**, 80(5), 2014, 973-993.
- ⁴¹ G. Wang, J. Yu, & Q. Xie. “*Security Analysis of a Single Sign-On Mechanism for Distributed Computer Networks*”. **IEEE Transactions on Industrial Informatics**, 9(1), 2012, 294-302.
- ⁴² A. Kumara, & C.D. Jaidhar. “*Hypervisor and Virtual Machine Dependent Intrusion Detection and Prevention System for Virtualized Cloud Environment*”. **1st International**

Conference on Telematics and Future Generation Networks (TAFGEN) IEEE. 2015, pp. 28-33.

⁴³T. Ormandy. “*An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments*”, 2007.

⁴⁴ G.M Makrakis, C. Kolias, G. Kambourakis, C. Rieger, & J. Benjamin. “*Vulnerabilities and Attacks against Industrial Control Systems and Critical Infrastructures*”. **arXiv preprint arXiv:2109.03945**, 2021

⁴⁵ J-P. A. Yaacoub, H.N. Noura, O. Salman, & A. Chehab. “A Survey on Ethical Hacking: Issues and Challenges”. **arXiv preprint**, 2021.

⁴⁶ B. Pinkas, & T. Sander. “*Securing Passwords against Dictionary Attacks*”. **In Proceedings of the 9th ACM Conference on Computer and Communications Security**, 2002, pp. 161-170.

⁴⁷ K.N Mallikarjunan, K. Muthupriya, & S.M Shalinie. “*A Survey of Distributed Denial of Service Attack*”. **In 2016 10th International Conference on Intelligent Systems and Control (ISCO) IEEE**, 2016, pp. 1-6.

⁴⁸ C. Haley, R. Laney, J. Moffett, & B. Nuseibeh. “*Security Requirements Engineering: A Framework for Representation and Analysis*”. **IEEE Transactions on Software Engineering**, 34(1), 2008, 133-153.

⁴⁹ S. Prowell, R. Kraus, & M. Borkin, M. “*Seven Deadliest Network Attacks*”. **Elsevier**. 2010.

⁵⁰ A. Nagarajan, V. Varadharajan, & N. Tarr. “*Trust Enhanced Distributed Authorisation for Web Services*”. **Journal of Computer and System Sciences**, 80(5), 2014, 916-934.

⁵¹ M, Ozkan-Okay, R. Samet, O. Aslan, & D. Gupta. “*A Comprehensive Systematic Literature Review on Intrusion Detection Systems*” **IEEE Access**, 9, 2021, 157727-157760.

⁵²E.E. Abdallah, & A.F. Otoom. “*Intrusion Detection Systems using Supervised Machine Learning Techniques: A Survey*”. **Procedia Computer Science**, 2022, 201, 205-212.

⁵³ A. Abbas, M.A. Khan, S. Latif, M. Ajaz, A.A. Shah, & J. Ahmad. “*A New Ensemble-Based Intrusion Detection System for Internet of Things*”. **Arabian Journal for Science and Engineering**, 2021, 1-15.

⁵⁴ D.A. Kumar, & S.R. Venugopalan. “*Intrusion Detection Systems: A Review*”. **International Journal of Advanced Research in Computer Science**, 8(8), 2017, 356-370.

⁵⁵ A. Khraisat, G. Iqbal, V. Peter, & K. Joarder. “*Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges*.” **Cybersecurity**, 2: 2019, 1-22.

⁵⁶ A. Patcha, & J.M Park. “*Network Anomaly Detection with Incomplete Audit Data*”. **Computer Networks**, 51(13), 2007, 3935-3955.

⁵⁷ A. Khraisat, A., & A. Alazab. “*A Critical Review of Intrusion Detection Systems in the Internet of Things: Techniques, Deployment Strategy, Validation Strategy*,”

Attacks, Public Datasets and Challenges.” **Cybersecurity**, 4, 2021, 1-27.

⁵⁸ J. Díaz-Verdejo, J. Muñoz-Calle, A. Estepa Alonso, R. Estepa Alonso, & G. Madinabeitia. “On the Detection Capabilities of Signature-Based Intrusion Detection Systems in the Context of Web Attacks”. **Applied Sciences**, 12(2), 2022, 852.

⁵⁹ E. Jaw, & X. Wang. “A Novel Hybrid-Based Approach of Short Automatic Rule Generator and Security Event Correlation “(SARG-SEC). **PeerJ Computer Science** 8, 2022, e900.

⁶⁰ H. Holm. “Signature Based Intrusion Detection for Zero-Day Attacks:(not) A Closed Chapter?”. In **2014 47th Hawaii International Conference on System Sciences IEEE**, 2014, pp. 4895-4904.

⁶¹ E. Griffiths, & J.M. Chavez. “Longitudinal Crime Trends at Places”. In: **Bruinsma, G., Weisburd, D. (eds) Encyclopedia of Criminology and Criminal Justice**. Springer, New York, NY, 2014.

⁶² N. Agarwal, & S.Z. Hussain. “A Closer Look at Intrusion Detection System for Web Applications”. **Security and Communication Networks**, 2018.

⁶³ A.R. Khan, M. Kashif, R.H. Jhaveri, R. Raut, T. Saba, & S.A. Bahaj. “Deep Learning for Intrusion Detection and Security of Internet of things (IoT): Current Analysis, Challenges, and Possible Solutions”. **Security and Communication Networks**, 2022.

⁶⁴ M. Abdel-Basset, A. Gamal, K.M. Sallam, i. Elgendi, K. Munasinghe, & A. Jamalipour. “An Optimization Model for Appraising Intrusion-Detection Systems for Network Security Communications: Applications, Challenges, and Solutions”. **Sensors**, 22(11), 2022, 4123.

⁶⁵ E.D. Knapp, & J.T. Langill. “Exception, anomaly, and threat detection”. **Industrial Network Security—Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems**, 2015

⁶⁶ G. Creech, & J. Hu. “A Semantic Approach to Host-Based Intrusion Detection Systems using Contiguous and Discontiguous System Call Patterns”. **IEEE Transactions on Computers**, 63(4), 2013, 807-819.

⁶⁷ J. Lin, & S. Amini, J. Hong, N. Sadeh, J. Lindqvist, & J. Zhang. “Expectation and Purpose: Understanding Users' Mental Models of Mobile App Privacy through Crowdsourcing”. **UbiComp'12 - Proceedings of the 2012 ACM Conference on Ubiquitous Computing**, 2012, 501-510.

⁶⁸ D. Miyamoto, H. Hazeyama, & Y. Kadobayashi. “An Evaluation of Machine Learning-Based Methods for Detection of Phishing Sites”. In **Advances in Neuro-Information Processing: 15th International Conference, ICONIP 2008, Auckland, New Zealand, November 25-28, 2008, Revised Selected Papers, Part I 15** Springer Berlin Heidelberg. 2009, pp. 539-546.

⁶⁹ Z. Alkhalil, C. Hewage, L. Nawaf, & I. Khan. “Phishing Attacks: A Recent Comprehensive Study and A New Anatomy”. **Frontiers in Computer Science**, 3, 2021, 563060.

⁷⁰ A. Seyfollahi, & A. Ghaffari. “A Review of Intrusion Detection Systems in

RPL Routing Protocol Based on Machine Learning for Internet of Things Applications". **Wireless Communications and Mobile Computing**, 2021, 1-32.

⁷¹ C. Haley, R. Laney, J. Moffett, & B. Nuseibeh. "Security Requirements Engineering: A Framework for Representation and Analysis". **IEEE Transactions on Software Engineering**, 34(1), 2008, 133-153.

⁷² M. Banerjee, J. Lee, & K.K.R. Choo. "A Blockchain Future for Internet of Things Security: A Position Paper". **Digital Communications and Networks**, 4(3), 2018, 149-160.

⁷³ D. Al-Jumeily, D. Arshad, N. Tariq, T. Baker, H. Tawfik, & M. Asim. "A Lightweight Trust-Enabled Routing in RPL-based IoT Networks Against Sybil Attack". **PLoS One**, 17(7), 2022.

⁷⁸ C.W. Badenhop, S.R. Graham, B.W. Ramsey, B.E. Mullins, & L.O. Mailloux. "The Z-Wave Routing Protocol and its Security Implications". **Computers & Security**, 2017, 68, 112-129.

⁷⁹ P. Schneider, & F. Xhafa. "Anomaly Detection Classification and CEP with ML Methods". **Anomaly Detection and Complex Event Processing over IoT Data Streams**, 2022, 193-233.

⁸⁰ M.W. Watkins. "Exploratory Factor Analysis: A Guide to Best Practice". **Journal of Black Psychology**, 44(3), 2018, 219-246.

⁸¹ N.M. Ahmed, A.H. Abdullah, & O. Kaiwartya. "FSM-F: Finite State Machine Based Framework for Denial of Service and Intrusion Detection in MANET". **Plos one**, 11(6), 2016, e0156885.

⁸² A.L. Schaffer, T.A. Dobbins, & S.A. Pearson. "Interrupted Time Series Analysis using Autoregressive Integrated Moving Average (ARIMA) Models: A Guide for Evaluating Large-Scale Health Interventions". **BMC Medical Research Methodology**, 21(1), 2021, 1-12.

⁸³ L. Sun, & W. Gu. "Pavement Condition Assessment using Fuzzy Logic Theory and Analytic Hierarchy Process". **Journal of Transportation Engineering**, 137(9), 2011, 648-655.

⁸⁴ G. Nascimento & M. Correia, "Anomaly-Based Intrusion Detection in Software as A Service," **2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), Hong Kong, China**, 2011, pp. 19-24.

⁸⁵ K. Hölldobler, B. Rumpe, & A. Wortmann. "Software Language Engineering in the Large: Towards Composing and Deriving Languages". **Computer Languages, Systems & Structures**, 54, 2018, 386-405.

⁸⁶ M. Aldwairi, A.M. Abu-Dalo, & M. Jarrah. "Pattern Matching of Signature-Based IDS using Myers Algorithm under MapReduce Framework". **EURASIP Journal on Information Security**, 2017, 1-11.

⁸⁷ D. Saha, & A. Manickavasagan. "Machine Learning Techniques for Analysis of Hyperspectral Images to Determine Quality of Food Products: A Review". **Current Research in Food Science**, 4, 2021, 28-44.

- ⁸⁸ J. Zhang, B. Olatosi, X. Li, & X. Sun. "Application of Machine Learning Techniques in Classification of HIV Medical Care Status for People Living with HIV in South Carolina". In **APHA's 2020 VIRTUAL Annual Meeting and Expo**, 2020.
- ⁸⁹ M. Javaid, A. Haleem, R.P. Singh, R. Suman, & S. Rab. "Significance of machine learning in healthcare: Features, pillars and applications". **International Journal of Intelligent Networks**, 3, 2022, 58-73.
- ⁹⁰ A.S. Dina, & D. Manivannan. "Intrusion Detection Based on Machine Learning Techniques in Computer Networks". **Internet of Things**, 16, 2021, 100462.
- ⁹¹ R.C. Chen, C. Dewi, S.W. Huang, & R.E. Caraka. "Selecting Critical Features for Data Classification Based on Machine Learning Methods". **Journal of Big Data**, 7(1), 2020, 52.
- ⁹² M. Bijone. 2016. "A Survey on Secure Network: Intrusion Detection & Prevention Approaches". **American Journal of Information Systems**, 4(3), 2016, 69-88.
- ⁹³ M. Ahsan, K.E. Nygard, R. Gomes, M.M. Chowdhury, N. Rifat, N., & J.F. Connolly. "Cybersecurity Threats and their Mitigation Approaches using Machine Learning—A Review". **Journal of Cybersecurity and Privacy**, 2(3), 2022, 527-555.
- ⁹⁴ G. Gür, S. Bahtiyar, & F. Alagöz. "Security Analysis of Computer Networks: Key Concepts and Methodologies". **Modeling and Simulation of Computer Networks and Systems**, 2015, 861-898.
- ⁹⁵ B.C. Stoel. "Artificial Intelligence in Detecting Early RA". In **Seminars in Arthritis and Rheumatism**, 2019, pp. S25-S28.
- ⁹⁶ R. Faia, T. Pinto, & Z. Vale. "GA Optimization Technique for Portfolio Optimization of Electricity Market Participation". In **2016 IEEE Symposium Series on Computational Intelligence (SSCI)**, 2016, pp. 1-7.
- ⁹⁷ J. Alzubi, A. Nayyar, & A. Kumar. "Machine learning from theory to algorithms: an overview". In **Journal of physics: conference series IOP Publishing**, (Vol. 1142, p. 012012), 2018.
- ⁹⁸ M.A. Mahmood, A.I. Visan, C. Ristoscu, & I.N. Mihailescu. "Artificial Neural Network Algorithms for 3D Printing". **Materials**, 14(1), 2020, 163.
- ⁹⁹ H. Zalaghi, & M. Khazaei. "The Role of Deductive and Inductive Reasoning in Accounting Research and Standard Setting". **Asian Journal of Finance & Accounting**, 8(1), 2016, 23-37.
- ¹⁰⁰ K.F. Hyde. "Recognising Deductive Processes in Qualitative Research". **Qualitative Market Research: An International Journal**, 3(2), 2000, 82-90.
- ¹⁰¹ S. Iqbal, & S.A. Alharbi. "Advancing Automation in Digital Forensic Investigations using Machine Learning Forensics". **Digital Forensic Science**. 2019.
- ¹⁰² N. Shah, N. Bhagat, & M. Shah. "Crime Forecasting: A Machine Learning and Computer Vision Approach to Crime Prediction and Prevention". **Visual Computing for Industry, Biomedicine, and Art**, 4, 2021, 1-14.

- ¹⁰⁴ M. Hina, M. Ali, A.R. Javed, F. Ghabban, L.A. Khan, & Z. Jalil. “Sefaced: Semantic-Based Forensic Analysis and Classification of E-mail Data using Deep Learning”. **IEEE Access**, 9, 2021, 98398-98411.
- ¹⁰⁵ H. Saadatfar, S. Khosravi, J.H. Joloudari, A. Mosavi, & S. Shamshirband, S. “A New K-Nearest Neighbors Classifier for Big Data Based on Efficient Data Pruning”. **Mathematics**, 8(2), 2020, 286.
- ¹⁰⁶ D.A. Adeniyi, Z. Wei, & Y. Yongquan. “Automated Web Usage Data Mining and Recommendation System using K-Nearest Neighbor (KNN) Classification Method”. **Applied Computing and Informatics**, 12(1), 2016, 90-108.
- ¹⁰⁷ Y. Liao, & V.R. Vemuri. “Use of K-Nearest Neighbor Classifier for Intrusion Detection”. **Computers & Security**, 21(5), 2002, 439-448.
- ¹⁰⁸ H. Cheng, P.N. Tan, & R. Jin. “Efficient Algorithm for Localized Support Vector Machine”. **IEEE Transactions on Knowledge and Data Engineering**, 22(4), 2009, 537-549.
- ¹⁰⁹ G.M. Foody, & A. Mathur. “Toward Intelligent Training of Supervised Image Classifications: Directing Training Data Acquisition for SVM Classification”. **Remote Sensing of Environment**, 93(1-2), 2004, 107-117.
- ¹¹⁰ R.G. Brereton, & G.R. Lloyd. “Support Vector Machines for Classification and Regression”. **Analyst**, 135(2), 2010, 230-267.
- ¹¹¹ X. Ma, Y. Zhang, H. Cao, S. Zhang, & Y. Zhou. “Nonlinear Regression with High-Dimensional Space Mapping for Blood Component Spectral Quantitative Analysis”. **Journal of Spectroscopy**, 2018, 1-8.
- ¹¹² M. Khorshid, T.H. Abou-El-Enien, & G. Soliman. “A Comparison among Support Vector Machine and other Machine Learning Classification Algorithms”. **IPASJ International Journal of Computer Science (IJCS)**, 2015.
- ¹¹³ Z. Chiba, N. Abghour, K. Moussaid, A. El Omri, & M. Rida. “A Survey of Intrusion Detection Systems for Cloud Computing Environment”. In **2016 International Conference on Engineering & MIS (ICEMIS) IEEE**, 2016, pp. 1-13.
- ¹¹⁴ K. Harimoorthy, & M. Thangavelu. “Multi-Disease Prediction Model using Improved SVM-Radial Bias Technique in Healthcare Monitoring System”. **Journal of Ambient Intelligence and Humanized Computing**, 12, 2021, 3715-3723.
- ¹¹⁵ M.F.A. Saputra, T. Widiyaningtyas, & A.P. Wibawa. “Illiteracy Classification using K Means-Naïve Bayes Algorithm”. **JOIV: International Journal on Informatics Visualization**, 2(3), 2018, 153-158.
- ¹¹⁶ S. Suthaharan, & S. Suthaharan. “Decision Tree Learning”. **Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning**, 2016, 237-269.
- ¹¹⁷ L. Rutkowski, M. Jaworski, L. Pietruczuk, & P. Duda. “The CART Decision Tree for Mining Data Streams”. **Information Sciences**, 266, 2014. 1-15.

- ¹¹⁸ A.H. Hambali, K.S. Yakub, T.O. Oladele, & M.D. Gbolagade. “*ADABOOST Ensemble Algorithms for Breast Cancer Classification*” .2019.
- ¹¹⁹ C Vens, “*Random Forest,*” in **Encyclopedia of Systems Biology** (New York, NY: Springer New York), 2013.
- ¹²⁰ N.A. Rahim, M. Paulraj, & A.H. Adom. “*Adaptive Boosting with SVM Classifier for Moving Vehicle Classification*”. **Procedia Engineering**, 53, 2013. 411-419.
- ¹²¹ N. Torelli, & G. Menardi. “*Evaluating Enterprise Risk of Default using Boosting Procedures*”. 2008
- ¹²² H. Hosseinzadeh, & Z. Einalou. “*Logistic Regression Projection-Based Feature Representation for Visual Domain Adaptation*”. **Signal, Image and Video Processing**, 14, 2020, 1115-1123.
- ¹²³ Y. Xiao, C. Xing, T. Zhang, & Z. Zhao. “*An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks*”. **IEEE Access** 7, 2019. 42210-42219.
- ¹²⁴ J. Kim, J. Kim, H. Kim, M. Shim, & E. Choi. “*CNN-Based Network Intrusion Detection against Denial-of-Service Attacks*”. **Electronics**, 9(6), 2020. 916.
- ¹²⁵ L. Mohammadpour, T.C. Ling, C.S. Liew, & A. Aryanfar. “*A Mean Convolutional Layer for Intrusion Detection System*”. **Security and Communication Networks**, 2020, 1-16.
- ¹²⁶ P. Gopika, V. Sowmya, E.A. Gopalakrishnan, & K.P. Soman. “*Transferable Approach for Cardiac Disease Classification using Deep Learning*”. In **Deep learning Techniques for Biomedical and Health Informatics Academic Press**. 2020, pp. 285-303.
- ¹²⁷ Z. Ahmad, A.K. Shahid, C.S. Wai, J. Abdullah, & F. Ahmad. “*Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches*”. **Transactions on Emerging Telecommunications Technologies**, 32(1), 2021, e4150.
- ¹²⁸ F.M. Bianchi, E. Maiorino, M.C. Kampffmeyer, A. Rizzi, R. Jenssen, F.M. Bianchi, M. Enrico, C.K. Michael, R. Antonello & R. Jenssen. “*Properties and Training in Recurrent Neural Networks*”. **Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis**, 2017, 9-21.
- ¹²⁹ I.H. Sarker. “*Machine Learning: Algorithms, Real-World Applications and Research Directions*”. **SN Computer Science**, 2(3), 2021, 160.
- ¹³⁰ I. Kigwana, V.R. Kebande, & H.S. Venter. “*A Proposed Digital Forensic Investigation Framework for an eGovernment Structure for Uganda*”. In **2017 IST-Africa Week Conference (IST-Africa) IEEE**, 2017, pp. 1-8.
- ¹³¹ K.N. Singh, S.D. Devi, H.M. Devi, & A.K. Mahanta. “*A Novel Approach for Dimension Reduction using Word Embedding: An Enhanced Text Classification Approach*”. **International Journal of Information Management Data Insights**, 2(1), 2022, 100061.

- ¹³² L.H. Nguyen, & S. Holmes. “Ten Quick Tips for Effective Dimensionality Reduction”. **PLoS Computational Biology**, 15(6), 2019, e1006907.
- ¹³³ S. Cateni, M. Vannucci, M. Vannocci, & V. Colla. “Variable Selection and Feature Extraction through Artificial Intelligence Techniques”. **Multivariate Analysis in Management, Engineering and the Science** 6, 2012, 103-118.
- ¹³⁴ L. McInnes, J. Healy, & J. Melville. “Umap: Uniform Manifold Approximation and Projection for Dimension Reduction”. **arXiv preprint arXiv:1802.03426**, 2018.
- ¹³⁵ H.P. Dai, D.D. Chen, & Z.S. Zheng. “Effects of Random Values for Particle Swarm Optimization Algorithm”. **Algorithms**, 11(2), 2018, 23.
- ¹³⁶ M.L. McHugh. 2013. “The Chi-Square Test of Independence”. **Biochemia Medica**, 23(2), 2013, 143-149.
- ¹³⁷ J.F. Wellmann. “Information Theory for Correlation Analysis and Estimation of Uncertainty Reduction in Maps and Models”. **Entropy**, 15(4), 2013. 1464-1485.
- ¹³⁸ H. Shi, B. Schmidt, W. Liu, & W. Müller-Wittig. “Parallel Mutual Information Estimation for Inferring Gene Regulatory Networks on GPUs”. **BMC research notes**, 4(1), 2011, 1-10.
- ¹³⁹ P. Bansal, L. Rishabh, J. Vaibhav, J. Tanmay, S. Sanchit, K. Sumit, K. S. Pradeep, and K. Baseem. “GGA-MLP: A Greedy Genetic Algorithm to Optimize Weights and Biases in Multilayer Perceptron.” **Contrast Media & Molecular Imaging**, 2022.
- ¹⁴⁰ J. Xuan, H. Jiang, & Z. Ren. “Pseudocode of Genetic Algorithm and Multi-Start Strategy Based Simulated Annealing Algorithm for Large Scale Next Release Problem”. **Dalian University of Technology**, 2011.
- ¹⁴¹ T. Jombart, S. Devillard, & F. Balloux. “Discriminant Analysis of Principal Components: A New Method for the Analysis of Genetically Structured Populations”. **BMC Genetics**, 11(1), 2010, 1-15.
- ¹⁴² K.J. Chabathula, C.D. Jaidhar, & M.A. Kumara. “Comparative Study of Principal Component Analysis Based Intrusion Detection Approach using Machine Learning Algorithms”. In **2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN) IEEE**, 2015. pp. 1-6.
- ¹⁴³ B. Zhang, Z. Liu, Y. Jia, J. Ren, & X. Zhao. “Network Intrusion Detection Method Based on PCA and Bayes Algorithm”. **Security and Communication Networks**, 2018, 1-11.
- ¹⁴⁴ J. F. Barraza, E. L. Droguett, & M.R. Martins. “Towards Interpretable Deep Learning: A Feature Selection Framework for Prognostics and Health Management using Deep Neural Networks”. **Sensors**, 21(17), 2021, 5888.
- ¹⁴⁵ L. Alzubaidi, J. Zhang, A.J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, & L. Farhan. “Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions”. **Journal of Big Data**, 8, 2021, 1-74.
- ¹⁴⁶ R. Vargas, A. Mosavi, & R. Ruiz. “Deep Learning: A Review”. **.10.20944/preprints**, 2018.

- ¹⁴⁷ J.G. Lee, S. Jun, Y.W. Cho, H. Lee, G.B. Kim, J.B. Seo, & N. Kim, N. “*Deep Learning in Medical Imaging: General Overview*”. **Korean Journal of Radiology**, 18(4), 2017, 570-584.
- ¹⁴⁸ R. Yamashita, M. Nishio, R.K.G. Do, & K. Togashi. “*Convolutional Neural Networks: An Overview and Application in Radiology*”. **Insights into Imaging** 9, 2018, 611-629.
- ¹⁴⁹ T.M. Navamani. “*Efficient Deep Learning Approaches for Health Informatics*”. **In Deep Learning and Parallel Computing Environment for Bioengineering Systems**, Academic Press, 2019, pp. 123-137.
- ¹⁵⁰ J.Y. Kim. “*FPGA Based Neural Network Accelerators*”. **Advances in Computers Elsevier**, Vol. 122, pp. 2021, 135-165.
- ¹⁵¹ A. Thakare, M. Bhende, N. Deb, S. Degadwala, B. Pant, & Y.P. Kumar. “*Classification of Bioinformatics EEG Data Signals to Identify Depressed Brain State Using CNN Model*”. **BioMed Research International**, 2022.
- ¹⁵² R. DiPietro, & G.D. Hager. “*Deep Learning: RNNs and LSTM*”. **Handbook of Medical Image Computing and Computer Assisted Intervention**. Academic Press, 2020, pp. 503-519.
- ¹⁵³ M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, & E.K. Markakis. “*A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues*”. **IEEE Communications Surveys & Tutorials**, 22(2), 2020, 1191-1221.
- ¹⁵⁴ Z.M. Hira, & D.F. Gillies. “*A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data*”. **Advances in Bioinformatics**, 2015.
- ¹⁵⁵ M. Patel, J. Zhan, K. Natarajan, F. Robert, D. Nigel, S. Paul, J. Grist, J. Duddalwar, A. Peet, and V. Sawlani. “*Machine Learning-Based Radiomic Evaluation of Treatment Response Prediction in Glioblastoma*.” **Clinical radiology** 76, 2021, 628-e17.
- ¹⁵⁶ S.P. Thirimanne, L. Jayawardana, L. Yasakethu, P. Liyanaarachchi, & C. Hewage. “*Deep Neural Network Based Real-Time Intrusion Detection System*”. **SN Computer Science**, 3(2), 2022, 145.
- ¹⁵⁷ A. Dhiman, K. Gupta, & D.K. Sharma. “*An Introduction to Deep Learning Applications in Biometric Recognition*”. **Trends in Deep Learning Methodologies**, 2021, 1-36.
- ¹⁵⁸ A. Asperti, & M. Trentin. “*Balancing Reconstruction Error and Kullback-Leibler Divergence in Variational Autoencoders*”. **IEEE Access** 8, 2020, 199440-199448.
- ¹⁵⁸ S. Wang, J. Di, D.Wang, X. Dai, Y. Hua, X. Gao, & Z. Aiping & J.Gao. “*State-Of-The-Art Review of Artificial Neural Networks to Predict, Characterize and Optimize Pharmaceutical Formulation*”. **Pharmaceutics**, 14(1), 2022, 183.
- ¹⁵⁹ B. Ráduly, K.V. Gernaey, A.G. Capodaglio, P.S. Mikkelsen, & M. Henze. “*Artificial Neural Networks for Rapid WWTP Performance Evaluation: Methodology and Case Study*”. **Environmental Modelling & Software**, 22(8), 2007, 1208-1216.

- ¹⁶⁰C. Junli, & J. Licheng. “*Classification Mechanism of Support Vector Machines*”. In **WCC 2000-ICSP 2000. 2000 5th International Conference on Signal Processing Proceedings. 16th World Computer Congress 2000, IEEE, 2000**, pp. 1556-1559.
- ¹⁶¹ R. Souza, L. Rittner, & R. Lotufo. “*A Comparison Between K-Optimum Path Forest and K-Nearest Neighbors Supervised Classifiers*”. **Pattern Recognition Letters**, 39, 2014, 2-10.
- ¹⁶² R.A. Jamadar. “*Network Intrusion Detection System using Machine Learning*”. **Indian J. Sci. Technol**, 7, 2018, 1-6.
- ¹⁶³ R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, & A. Abuzneid. “*Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection*”. **Electronics**, 8(3), 2019, 322.
- ¹⁶⁴ F. Salo, A.B. Nassif, & A. Essex. “*Dimensionality Reduction with IG-PCA and Ensemble Classifier for Network Intrusion Detection*”. **Computer Networks**, 2019, 164-175.
- ¹⁶⁵ M.H. Kamarudin, C. Maple, & T. Watson. “*Hybrid Feature Selection Technique for Intrusion Detection System*”. **International Journal of High Performance Computing and Networking**, 13(2), 2019, 232-240.
- ¹⁶⁶ I. Sharafaldin, A. Gharib, A.H. Lashkari, & A.A. Ghorbani. “*Towards a Reliable Intrusion Detection Benchmark Dataset*”. **Software Networking**, 2018, 177-200.
- ¹⁶⁷ I.H. Sarker, A.S.M. Kayes, S. Badsha, H. Alqahtani, P. Watters, & A. Ng. “*Cybersecurity Data Science: An Overview from Machine Learning Perspective*”. **Journal of Big data**, 7, 2020, 1-29.
- ¹⁶⁸ I. Sharafaldin, A.H. Lashkari, & A.A. Ghorbani. “*Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*”. **ICISSP**, 2018, 108-116.
- ¹⁶⁹ R. Zuech, T.M. Khoshgoftaar, N. Seliya, M.M. Najafabadi, & C. Kemp. “*A New Intrusion Detection Benchmarking System*”. In **The Twenty-Eighth International Flairs Conference**. 2015.
- ¹⁷⁰ J.O. Nehinbe. “*A Critical Evaluation of Datasets for Investigating IDSs and IPSs Researches*”. In **2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS) IEEE, 2011**, pp. 92-97.
- ¹⁷¹ A. Shiravi, H. Shiravi, M. Tavallaei, & A.A. Ghorbani. “*Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection*”. **Computers & Security**, 31(3), 2012, 357-374.
- ¹⁷² J.L. Leevy, & T.M. Khoshgoftaar. “*A Survey and Analysis of Intrusion Detection Models Based on Cse-Cic-Ids 2018 Big Data*”. **Journal of Big Data**, 7(1), 2020, 1-19.
- ¹⁷³ S. Sindian, & S. Samer. “*An Enhanced Deep Autoencoder-Based Approach for DDoS Attack Detection*”. **Wseas Trans. Syst. Control**, 15, 2020, 716-725.
- ¹⁷⁴ F. Hussain, S.G. Abbas, M. Husnain, U.U. Fayyaz, F. Shahzad, & G.A. Shah. “*IoT DoS and DDoS Attack Detection using ResNet*”. In **2020 IEEE 23rd International Multitopic Conference (INMIC)**, 2020, pp. 1-6.

- ¹⁷⁵ H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, & X. Bellekens. “*Mqtt-iiot-ids2020: Mqtt Internet of Things Intrusion Detection Dataset*”. **IEEE Dataport**, 2020.
- ¹⁷⁶ R. Corizzo, E. Zdravevski, M. Russell, A. Vagliano, & N. Japkowicz. “*Feature Extraction Based on Word Embedding Models for Intrusion Detection in Network Traffic*”. **Journal of Surveillance, Security and Safety**, 2020.
- ¹⁷⁷ M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, & M. Portmann. “*Feature Extraction for Machine Learning-Based Intrusion Detection in IoT Networks*”. **Digital Communications and Networks**. August 28, 2021.
- ¹⁷⁸ S. Moustakidis, & P. Karlsson. “*A Novel Feature Extraction Methodology using Siamese Convolutional Neural Networks for Intrusion Detection*”. **Cybersecurity**, 3(1), 2020, 1-13.
- ¹⁷⁹ Wei, P., Li, Z. Zhang, T. Hu, Z. Li, & D. Liu. “*An Optimization Method for Intrusion Detection Classification Model Based on Deep Belief Network*”. **IEEE Access**, 7, 2019, 87593-87605.
- ¹⁸⁰ Y. Li, Y. Ghoreishi, S. M. & A. Issakhov. “*Improving the Accuracy of Network Intrusion Detection System in Medical IoT Systems through Butterfly Optimization Algorithm*”. **Wireless Personal Communications**, 126(3), 2022, 1999-2017.
- ¹⁸¹ M.K. Chandol, & M.K. Rao. “*Border Collie Cat Optimization for Intrusion Detection System in Healthcare IoT Network Using Deep Recurrent Neural Network*”. **The Computer Journal**, 65(12), 2022, 3181-3198.
- ¹⁸² Y. Zhang, P. Li, & X. Wang. “*Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network*”. **IEEE Access**, 7, 2019, 31711-31722.
- ¹⁸³ C. Liang, B. Shanmugam, S. Azam, A. Karim, A. Islam, M. Zamani, K. Sanaz & N.B. Idris. “*Intrusion Detection System for the Internet of Things Based on Blockchain and Multi-Agent Systems*”. **Electronics**, 9(7), 2020, 1120.
- ¹⁸⁴ P. Verma, A. Dumka, R. Singh, A. Ashok, A. Gehlot, P.K. Malik, S.G. Gurjot, & M. Hedabou. “*A Novel Intrusion Detection Approach using Machine Learning Ensemble for IoT Environments*”. **Applied Sciences**, 11(21), 2021, 10268.
- ¹⁸⁵ I. Ullah, & Q.H. Mahmoud. “*Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks*”. **IEEE Access** 9, 2021 103906-103926.
- ¹⁸⁶ D. Selvapandian, & R. Santhosh. “*Deep Learning Approach for Intrusion Detection in IoT-Multi Cloud Environment*”. **Automated Software Engineering**, 2021, 28, 1-17.
- ¹⁸⁷ A. Fatani, M.A. Elaziz, Dahou, M.A. Al-Qaness, & S. Lu. “*IoT Intrusion Detection System using Deep Learning and Enhanced Transient Search Optimization*”. **IEEE Access** 9, 2021, 123448-123464.
- ¹⁸⁸ P.L.S. Jayalaxmi, R. Saha, G. Kumar, & T.H. Kim. “*Machine and Deep Learning Amalgamation for Feature Extraction in Industrial Internet-of-Things*”. **Computers & Electrical Engineering**, 97, 2022, 107610.

¹⁸⁹ T.T.H. Le, Y. Kim, & H. Kim. “*Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks*”. **Applied Sciences**, 9(7), 2019, 1392.

¹⁹⁰ K. Albulayhi, A.A. Smadi, F.T. Sheldon, & R.K. Abercrombie. “*IoT Intrusion Detection Taxonomy, Reference Architecture, and Analyses*”. **Sensors**, 21(19), 2021, 6432.

¹⁹¹ A. Abbas, M.A. Khan, S. Latif, M. Ajaz, A.A. Shah, & J. Ahmad. “*A New Ensemble-Based Intrusion Detection System for Internet of Things*”. **Arabian Journal for Science and Engineering**, 2021, 1-15.

¹⁹² H. Alkahtani, & T.H. Aldhyani. “*Intrusion Detection System to Advance Internet of Things Infrastructure-Based Deep Learning Algorithms*”. **Complexity**, 2021, 1-18.

¹⁹³ H. Alkahtani, & T.H. Aldhyani. “*Intrusion Detection System to Advance Internet of Things Infrastructure-Based Deep Learning Algorithms*”. **Complexity**, 2021, 1-18.

¹⁹⁴ I. Ahmad, Q.E. Ul Haq, M. Imran, M.O. Alassafi, & R.A. AlGhamdi. “*An Efficient Network Intrusion Detection and Classification System*”. **Mathematics**, 10(3), 2022, 530.

¹⁹⁵ K. Barhate, & C.D. Jaidhar. “*Automated Digital Forensic Technique with Intrusion Detection Systems*”. In **2013 3rd IEEE International Advance Computing Conference (IACC) IEEE**. 2013, pp. 185-189.

Chapter Three

Methodology

This research provides a unique model and optimization method to improve the basic network topology. Figure 3.1 below is the Architectural Workflow of the model. The feature Optimisation stage, the second stage, takes the reduced dataset as input and further refines the features. This stage is subdivided into two phases sequentially. The Generic Algorithm is improved with a Levy Flight, and the output is further optimized with another optimization technique: Local Linear Embedding. The output of the optimization stage's first and second phases are then consequently SVM and KNN machine learning classifiers. Their performances are measured with Accuracy, precision, f1Score, Time Complexity, Sensitivity, and Specificity metrics.

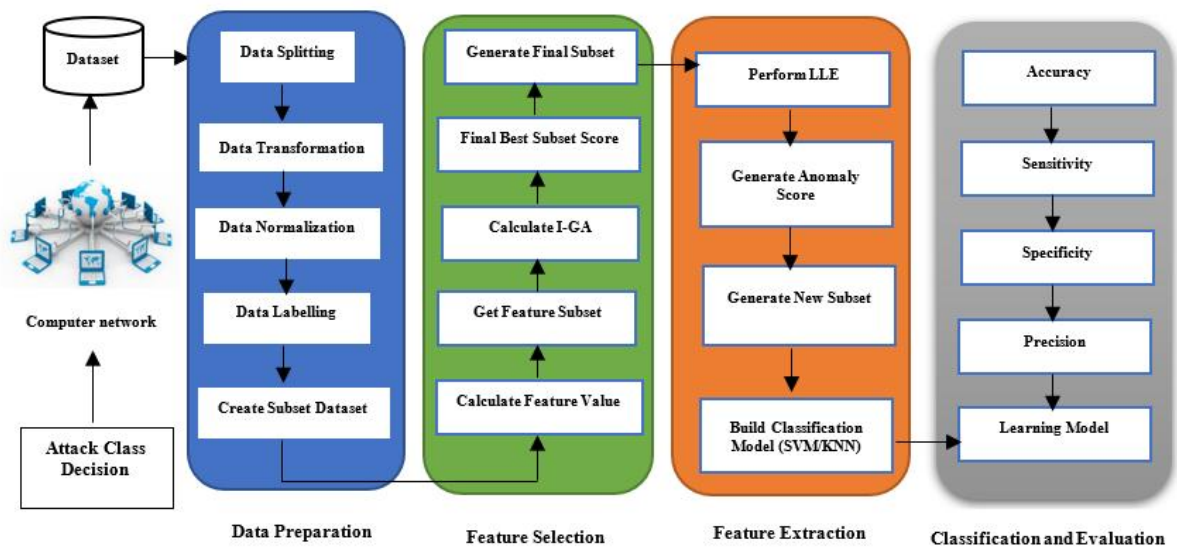


Figure 3.1: Architectural Workflow(Researcher Olufemi T. O. 2023)

The Model Design, as shown in Figure 3.2, further shows the workability of the model. Data pre-processing involves

- data collection,
- preparing and segregating the case scenarios based on specific features involved with the decision-making cycle and

- forwarding the data to the processing unit for further categorization.

This stage is sometimes called the data pre-processing stage. The data model expects reliable, fast, and elastic data, which may be discrete or continuous. Data pre-processing is a conventional technique for eliminating such problems, also known as noise. Pre-processing involves certain activities like data cleaning, integrating the data, data transformation, data reduction, data discretization, and data cleaning. The output of this first stage is a reduced dataset, which is more compact and has less noise or irrelevant features.

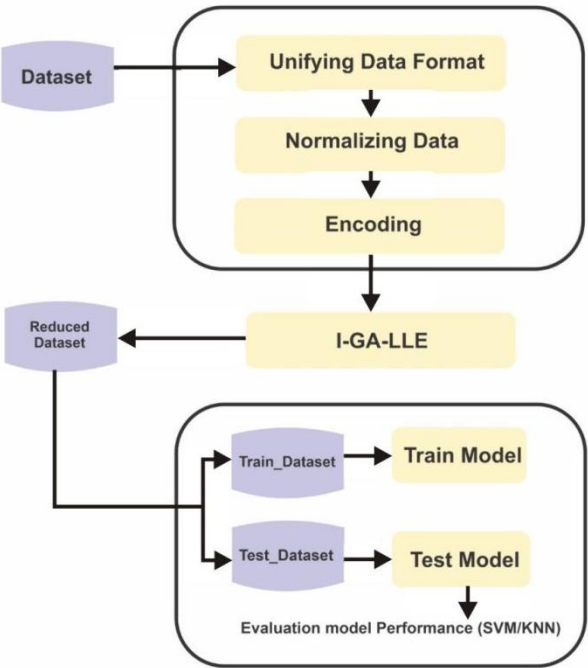


Figure 3.2: Model Design(Researcher Olufemi T. O. 2023)

3.1.1 Dataset

An intrusion detection system (IDS) is an essential security mechanism for preventing complex and widespread network attacks, and its progress depends on data that is freely accessible to the public. When testing IDS solutions, several studies have relied on secret data obtained from a single business or manually acquired data, which negatively impacts the dependability of their findings. Instead of actual network traffic, available datasets such as KDD99 and NSL-KDD were analyzed¹, containing data containing manually selected

stream characteristics. Compared to recent attack tactics, the data-collecting timing needs to be updated.

This research selects the CICIDS2017 dataset (Canadian Institute for Cybersecurity) to reflect the best real traffic scenarios in real networks and newer means of attack^{2,3}. It includes both benign traffic and up-to-date everyday attack traffic, both representative of the environment that a genuine network would have. This dataset represents the abstract behavior of 25 users by utilizing the protocols HTTP, HTTPS, FTP, SSH, and e-mail. The dataset aims to imitate the environment of a real network precisely. The data is gathered beginning at nine in the morning on July 3, 2017, and continuing until five in the afternoon on July 7, 2017, providing a total of 51.1 g of data flow throughout the five days. The eight different kinds of attacks that are identified are as follows: FTP-Patator, SSH-Patator, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. As can be seen in Table 3.1, the attacks happened in the morning and the afternoon on Tuesday, Wednesday, Thursday, and Friday, respectively. On Monday, the traffic generated throughout the day was typical, and from Tuesday through Friday, the traffic generated was moderate. The Pcap file format is used as the data type for this dataset.

After we had finished collecting the data for the dataset, we analyzed the primary data. We selected seven categories for additional study based on the total amount of data and the amount of noise it included. Several varieties include Normal, FTP-Patator, SSH-Patator, DoS, Heartbleed, Infiltration, and PortScan. The CICIDS2017 dataset is sorted and structured into a single Pcap file daily. These Pcap files include a significant amount of data, making it challenging to train the machine. Because of this, the primary objective of the traffic classification is to separate continuous Pcap files into discrete units with a specific granularity. The six ways for slicing network traffic are the Transmission Control Protocol (TCP), connection, network flow, session, and service class. When data on traffic is segmented using

different approaches, it is into a wide variety of diverse forms; hence, the selected path to network traffic segmentation has a considerable bearing on the subsequent analysis.

Throughout this research, " session " refers to any packet with a bidirectional flow and interchangeable source and destination addresses and ports. More specifically, this will refer to any packet that uses the same quad (source IP, source port, destination IP, destination port, and transport layer protocol).

Table 3.1: CICIDS2017 Dataset⁸⁷

Date	Type	Size
Monday	Normal	11.0GB
Tuesday	Normal +Force + SFTP+ SSH	11.0GB
Wednesday	Normal +Dos + Heartbleed Attacks	13.0GB
Thursday	Normal + XSS + Web Attack + Infiltration	7.8GB
Friday	Normal + Botnet + PortScan + DDoS	8.3GB

3.1.2 Dataset

An intrusion detection system (IDS) is an essential security mechanism for preventing complex and widespread network attacks, and its progress depends on data that is freely accessible to the public. When testing IDS solutions, several studies have relied on secret data obtained from a single business or manually acquired data, which negatively impacts the dependability of their findings. Instead of actual network traffic, available datasets such as KDD99 and NSL-KDD were analyzed¹, containing data containing manually selected stream characteristics. Compared to recent attack tactics, the data-collecting timing needs to be updated.

This research selects the CICIDS2017 dataset (Canadian Institute for Cybersecurity) to reflect the best real traffic scenarios in real networks and newer means of attack^{2,3}. It includes both benign traffic and up-to-date everyday attack traffic, both representative of the environment that a genuine network would have. This dataset represents the abstract behavior of 25 users by utilizing the protocols HTTP, HTTPS, FTP, SSH, and e-mail. The dataset aims to imitate the environment of a real network precisely. The data is gathered beginning at nine in the morning on July 3, 2017, and continuing until five in the afternoon on July 7, 2017, providing a total of 51.1 g of data flow throughout the five days. The eight different kinds of attacks that are identified are as follows: FTP-Patator, SSH-Patator, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. As can be seen in Table 3.1, the attacks happened in the morning and the afternoon on Tuesday, Wednesday, Thursday, and Friday, respectively. On Monday, the traffic generated throughout the day was typical, and from Tuesday through Friday, the traffic generated was moderate. The Pcap file format is used as the data type for this dataset.

3.1.3 Python as an Implementation Tool

Python is a modern scripting language that was chosen for various reasons. It has a very clean and simple syntax that is easy to learn for both a programmer and a beginner. Despite its simplicity, Python is an industry-strength language. For instance, Python is behind many of Google's technologies, which is also why Google is one of the major sponsors of Python's development. Since programming in Python is fast, it is very suitable for prototyping new methods. Python allows for relatively easy extension with modules written in C or C++. For this reason, Python is occasionally dubbed a glue language due to its use for gluing libraries in C or Fortran. Lately, one seldom needs to implement specialized routines in low-level languages due to the availability of high-quality libraries like Numpy and Scipy.

3.2 Optimization

This section explains the methods used to address the first objective. This is done using

various optimization algorithms to develop some models.

3.2.1 Improved Genetic Algorithm

The key ideas and implementation strategies of an enhanced GA are carried out in this work, as shown in Figure 3.7, to tackle the problem of GA having slow iteration with convergence speed and falling into the local optimum prematurely when handling complicated problems. A Lévy-GA is proposed in this study.

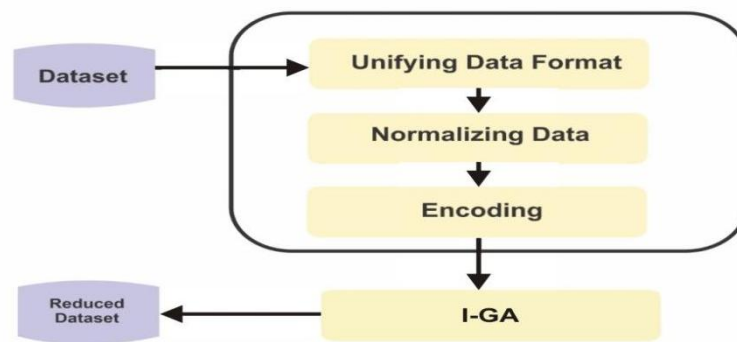


Figure 3.3 Improved Generic Algorithm (I-GA) Model(Researcher Olufemi T. O. 2023)

3.2.1.1 Genetic Algorithm (GA)

An evolution model that may be used to locate the most optimal solutions in a vast search space is called a genetic algorithm (GA). This model is a type of genetic algorithm. The methodology is initially presented by utilizing a population of randomly generated solutions to represent chromosomes as the initial starting point of the program. The number of chromosomes inherited from generation to generation is the primary factor affecting the population's size in most cultures. It depicts the binary alphabet-coded shape of each chromosome. A vector of variables represents each chromosome, and the binary alphabet can only be used to describe a limited number of these variables using a small number of characters. The population was filled by successively shifting chromosomes from another population into the current one, a process referred to as generations. GA employs genetic operators to ensure the organization's commitment to maintaining genetic diversity

is met during its expansion. For evolution to occur, there must first be genetic variation in the population.

In terms of structure and function, genetic operators are comparable to processes that take place in real-world biological evolution, while in frequency, they differ.

(I) Chromosome selection: The fitness value of each chromosome was calculated based on its quality, and the chromosomes with the highest fitness values were passed on to future generations.

(II) Chromosome selection: In crossover/recombination, the chromosomes from the chosen set were united to generate a new set of chromosomes.

Random modifications to the binary encoding of chromosomes were introduced to get the result. This helps preserve population diversity while avoiding the problem of solutions being imposed too soon.

3.2.1.2 Lévy Flight

The French mathematician Paul Levy introduced levy flight in 1937 (1886-1971). Levy flight is a statistical description of motion that extends beyond the more traditional Brownian motion discovered over one hundred years earlier. A diverse range of natural and artificial phenomena are now being described regarding Levy statistics. Lévy flight random walk is a more robust search strategy than Gaussian mutation because its flight step size meets a stable distribution with hefty tails⁴. We can broaden the search range, boost population diversity, and make it simpler to break out of local convergence by incorporating Lévy flight into the optimization method. The equation below shows the modified Lévy flight formula:

$$x_{t+1}^i = x_{ti} + \alpha \times Levy(\lambda) \quad 3.1$$

Where x_{ti} denotes the position and H generation, is the point multiplication operation, is the Lévy flight step control amount, $Levy(\lambda)$ is a random search path, and $u = t(\lambda)$,

We effectively combine Lévy flying with the GA algorithm in this part. To facilitate the problem of sluggish convergence of the GA algorithm and solve complicated situations where the system prematurely falls into the problem of local optima, we add Lévy flight to the iterative process of the GA algorithm, which increases the population diversity of the algorithm. The Lévy flight brings the solution to a new update as $x_{ij} = x_{ij} + \alpha \times Levy(\lambda)$. To overcome this problem, we use a parallel technique to compensate for the detrimental effects of Lévy flight and improve population communication skills. Usually, this strategy does not add to the algorithm's complexity.

The population is divided into samples, and the algorithm steps are as follows:

- (i) Set up the population P and related parameters, then divide it into four groups (1, 2, 3, 4).
- (ii) Evaluation: Determine each particle's fitness value in the population.
- (iii) New information: To update the position of each particle, using Equations (2) and (4). Then, for each group equation, 2, parallel, update the historical ideal best and global optimal best.
- (iv) Communication: Every k generation, communicate across groups and choose the best Lévy flight solution in the group.
- (v) Steps ii-v are repeated to complete the process. Record the global optimal particle and its fitness value until the optimization process is finished if the predetermined function value has been obtained or all iterations have been completed. In addition, Algorithm 3.1 shows the algorithm's flowchart and pseudocode of the proposed algorithm⁵.

The reason for choosing and using the Levy Flight algorithm with a Genetic Algorithm (GA) is to enhance the exploration capability of the GA and improve its ability to search for optimal or near-optimal solutions in complex search spaces. Here are some reasons why Levy Flight can be beneficial with a GA.

Exploration of Large Search Spaces: Levy Flight introduces occasional long jumps in the search space, allowing the algorithm to explore more of the search space than traditional GA operators. This helps escape local optima and search for more promising regions with better solutions.

Efficient Exploration and Exploitation: Levy Flight balances exploration and exploitation. The heavy-tailed Levy flights enable the algorithm to explore new areas efficiently while exploiting promising regions with good results. This combination promotes a more practical search process.

Escape Local Optima: Levy Flight can help the GA escape from local optima, suboptimal solutions in which the algorithm might get trapped. The long jumps introduced by Levy Flight can potentially overcome barriers and enable the GA to explore other regions of the search space that may contain better solutions.

Diverse Solution Space Coverage: Levy Flight adds stochasticity to the search process, resulting in a more diverse solution space exploration. This diversity increases the chances of discovering different solutions, potentially leading to a better understanding of the problem and finding more diverse, high-quality solutions.

Complementary to Genetic Operators: The Levy Flight algorithm can use traditional genetic operators such as mutation and crossover in a GA. While genetic operators provide local search capabilities and exploit existing solutions, Levy Flight complements them by providing global exploration capabilities. Combining these techniques can lead to a more robust and effective search process.

Handling High-Dimensional Data: Intrusion detection systems often operate on high-dimensional feature spaces, posing challenges regarding feature selection, model complexity, and computational efficiency. Combining Levy Flight with a GA can help navigate and search through these high-dimensional spaces more effectively. The random exploration and

exploitation provided by Levy Flight can aid in identifying relevant features and optimizing the classification process.

Promoting Diversity in Solutions: The Levy Flight algorithm adds stochasticity to the search process, leading to a more diverse solution for space exploration. This diversity helps identify a wide range of potential solutions and prevents the algorithm from getting stuck in a limited set of solutions. The algorithm can handle different intrusions and exhibit improved generalization capabilities by promoting diversity.

Hybridization for Improved Performance: The combination of Levy Flight with a GA creates a hybrid approach that leverages the strengths of both algorithms. The GA provides a framework for evolution and optimization, while Levy Flight enhances exploration and exploitation. This synergistic combination can improve IDS classification accuracy, robustness, and efficiency.

Algorithm 3.1 below fuses Levy Flight with Generic Algorithm, which produces a new Novel Algorithm (I-GA) used for this study.

This algorithm combines elements of Genetic Algorithms and Levy flight for optimization purposes. It utilizes genetic operators like crossover, mutation, and Levy flight to explore the search space and find the best solution. The specific details of the Genetic Algorithm parameters and fitness Evaluation would need further clarification to understand the algorithm's workings fully. Algorithm 3.1 is a hybrid algorithm that combines Genetic Algorithms (GAs) and Levy flight for optimization purposes. The algorithm starts by creating an initial population of individuals based on specified GA parameters. The fitness of everyone in the population is Validated, and the best solution found so far is recorded. The algorithm then enters a loop until a stopping criterion is met. During each iteration of the loop, a Levy flight process is introduced. Levy flight is a random walk that helps explore the search space more efficiently.

Within the Levy flight process, parents are selected from the population for reproduction, and crossover and mutation operations are applied to generate offspring. The fitness of the offspring is Validated, and they are added to the population. After the Levy flight process, the population is divided into four groups. The fitness values of individuals in the population are determined, and the positions of the particles are updated. The ideal population and global optimal solution are updated for each group. The algorithm then enters another loop for every k generation. Communication occurs between the groups, and each group's best Lévy flight solution is chosen. This process is repeated until completion. The algorithm concludes the Levy flight process and returns the best solution as the output.

Algorithm 3.1 fuses the principles of Genetic Algorithms and Levy flight to optimize solutions. It utilizes genetic operators for reproduction and variation, incorporating Levy flight to enhance search space exploration. Combining these techniques, the algorithm aims to find the best solution for the given optimization problem.

Algorithm 3.1: Improved Generic Algorithm (I-GA)

1. Input: Genetic Algorithm parameters
2. Start
3. Population $\leftarrow$ Create Initial Population sizes ();
4. Generate Top Result $\leftarrow$ Estimate Fitness (Population)
5. while the ending measure is not gotten done
6. Introduce Levy
7. Start
8. Parents $\leftarrow$ Collection (Population)
9. Offspring $\leftarrow$ Crossover (Parents)
10. Offspring $\leftarrow$ Mutation (Offspring)
11. Finest – Outcome $\leftarrow$ Validate - Fitness (Offspring)
12. Population $\leftarrow$ Population $\cup$ Offspring

13. End while doing Levy.
 14. Set population and parameters,
 15. Divide Population into four groups: g_i $i = 1, 2, 3, 4$).
 16. Determine each fitness value in the population.
 17. Update the position of each particle.
 18. For each group $i = 1, 2, 3, 4$,
 19. Update ideal population best and global optimal gbesti.
 20. For every k generation
 21. Communicate across groups and choose the best Lévy flight solution in the group.
 22. Repeat to complete the process.
 23. End Levy
 24. End
 25. Output: Best-Solution
-

Improved Generic Algorithm(I-GA)²

The processes in the Algorithm 3.1 is as described below.

- i. Input: Genetic Algorithm parameters - This step is where you would provide the necessary parameters for the Genetic Algorithm, such as population size, crossover rate, mutation rate, etc.
- ii. Start - This signifies the start of the algorithm.
- iii. Population $\leftarrow$ Create Initial Population sizes (); - Here, the initial population is created. The size of the population is determined by the parameter provided in the input.
- iv. Generate Top Result $\leftarrow$ Estimate Fitness (Population) - The fitness of each individual in the population is Validated, and the best solution found so far is stored as the top result.
- v. While the ending measure is not gotten done - This initiates a loop that continues until a certain stopping criterion is met.
- vi. Introduce Levy - This step introduces the Levy flight, which is a random walk that can help explore the search space more efficiently.

- vii. Start - This starts the Levy flight process.
- viii. Parents $\leftarrow$ Collection (Population) - Parents are selected from the population for reproduction.
- ix. Offspring $\leftarrow$ Crossover (Parents) - The selected parents undergo crossover, which combines their genetic information to produce offspring.
- x. Offspring $\leftarrow$ Mutation (Offspring) - Mutation is applied to the offspring to introduce further variation.
- xi. Finest – Outcome $\leftarrow$ Validate - Fitness (Offspring) - The fitness of the offspring is Validated.
- xii. Population $\leftarrow$ Population $\cup$ Offspring - The offspring are added to the population.
- xiii. End while doing Levy - This marks the end of the Levy flight process.
- xiv. Set population and parameters - The population and its parameters are set for the next step.
- xv. Divide Population into four groups: $g_i \text{ } i = 1, 2, 3, 4$ - The population is divided into four groups.
- xvi. Determine each fitness value in the population - The fitness value of each individual in the population is determined.
- xvii. Update the position of each particle - The position of each particle in the population is updated.
- xviii. For each group $i = 1, 2, 3, 4$ - This initiates a loop for each group.
- xix. Update ideal population best and global optimal $gbest_i$ - The ideal population best and global optimal solution for each group are updated.
- xx. For every k generation - This initiates a loop for every k generation.
- xxi. Communicate across groups and choose the best Lévy flight solution in the group - Communication occurs between the groups, and the best Lévy flight solution is chosen within each group.

- xxii. Repeat to complete the process - The process is repeated until completion.
- xxiii. End Levy - This marks the end of the Levy flight process.
- xxiv. End - This signifies the end of the algorithm.
- xxv. Output: Best-Solution - The best solution found by the algorithm is returned as the output.

Flowchart 3.1 represents the iterative process of the Improved Genetic Algorithm. It starts with initializing the population, followed by selection, crossover, mutation, and fitness Evaluation. The process continues until a termination condition is met, at which point the best solution is returned. If the termination condition is unmet, the algorithm loops back to the initialization step and repeats the process.

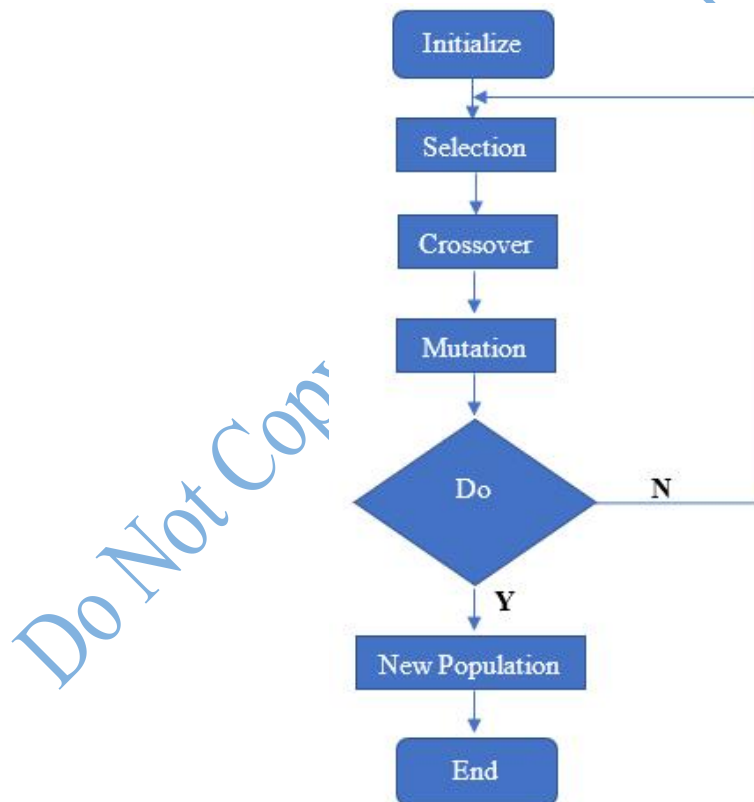


Figure 3.4: Flowchart for I-GA(Researcher Olufemi T. O. 2023)

Initialize: This is the starting point of the algorithm. It involves setting up the initial population, which consists of a set of individuals representing potential solutions.

Selection: In this step, a selection process is applied to the current population to choose individuals for reproduction based on their fitness. Individuals with higher fitness are more likely to be selected, as they have better chances of passing on their genetic information.

Crossover: The selected individuals undergo crossover, combining their genetic information to create new offspring. This process mimics the concept of mating and recombination in natural evolution.

Mutation: After crossover, a mutation operation is applied to introduce random changes in the offspring's genetic material. Mutation helps maintain genetic diversity and allows for exploring new solution spaces.

Do: This step represents the Evaluation of the fitness of the newly generated offspring. The fitness of everyone is assessed, usually based on a predefined fitness function or objective.

If (yes): This is a decision point where the algorithm checks if a termination condition has been met. If the termination condition is satisfied (e.g., a desired fitness level or a maximum number of generations reached), the algorithm proceeds to the next step.

New Population: If the termination condition is met, the algorithm ends and returns the best solution. Otherwise, a new population is formed by replacing the old population with the offspring generated in the previous steps.

Else: If the termination condition is not met, the algorithm returns to the beginning, starting from the Initialize step. This allows for the algorithm's iterative nature, where selection, crossover, mutation, and Evaluation are repeated until the termination condition is fulfilled.

End: This marks the end of the algorithm. It is reached when the termination condition is met, the best solution is returned, or when the algorithm reaches a predefined maximum number of iterations.

3.2.2 Locally Linear Embedding (LLE)

Local linear embedding is a manifold learning method proposed in 2000 by Saul and Rowe. Utilizing a local linear approximation of the global non-linearity and maintaining the local geometry structure by utilizing an overlapping local neighborhood to supply global information to preserve the global geometric features is its premise. The input of the LLE method is a matrix made of $N D$ – dimensional vectors. It produces a matrix consisting of N and d – dimensional vectors. The K column of the matrix matched the K column of the matrix. The LLE algorithm primarily consisted of three steps⁷.

Calculate the distance between each sample point of the high-dimensional space and the other $N-1$ samples. Find the closest points as neighbors based on their distance. Typically, Euclidean distance is employed to determine the separation between two places and compute the weight difference between a feature, its neighbors, and additional features. As the objective is to preserve the locally linear structure of a high-dimensional space in a low-dimensional space as much as feasible, the weight will be fixed to represent local information. Minimize the loss function shown below.

Input:

X : A set of original samples represented as vectors in R^d .

l : A parameter used to select the number of nearest neighbors for virtual sample generation.

k : A parameter used to select the number of neighbors for LLE.

m : The desired dimensionality of the output coordinates.

Process:

Initialize an empty set v to store the virtual samples.

For each sample x_i in X :

Identify the l -nearest neighbors of x_i based on the Euclidean distance.

For each dimension j in the original space:

Select a pair of neighbors randomly and non-repeatedly, denoting them as x_{1R1} and x_{1R2} , where $1 \leq R1 \leq R2 \leq l$.

Calculate the plumb root rij of x_i on the neighbor line x_{1R1}, x_{1R2} .

If the calculated rij is identical to x_i , it means that the neighbor line x_{1R1}, x_{1R2} passes through x_i . Skip to the next iteration.

If rij is different from x_i , assign i to $j-1$.

Add rij to the set V .

Repeat steps 4-11 for all dimensions.

Create a new set X' by combining the original samples X with the virtual samples V .

Apply the LLE algorithm to X' using a neighbor selection parameter of $(d + 1) \times k$ and a desired output dimensionality of m . This step produces a new set Z' of lower-dimensional coordinates.

Obtain the lower-dimensional coordinates Z for the original samples X by using the function $getXcor(Z', X)$.

Output: The lower-dimensional coordinates Z representing the original samples X in the reduced space.

The LLE algorithm aims to find a lower-dimensional representation of high-dimensional data by preserving the local relationships between samples. It starts by identifying the l -nearest neighbors for each sample and constructing virtual samples based on the plumb root calculation. These virtual samples are combined with the original samples, and LLE is applied to obtain the lower-dimensional coordinates. The output represents the original samples in a reduced space that captures the local structure of the data. The algorithm's output is a reduced-dimensional representation of the original samples, where the local relationships within the data are preserved.

LLE performs a two-step process: it generates virtual samples based on the plumb root calculation and then applies LLE to find a lower-dimensional representation. By capturing the local relationships between samples, LLE provides a way to visualize or analyze high-dimensional data in a lower-dimensional space.

Algorithm 3.2: Locally Linear Embedding (LLE)

Input:

- X : original samples $\{x_1, x_2, \dots, x_n\}, x \in R^d$
- l : the neighbor selection parameter used in virtual sample generation
- k : the neighbor selection parameter used in LLE
- m : the dimensionality of the output coordinates

Process:

1. $v = \emptyset$
2. **For** $i=1, 2, \dots, n$ **do**
- 3 identify x_i 's l -nearest neighbors according to Euclidean distance
- 4 **For** $j = 1, 2, \dots, d$ **do**
- 5 Select a pair of neighbors randomly and non-repeatedly, and assume they are x_{1R1} and x_{1R2} ($1 \leq R1 \leq R2 \leq l$)
- 6 $r_{ij} = \text{getPlumbRoot}(x_i, x_{1R1}, x_{1R2})$ % r_{ij} is the plumb root of x_i , on the neighbor line x_{1R1}, x_{1R2} ,
- 7 **If** r_{ij} is identical to x_i % x_{1R1}, x_{1R2} , passes through x_i
- 8 $i = j - 1$
- 9 **else**
- 10 $V = V \cup \{r_{ij}\}$
- 11 **End if**
- 12 **End For**
13. **End For**
14. $X' = X \cup V$
15. $Z' = \text{LLe}(X', (d + 1) \times k, m)$
 % $(d + 1) \times k$ is the neighbor selection parameter used by LLE

16. $Z = \text{getXcor}(Z', X)$ % Get X's lower-dimensional coordinates

Output: Z

Locally Linear Embedding (LLE)⁷

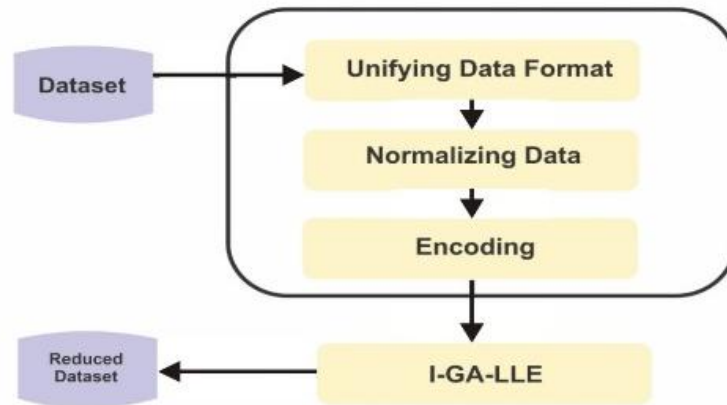


Figure 3.5: IGA-LLE Model(Researcher Olufemi T. O. 2023)

3.3 Classification

4 This section delves into the methods employed to classify the models achieved in 3.2. Support vector Machine (SVM) and K-Nearest Neighbour (K-NN) are chosen because of their uniqueness in classifying models for Intrusion Detection. They are to be used separately on the individual models generated.

5

6 3.3.1 Support Vector Machine

7 Vladimir Vapnik invented the supporting vector machine (SVM) in 1995. To construct an algorithm that predicts many types of outcomes, such as classification, regression, and outlier identification, SVM is a machine learning algorithm. It's most used to analyze small and medium-sized datasets. SVMs have a generalization approach based on structural risk minimization that allows them to capture vast feature spaces (SRM). The Radial Basis Function (RBF) is proposed as the kernel of a Support Vector Machine (SVM) for classification. Two real-valued parameters, C and, must be optimized for the RBF kernel. This parameter trades the potential for training, for

example, misclassification, for the ease with which the decision surface can be understood. The value determines how far an example can spread its impact. It was first scaled to values between zero and one for all feature vectors. In the second stage, a grid search process determines the best C and parameters for the classification. The parameters were scaled on an exponential scale in the first stage, and the results were fine-tuned around the parameter values that led to the best classification in the second step. The Leave-one-Out method is used to cross-validate each C and selection to generalize the classification findings and boost the system's dependability. It's also necessary to ensure that both training data sets are evenly sized by oversampling the smaller data set⁸. The Algorithm 3.3 below is that of the Support Vector Algorithm used for classification in this study.

Algorithm 3.3: SVM Algorithm

Begin

1. Initialized SVM parameter and structure.
2. Generate an initial number of birthing lairs.
3. $LI = (f = 1, 2, 3, \dots, n)$
4. While (Stopping criterion)
5. If noise = false
6. Search in the proximity for a new lair by using a Brownian walk.
7. Else
8. Expand the search for a way for new layer by using levy walk.
9. End if
10. Validate the fitness of each new lair and compare with previous.
11. If
12. $L^{best,t} > L^{best,k+1}$
13. Choose the new lair
14. $L^{best} = L^{best,k}$
15. Else
16. Go to 4
17. End if

18. Rank the solutions;
 19. Return the best lair
 20. The global best lair is fed to SVM classifier for training
 21. Training the SVM classifier
 22. End while
 23. End
-

SVM Algorithm⁸

3.3.2 K Nearest Neighbor

Regarding statistical discrimination, the KNN algorithm is the most straightforward and user-friendly. When utilizing this strategy, a new observation is produced in the group of observations closest to the new observation in terms of using variables. Distance measurements are used to determine similarity. A technique known as "kNN imputation" uses the kNN algorithm to find as many cases as possible that are statistically comparable to the row that is missing data. The kNNs were utilized to calculate an average of the missing data variables. Missing data were imputed using the five persons closest to the researcher in the current study ($k = 5$). Algorithm 3.4 below is the K-Nearest Neighbor Algorithm used as the second classifier deployed for classification purposes.

Input:

x_i : The test data point for which the classification will be predicted.

k : The number of nearest neighbors to consider.

g : Equidistant sample marker.

Initialize the algorithm by providing the test data point x_i and setting the value of k . Also, initialize the equidistant sample marker g .

Perform the following steps in a loop until a termination condition is met.

Calculate the distance, d , between the test data point x_i and each known data point.

Store the known data points in a collection Z , sorted in increasing order of their distances d_i .

Select the first k data points from Z and store them in a collection $A(k)$. These are the k nearest neighbors to the test data point.

If equidistant samples exist (data points with the same distance value), perform the following steps.

Obtain and store the central samples in a collection $B(m)$.

Set the central sample marker g to True, indicating the presence of central samples.

If there are no equidistant samples, set g to False.

If g is False, select the first k minimum distance values from $A(k)$ according to their ordering.

If g is True, determine the distribution of the equidistant samples and update the collection $A(k)$ based on certain equations (Eqs. (7) to (14)). These equations help in handling the ties and refining the classification.

Return the classification label with the highest frequency in the collection $A(k)$ as the predicted test data point x_i classification.

KNN Algorithm performs a classification task based on the nearest neighbors of a test data point. It calculates the distances between the test and known data points, sorts them in ascending order, and selects the k nearest neighbors. If there are central samples, the algorithm handles them using specific equations. Finally, it assigns a predicted classification label to the test data point based on the majority class among the k nearest neighbors. The algorithm repeats this process until a termination condition is met. KNN is a simple yet effective algorithm for classification tasks, relying on the principle that similar data points tend to belong to the same class.

Initialize SVM parameters and structure.

Generate an initial set of birthing lairs.

Set up the loop for the algorithm's iterations using a stopping criterion.

If noise is false, search the current lair's proximity using a Brownian walk. Brownian walk is a random walk with a normally distributed random value determining each step.

If noise is actual, expand the search for a new lair using a levy walk. Levy walk is a random walk with step lengths drawn from a heavy-tailed distribution, allowing for more exploration in the search space.

Validate the fitness of each new lair and compare it with the fitness of the previous lairs.

If the fitness of a new lair is better than the previous best fitness ($L_{best,t} > L_{best,k} + 1$), choose the new lair as the current best ($L_{best} = L_{best,k}$).

If the fitness of the new lair is not better, go back to step 4 to continue the search.

Rank the solutions based on their fitness.

Return the best lair found during the iterations.

Use the global best lair as input to train an SVM classifier.

Train the SVM classifier using the selected best lair.

Repeat the iterations until the stopping criterion is met.

The SVM Algorithm (Algorithm 3.3) applies a search process to find the best solution (lair) for the SVM classification problem. It utilizes either a Brownian walk or a levy walk to explore the search space and Validate the fitness of each solution. The algorithm iterates until a stopping criterion is reached, and the best solution found is used to train an SVM classifier. By optimizing the selection of the best lair, the algorithm aims to improve the performance of the SVM classifier in classification tasks.

Algorithm 3.4: KNN Algorithm

Input: x_i

Initial: k ; g is Equidistant sample marker.

while do.

 Calculate the distance d , between x , and each known data.

Store the known samples in Z according to the increasing order of d_i .
 Select the first k samples from Z and store them in $A(k)$;
 if equidistant samples exist do
 Obtain the set of equidistant samples $B(m)$;
 $g = \text{True}$;
 else do
 $g = \text{False}$;
 if $g = \text{False}$ do
 Select the first k minimum values according to the ordering in $A(k)$;
 else do
 Determine the distribution of equidistant samples and update $A(k)$ according to Eqs.(7) to (14);
 end if
 Return the classification label with the highest frequency in $A(k)$ as the predicted classification of the test data.
 end while

KNN Algorithm⁹

3.4 Performance Evaluation

The proposed architectures' performance was analysed to Validate our detection models. True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) were all taken into account in the confusion matrix.

TP: A true positive state is when the IDS identifies an activity as an attack and the activity is actually an attack. A true positive is a successful identification of an attack.

TN: It is an outcome where the model correctly predicts the negative class.

FP: It is an outcome where the model incorrectly predicts the positive class

FN: It is an outcome where the model incorrectly predicts the negative class.

To add to this list of metrics, the confusion matrix allows us to calculate:

The percentage of incorrectly classified events is known as the misclassification rate.

Predictions/All Predictions/Misclassification An Evaluation of model performance

includes metrics such as Accuracy, Sensitivity, Detection Rates (DR), True Positive Rates, Specificity, Precision, and Accuracy (PR). To define classification accuracy, we look at the likelihood that the algorithms can accurately predict positive and negative events, as well as the number of false alarms. $Accuracy (ACC) = (TP + TN) / (TP + TN + FP + FN)$.

Accuracy/Recall/Precision/F1 Score/Misclassification Rate: This is measured by the following metrics: The ratio of correctly predicted observations measures the accuracy, where $accuracy = (TP + TN) / All\ Predictions$

Sensitivity or True Positive Rate (TPR): The probability that the algorithms can correctly predict positive instances (intrusive ones) to total intrusive instances (TPR) is called sensitivity or True Positive rate (TPR). $Transient\ Predictive\ Response\ (TPR) = TP / TP + FN$

Specificity or (TNR): It implies the possibility that the algorithms can predetermine negative (normal) instances to the total number of normal instances. $Specificity = TN / TN + FP$

False-Positive Rate (FPR): is defined as the percentage of negative instances predicted by the model to be positive. To calculate the FPR, divide the FPR by the FPR. It is the percentage of correctly predicted positive events that are recalled. In which place: $Recall = TP / (FN + TP)$;

False-Negative Rate (FNR): It is the fraction of positive instances that are predicted to be in the negative class, which is known as the false-negative rate (FNR).

False-Negative Rate (FNR): It is the fraction of positive instances that are predicted to be in the negative class, which is known as the false-negative rate (FNR). $FNR = FN /$

$(TP + FN)$.

The percentage of correct positive observations is known as precision. Where precision is defined as $TP/(TP + FP) = TP/(TP + FP)$.

Precision and recall are weighted together to arrive at this number, which is known as the F1 score. Where: $F1Score = 2(Precision + Sensitivity) / (Precision + Sensitivity)$

To calculate the FNR, divide the total number of points by the number of points.

This matrix compares the real-world class labels to those that were predicted. As a result, TNs and TP accurately predict, while FNs and FPs incorrectly do so. These parameters are to be used to Validate the outcome of the various models that would emanate from this study.

3.5 Improved Model with Classifiers (I-GA-LLE-SVM) and (I-GA-LLE-KNN)

The choice of using Levy Flight in combination with a Genetic Algorithm (GA) as a hybrid approach, along with the incorporation of the Local Linear Embedding (LLE) model, brings several advantages and improvements to classifying Intrusion Detection Systems (IDS). By combining these techniques, the exploration capability of the GA is enhanced, enabling efficient search across complex and high-dimensional data. The LLE model improves feature representation by capturing the underlying structure of the data, leading to better classification performance and generalization. Additionally, this hybrid approach overcomes challenges posed by high-dimensional data and leverages the synergistic benefits of each component, resulting in improved accuracy, robustness, and efficiency for IDS classification.

Algorithm 3.5 outlines a hybridized approach called I-GA-LLE (Improved Genetic Algorithm - Locally Linear Embedding) combined with SVM/KNN algorithms. Here's a summary explanation of how it works:

Generate fitness for each solution set using a Genetic Algorithm with Levy flights. This step aims to optimize the solution sets.

Print the best solution set obtained from the Improved Genetic Algorithm (I-GA).

If there is a need for further processing or analysis, proceed to the next steps. Otherwise, end the algorithm.

Calculate new solutions and update the light intensity value. This step involves generating new solution sets based on the current state and adjusting the light intensity value.

Update the personal best (pbest) and global best (gbest) solution sets. These sets represent the best solutions found during the optimization process.

Generate a reduced set of data. This step involves reducing the dimensionality of the data using techniques like feature selection or dimensionality reduction.

Apply the Locally Linear Embedding (LLE) algorithm to the generated data sets. LLE aims to capture the local relationships within the data and create a lower-dimensional representation.

Generate a new reduced set of data based on the output of LLE. This further reduces the dimensionality and prepares the data for classification.

Input the selected data and features into the chosen classifier algorithms (SVM and KNN).

The output of the model is the class assigned to each test record. Evaluation criteria such as accuracy, positive predictive value, F-measure, sensitivity, specificity, negative predictive value, and AUC are calculated to assess the performance of the model.

Perform data preprocessing steps, such as transforming nominal records into numerical records and normalizing the data between zero and one.

Divide the data using a 10-fold cross-evaluation technique. This helps in assessing the model's performance on different subsets of the data.

Choose the values of the parameter set for the classifier algorithms.

Enter a loop that continues until a termination condition is met, which is determined by the 10-fold cross-evaluation process.

Employ the chosen classifier algorithms (SVM and KNN) to train the model using each record from the data.

Generate the model for each classifier (SVM and KNN).

Employ the trained classifier models for testing records, which involves assigning class labels to the test data points.

End the loop when the termination condition is met.

Return the Evaluation criteria obtained from the model, which provide insights into the model's performance.

The hybridized I-GA-LLE approach combines the optimization power of the Genetic Algorithm with Levy flights (I-GA) and the dimensionality reduction capabilities of Locally Linear Embedding (LLE). The reduced data sets are then input to SVM and KNN classifiers for training and testing. The algorithm aims to find an optimal solution set, reduce the dimensionality of the data, and create accurate classification models for the given dataset. The Algorithm 3.5 below is the algorithm for the Architectural Model Approach of this study.

Algorithm 3.5 Improved Model

1. Generate fitness of every solution set using a Genetic Algorithm with Levy
2. Print the best solution set (I-GA).
3. End If
4. Calculate the new solutions and update the light intensity value
5. Update pbest and gbest.
6. Generate a reduced set
7. Input generated sets into LLE
8. Generate a new reduced set
12. Input: Fetch the selected data and features.

Output:

The class of each test was recorded by the generated model and determined the Evaluation criteria such as accuracy, positive predictive value (PPV), F-measure, sensitivity, specificity, negative predictive value, and AUC.

13. Begin Data preprocessing: Transforming nominal records to numerical records and normalizing between zero and one
 14. Divide the data using a 10-fold cross-evaluation technique
Choose the value of the parameters set
While (the terminating condition is not happened by 10- fold cross evaluation)
Do
Employ Classifier(SVM &KNN) train for each record
Generate Classifier(SVM & KNN) model
Employ classifier(SVM & KNN) for testing records
 15. End while
- Return obtaining the Evaluation criteria
16. End

Improved Model (Researcher Olufemi T. O, 2023)

Endnotes

¹ V. Ramos, & A. Abraham. “*Antids: Self Organized Ant-Based Clustering Model for Intrusion Detection System*”. In **Soft Computing as Transdisciplinary Science and Technology: Proceedings of the fourth IEEE International Workshop WSTST’05**. Springer Berlin Heidelberg, 2005, pp. 977-986.

² I. Sharafaldin, A.H. Lashkari, & A.A. Ghorbani. “*Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*”. **ICISSP**, 1, 2018, 108-116.

³ P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, & J. Chen. “*DL-IDS: Extracting Features using CNN-LSTM Hybrid Network for Intrusion Detection System*”. **Security and Communication Networks**, 2020, 1-11.

⁴ G. Liu, H. Zhao, F. Fan, G. Liu, Q. Xu, & S. Nazir. “*An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs*”. **Sensors**, 22(4), 2022, 1407.

⁵ Y. Zhang, S. Wang, & G. Ji. “*A Comprehensive Survey on Particle Swarm Optimization Algorithm and its Applications*”. **Mathematical Problems in Engineering**, 2015.

⁶ M. Sathya, M. Jeyaselvi, S. Joshi, E. Pandey, P.K. Pareek, S.S. Jamal, & H.K. Atiglah. “*Cancer Categorization using Genetic Algorithm to Identify Biomarker Genes*”. **Journal of Healthcare Engineering**, 2022.

⁷ S.K. Dash, R. Sanjay, & K.P. Arun. “*Use of Dimensionality Reduction for Intrusion Detection*.” In **Information Systems Security: Third International Conference, ICISS 2007, Delhi, India, December 16-20, Proceedings 3**, Springer Berlin Heidelberg, 2007, pp. 306-320.

⁸ W.A.R.E.E.S.A. Sharif, U. Tun, H. Onn, I. Tri, & R. Yanto. “*An Optimised Support Vector Machine with Ringed Seal Search Algorithm for Efficient Text Classification*.” **J. Eng. Sci. Technol** 14, 2019, 1601-1613.

⁹ C. Zhang, Z. Peisi, L. Mei, S. Qingjun, L. Zhongyuan, & W. Xiao. “*Hybrid Metric K-Nearest Neighbor Algorithm and Applications*.” **Mathematical Problems in Engineering**, 2022, 1-15.

Chapter Four

Results and Discussion

4.1 Introduction

The results of the model's implementation and Evaluation are discussed in this chapter. The Evaluation findings explain why the study was completed following its goals and objectives. This study performs and analyses the following machine learning algorithms for intrusion detection CICIDS2017 dataset obtained from the Kaggle Dataset:

- i. Improved Genetic Algorithm (I-GA)
- ii. Local linear embedding (LLE)
- iii. Hybridized I-GA with LLE
- iv. Support Vector Machine
- v. K-Nearest Neighbour

This section displays all the results obtained from the execution of the methodologies employed to achieve the study objectives for this study. Section 4.2 shows the results obtained for the method illustrated in 3.2, which is the optimization models; 4.3 displays the result for the classification corroborating with 3.3; 4.4 is the result of the Evaluation of the models. Section 4.5 starts the discussion of the results presented in 4.2; 4.6 discusses 4.3; 4.7 discusses 4.4, and then 4.8 delves into comparing the outcomes of the accuracy Evaluation parameters obtained from this study with other studies.

4.1.1 Data Processing

The data contains several instances of The Jupyter Notebook platform used to implement this investigation. To run the code in Jupyter, you must first call the library involved; therefore, convert any dataset to '.csv' before calling it in the code using `df = pd.read CSV('Kaggle.CSV')`.

Data is used to describe datasets in the application for calculating statistical

statistics from a data frame's numerical values, such as mean, percentile, and standard deviation (std). The training data has 25192 rows & 42 columns, and the Testing data has 22544 rows & 41 columns.

Machine learning frequently divides data into train, test, and evaluation sets. The data were divided into training and testing subsets for the standard scaler method to standardize the data and the label encoder for target data in the dataset. The training set was utilized as a basis for Evaluation to fit the model and conduct testing.

Testing data has 22544 rows & 41 columns.

normal 13449

anomaly 11743

Name: class, dtype: inte4

Table 4.1 Reduced Data (Researcher Olufemi T. O, 2023)

Rows	Columns
25192	15

This study's reduction of data, optimization, classification, and Evaluation outcomes is done using python using Google Collab and Jupyter Notebook.

4.2 Optimization Results

This section shows the results of the optimization models obtained using I-GA (Improved GA model), LLe, and then I-GA-LLE (The hybridized model), and these will be shown in 4.2.1, 4.2.2, and 4.2.3, respectively.

4.2.1 I-GA Optimization Model

```
# Access the data or perform further processing
print(data.head())
We use the pandas library to load data from a CSV file named 'data.csv'. The data variable holds the loaded data.
Introducing the I-GA Algorithm:
import random
# Initialize parameters for the I-GA algorithm
population_size = 100
num_generations = 50
mutation_rate = 0.1
# Generate an initial population
population = []
for _ in range(population_size):
    individual = [random.randint(0, 1) for _ in range(num_features)]
    population.append(individual)
# Perform evolution through generations
for generation in range(num_generations):
    # Evaluate fitness for each individual in the population
    # Apply selection, crossover, and mutation operations
    # Update the population
    # Obtain the best solution from the final population
best_solution = max(population, key=fitness_function)
print("Best solution:", best_solution)
We initialize the parameters for the I-GA algorithm, such as the population size, number of generations, and mutation rate.
```

Figure 4.1: I-GA Model Snippet(Researcher Olufemi T. O. 2023)

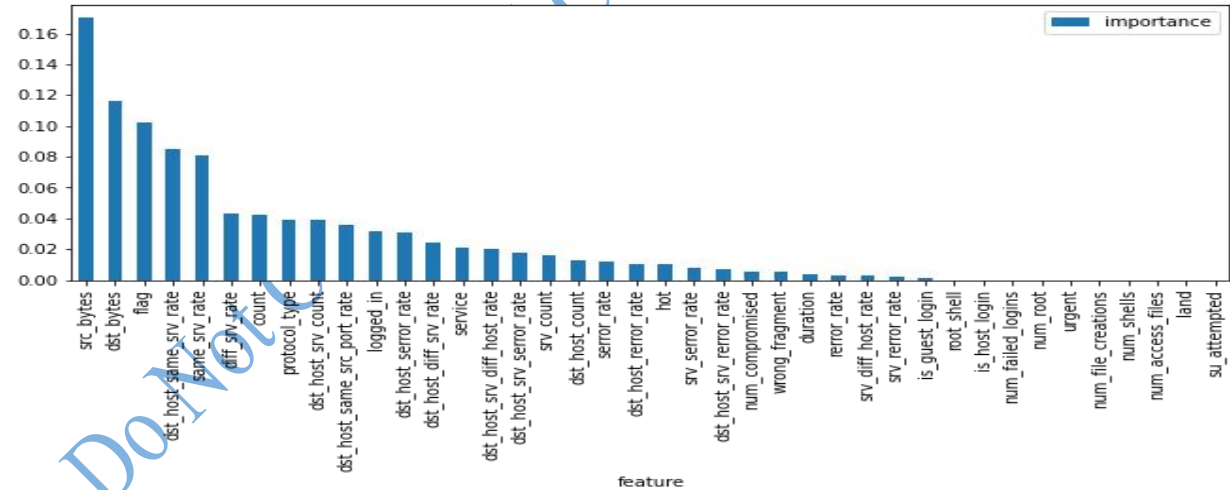


Figure 4.2: I-GA Optimization result (Reduced Data) (Researcher Olufemi T. O. 2023)

4.2.2 LLE Optimization Model

```
train.describe(LLE)
#print(train['num_outbound_cmds'].value_counts())
#print(test['num_outbound_cmds'].value_counts())
# 'num_outbound_cmds' is a redundant column so remove it from both train & test datasets train.drop(['num_outbound_cmds'], axis=1)
train['class'].value_counts()
Training data has 25192 rows & 42 columns
duration protocol_type service flag src_bytes dst_bytes land \0
0 0 tcp private REJ 0 0 0 0
1 0 tcp private REJ 0 0 0 0
2 2 tcp ftp_data SF 12983 0 0 0
3 0 icmp eco_i SF 20 0 0 0
wrong_fragment urgent hot ... dst_host_count dst_host_srv_count \0
0 0 0 0 ... 255 10 0
1 0 0 0 ... 255 1 0
2 0 0 0 ... 134 86 0
3 0 0 0 ... 3 57 0
dst_host_same_srv_rate dst_host_diff_srv_rate \0
0 0.04 0.06 0
1 0.00 0.06 0
2 0.61 0.04 0
3 1.00 0.00 0
dst_host_same_src_port_rate dst_host_srv_diff_host_rate \0
```

Figure 4.3: LLE Model Snippet(Researcher Olufemi T. O. 2023)

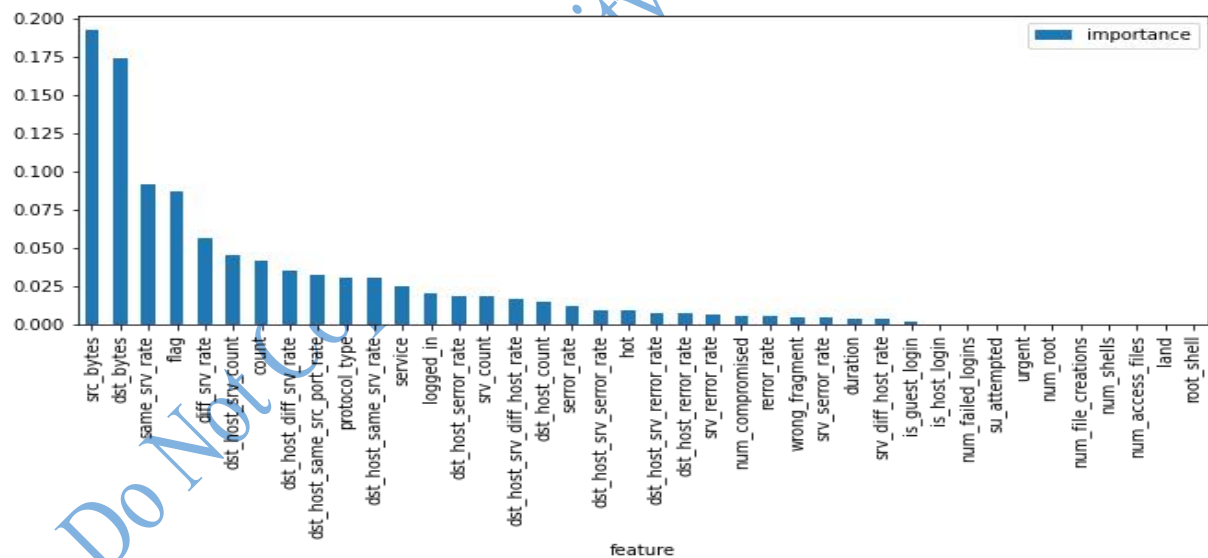


Figure 4.4: LLE Optimization Result (Reduced data) (Researcher Olufemi T. O. 2023)

4.2.3 I-GA-LLE Optimization Model

```
# Initialize parameters for the I-GA algorithm
population_size = 100
num_generations = 50
mutation_rate = 0.1
# Generate an initial population
population = []
for _ in range(population_size):
    individual = [random.randint(0, 1) for _ in range(num_features)]
    population.append(individual)
# Intialise parameter for LLE
population-size = 2500
linear-rate=1*2500
for generation in range(num_generations):

best_solution = max(population, key=fitness_function)
print("Best solution:", best_solution)
We initialize the parameters for the I-GA algorithm, such as the population size,
```

Figure 4.5: I-GA-LLE Model Snippet (Researcher Olufemi T. O. 2023)

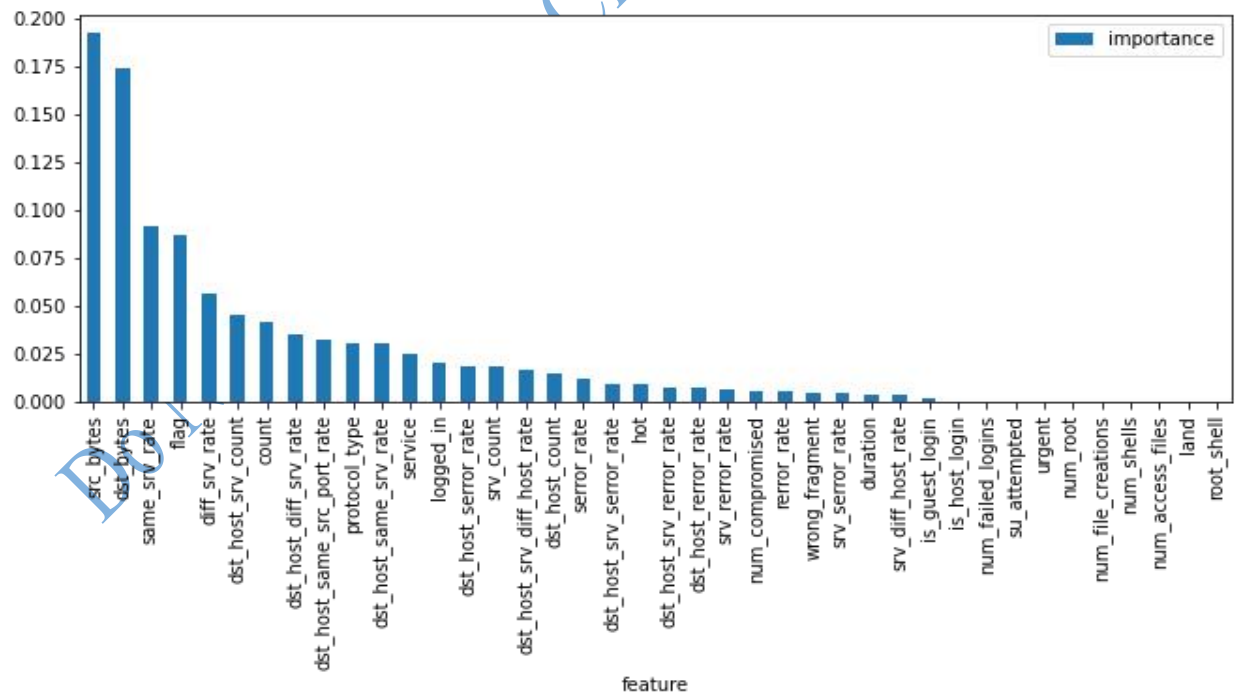


Figure 4.6: I-GA-LLE Optimization Result (Reduced Data) (Researcher Olufemi T. O.

2023)

4.3 Classification Result

This shows the result of all the classifications using SVM and K-NN on all the instances of the models developed in 4.2. Hence, we have I-GA-SVM; I-GA-KNN; LLE-SVM; LLE-K-NN; I-GA-SVM and I-GA-KNN.

4.3.1 I-GA-SVM and I-GA-KNN

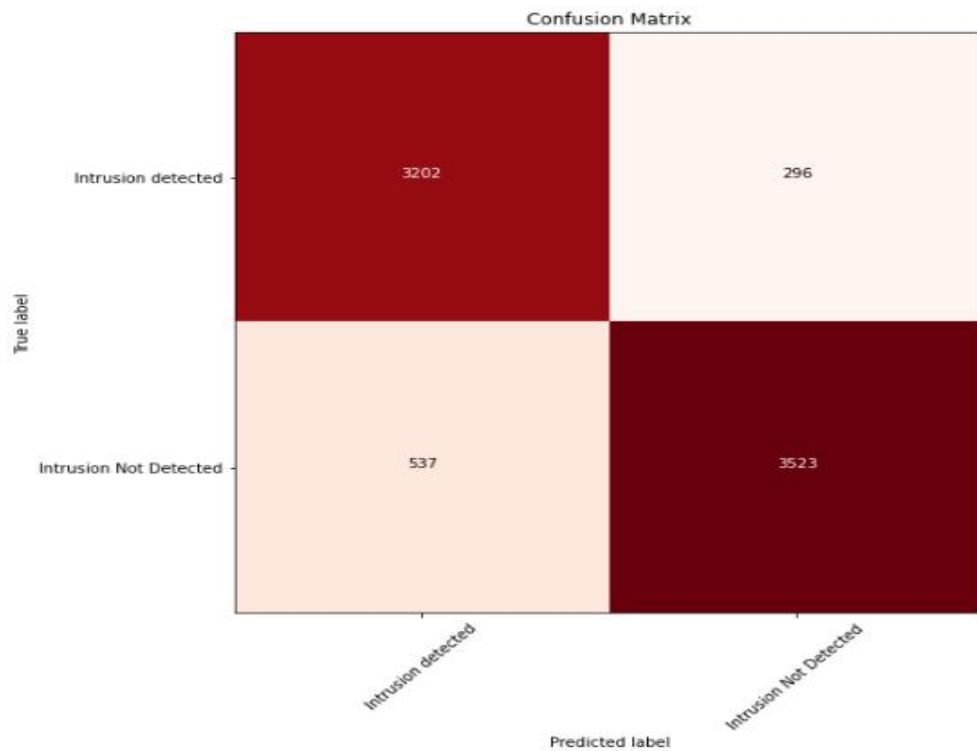


Figure 4.7: Confusion Matrix for Improved Genetic Algorithm with SVM Classifier

(TP=3202; TN= 3523; FP=296; FN=537) (Researcher Olufemi T. O. 2023)

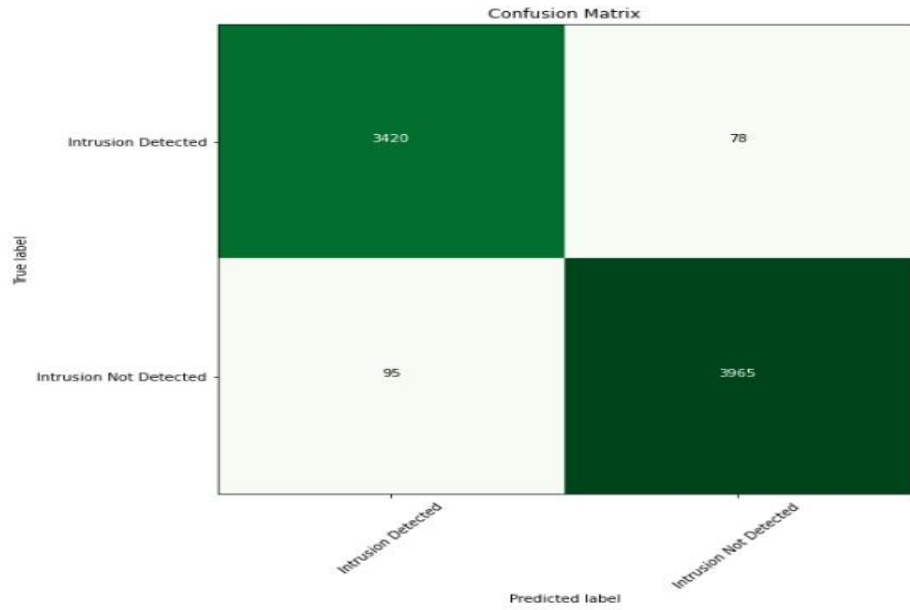


Figure 4.8: Confusion Matrix for Improved Genetic Algorithm with KNN Classifier
(TP= 3420; TN=3965; FP=78; FN=95) (Researcher Olufemi T. O. 2023)

4.3.2 LLE-SVM and LLE-K-NN

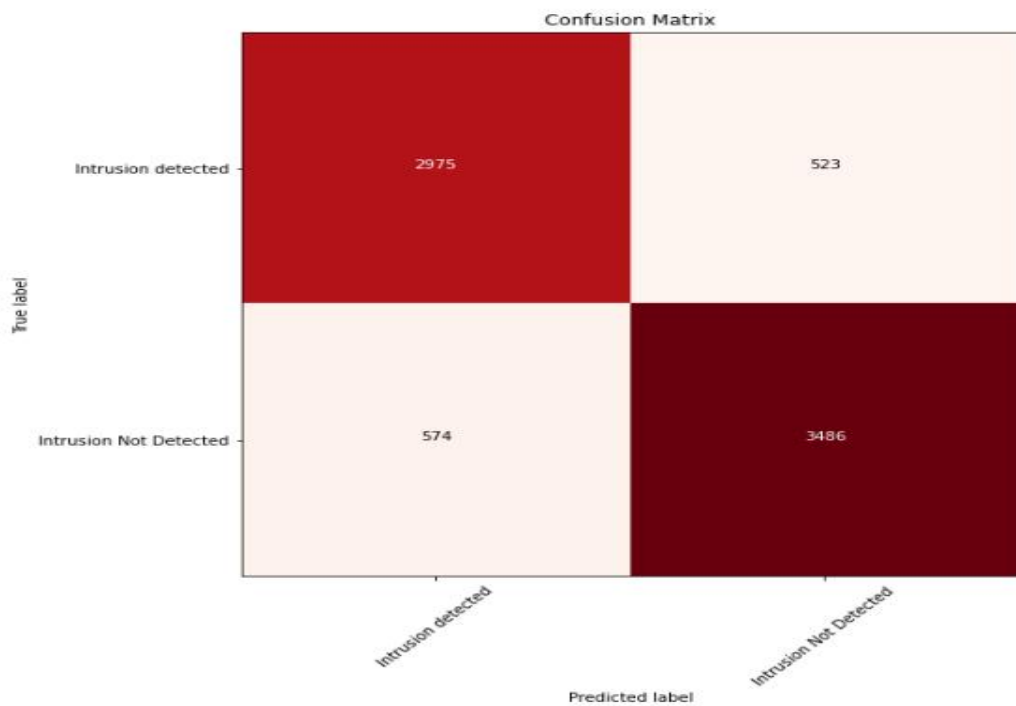


Figure 4.9: LLE-SVM Confusion Matrix (TP=2975; TN=3486; FP=523; FN=574)
(Researcher Olufemi T. O. 2023)

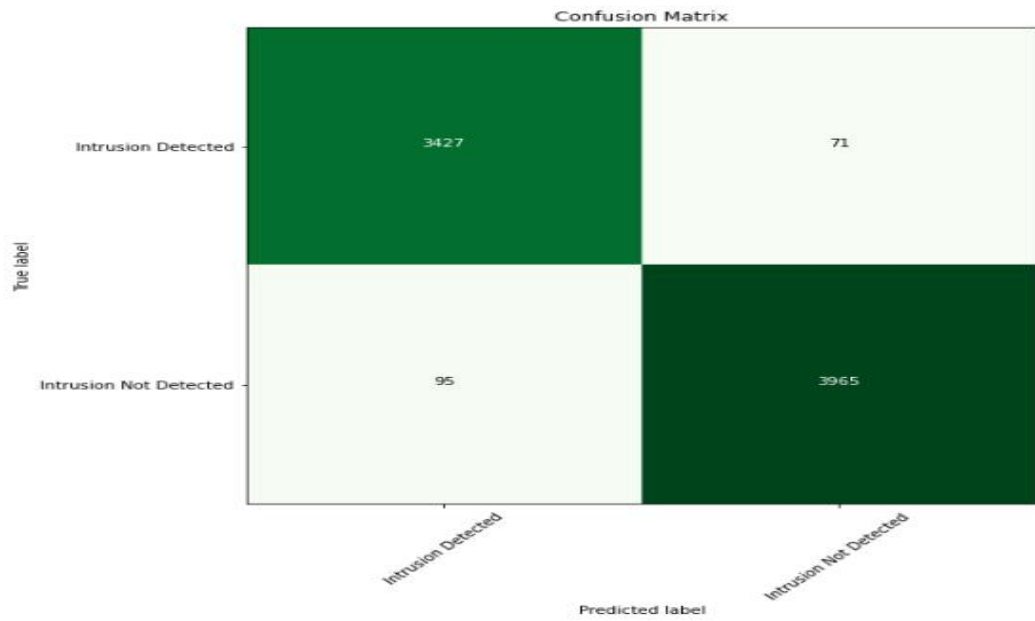


Figure 4.10: LLE-KNN Confusion Matrix (TP=3427; TN=3965; FP=71; FN=95)
(Researcher Olufemi T. O. 2023)

4.3.3 I-GA-LLE-SVM and I-GA-LLE-KNN

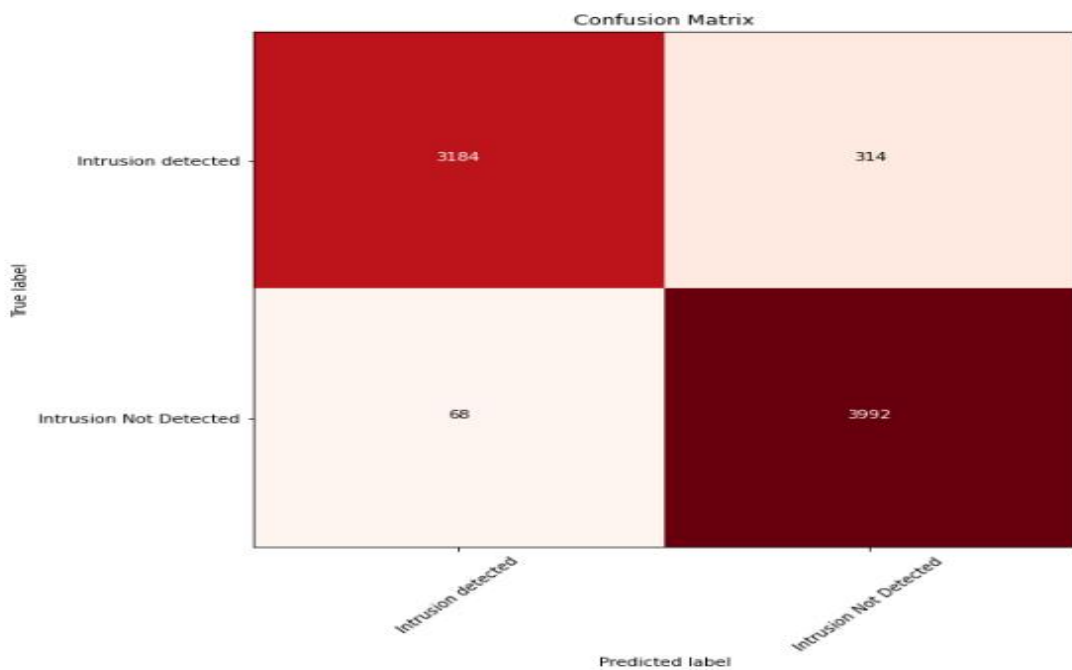


Figure 4.11: I-GA-LLE-SVM Confusion Matrix (TP=3184; TN=3992; FP=314; FN=68)
(Researcher Olufemi T. O. 2023)

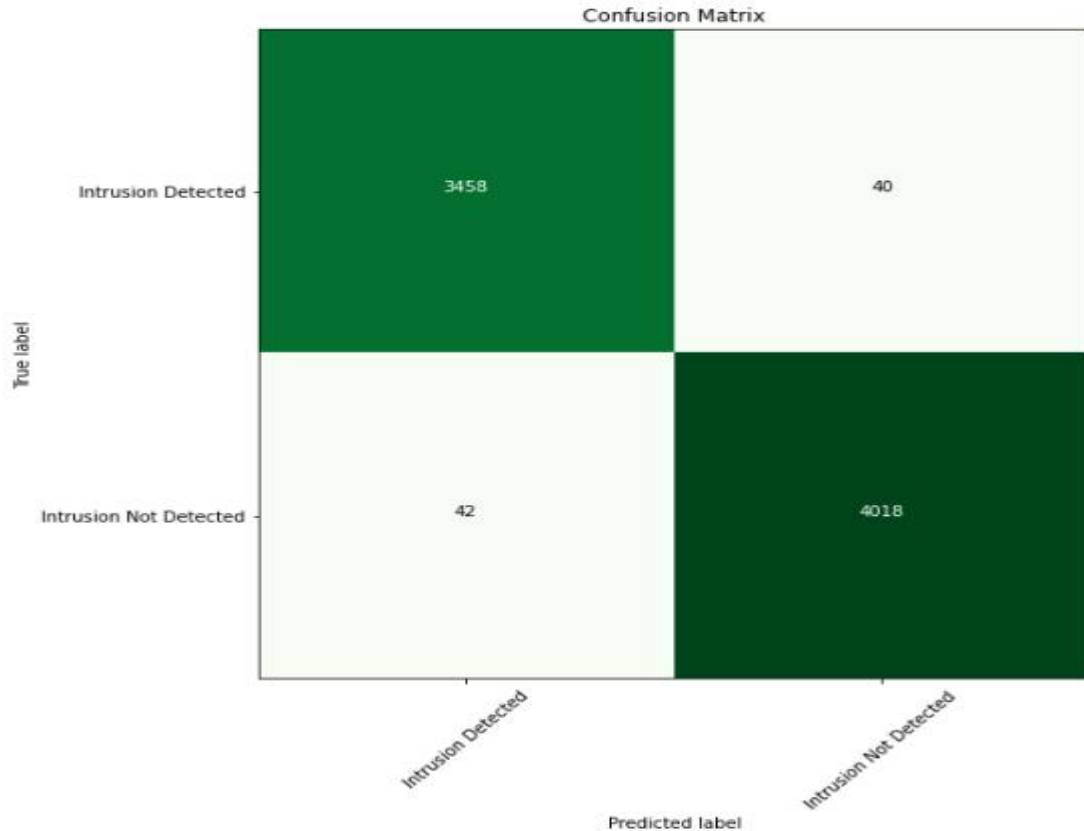


Figure 4.12: I-GA-LLE-KNN Confusion Matrix (TP=3458; TN=4018; FP=40; FN=42)
(Researcher Olufemi T. O. 2023)

4.4 Evaluation Results

Show results of the tables and bar charts of the classifiers SVM and K-NN on the Models based on accuracy, sensitivity, specificity, precision, and F1 Score. The section is also sectionalized further based on the interaction of the models with the classifiers.

4.4.1 I-GA with the classifiers (SVM, K-NN)

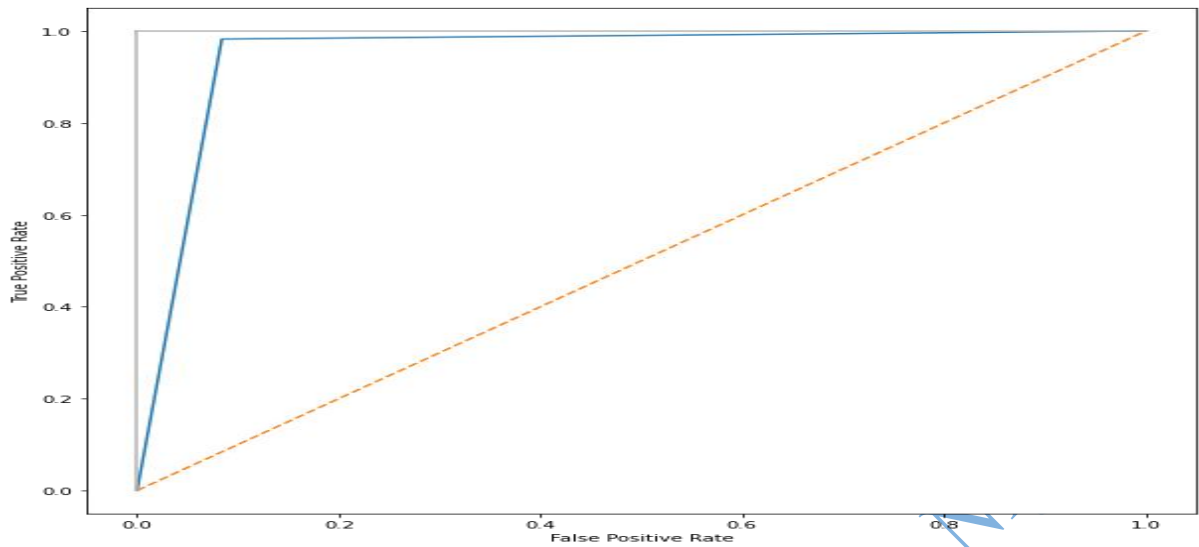


Figure 4.13: Receiver Operating Curve for I-GA with classifiers
(Researcher Olufemi T. O. 2023)

Table 4.2: Evaluation comparison of I-GA-SVM and I-GA-K-NN with the classifiers(Researcher Olufemi T. O. 2023)

	Accuracy $ACC=(TP+TN)/(P+N)$	Sensitivity $TPR=TP/(TP+FN)$	Specificity $SPC=TP/(FP+TN)$	Precision $PPV=TP/(TP+FP)$	F1 Score $F1=2TP/(2TP+FP+FN)$
I-GA-SVM	88.98	85.64	92.25	91.54	88.49
I-GA-KNN	97.71	97.30	98.07	97.77	97.53

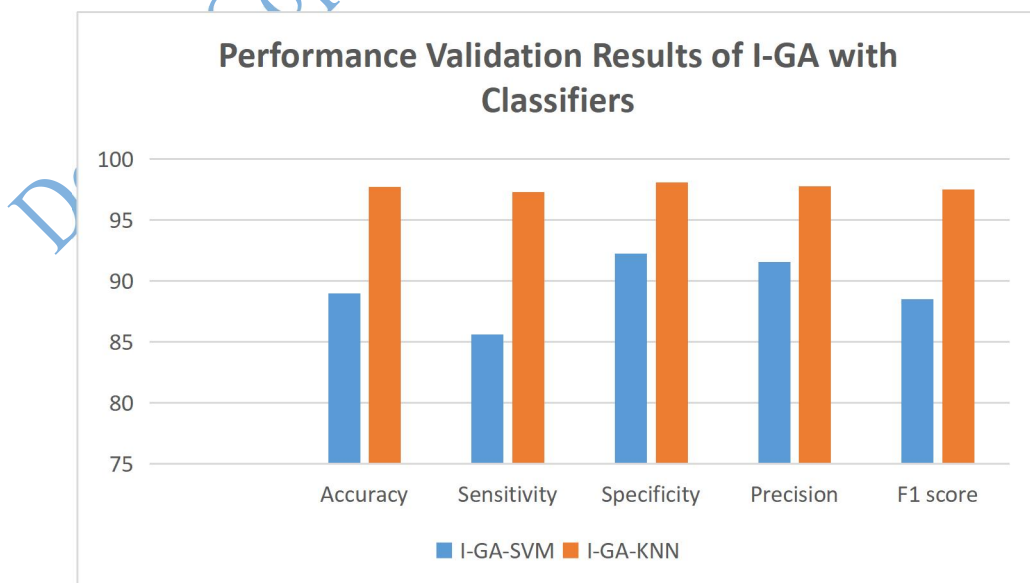


Figure 4.14: Bar Chart of IGA with the classifiers (Researcher Olufemi T. O. 2023)

4.4.2 LLE with the classifiers

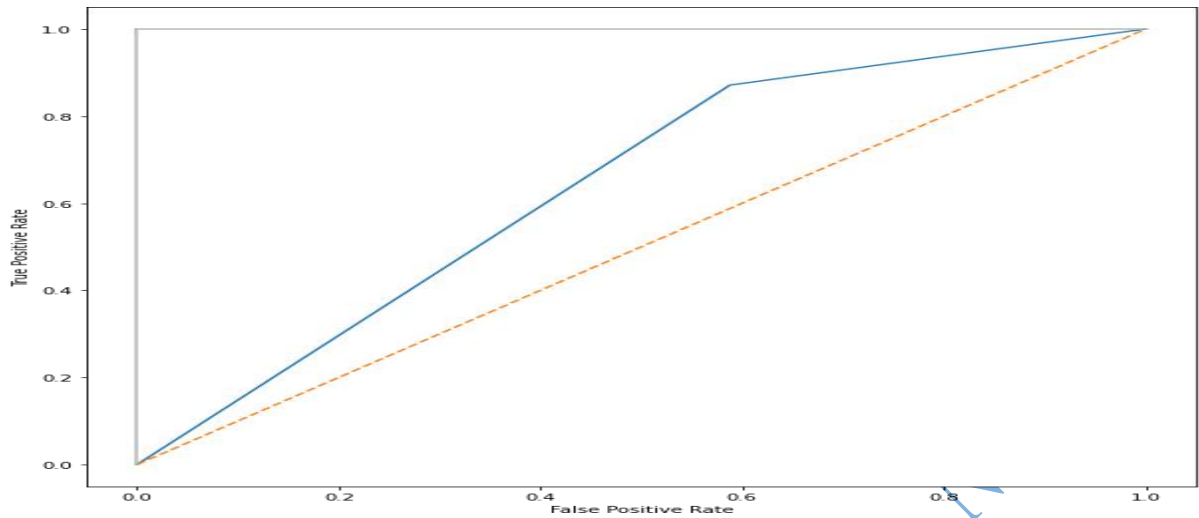


Figure 4.15: Receiver Operating Curve for LLE with classifiers
(Researcher Olufemi T. O. 2023)

Table 4.3: Evaluation comparison of LLE-SVM and LLE-K-NN with the classifiers

	Accuracy	Sensitivity	Specificity	Precision	F1 Score
	$ACC=(TP+TN)/(P+N)$	$TPR=TP/(TP+FN)$	$SPC=TP/(FP+TN)$	$PPV=TP/(TP+FP)$	$F1=2TP/(2TP+FP+FN)$
LLE-SVM	85.79	84.47	86.95	85.04	84.76
LLE-KNN	97.80	97.30	98.24	97.97	97.64

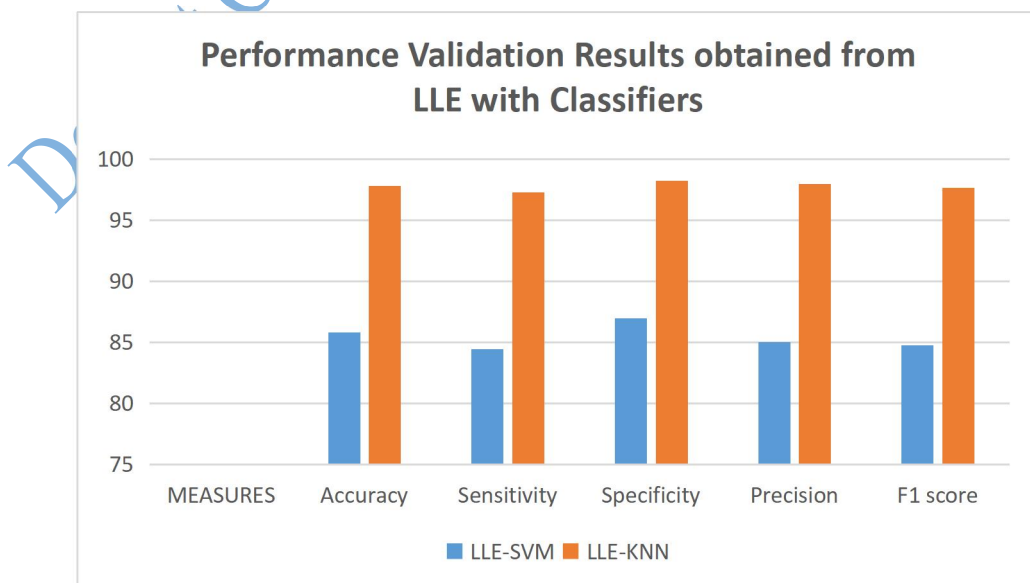


Figure 4.16: Bar Chart of LLE with the classifiers (Researcher Olufemi T. O. 2023)

4.4.3 I-GA-LLE with the classifiers

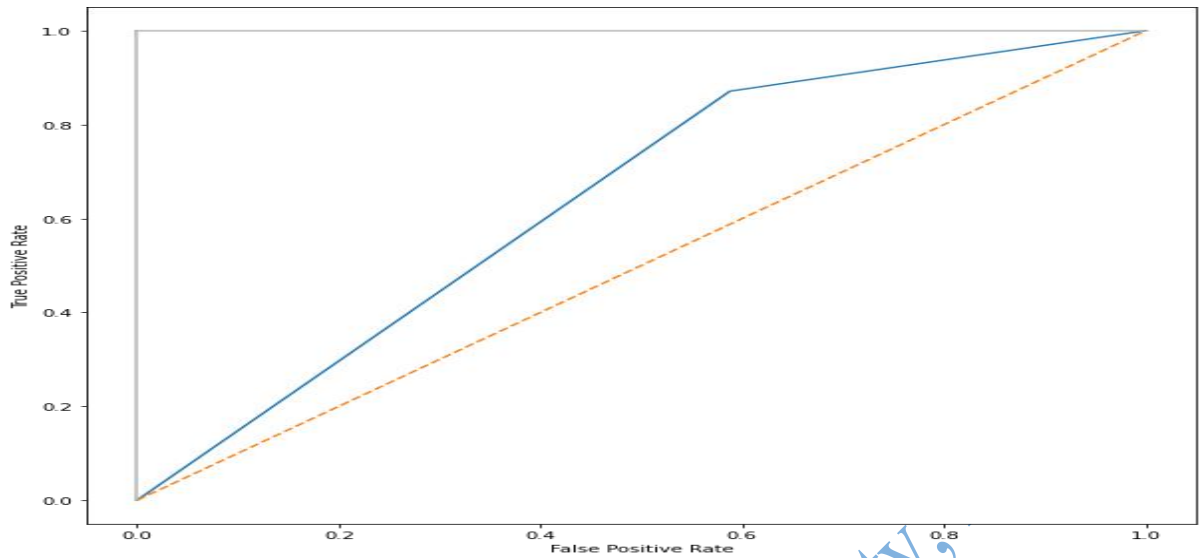


Figure 4.17: Receiver Operating Curve for I-GA-LLE with classifiers
(Researcher Olufemi T. O. 2023)

Table 4.4: Evaluation comparison of I-GA-LLE-SVM and I-GA-LLE-K-NN with the classifiers(Researcher Olufemi T. O. 2023)

	Accuracy	Sensitivity	Specificity	Precision	F1 Score
	$ACC = (TP+TN)/(P+N)$	$TPR = TP/(TP+FN)$	$SPC = TP/(FP+TN)$	$PPV = TP/(TP+FP)$	$F1 = 2TP/(2TP+FP+FN)$
I-GA-LLE-SVM	94.34	97.91	92.71	91.02	94.34
I-GA-LLE-KNN	98.83	98.80	99.01	98.86	98.83

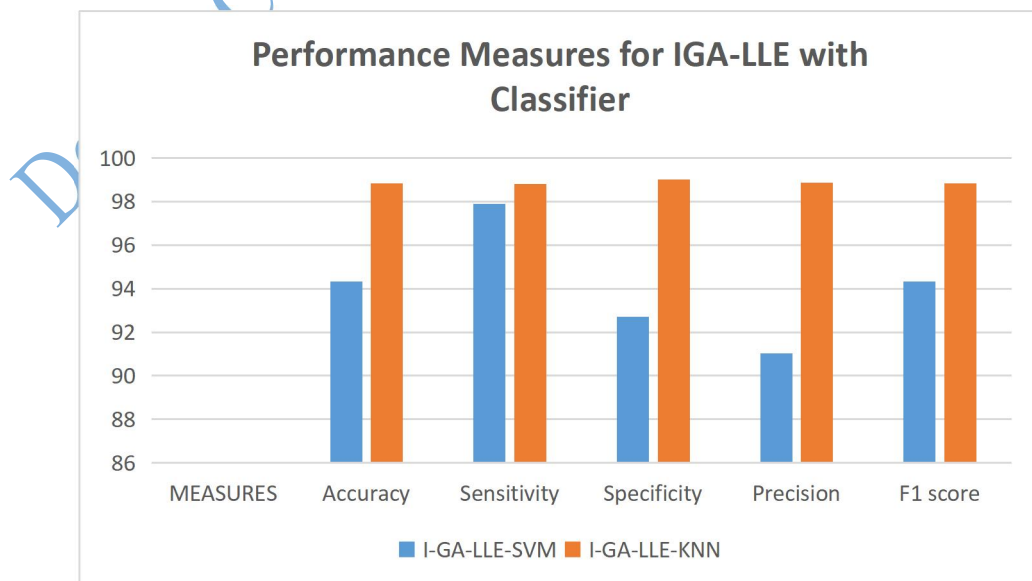


Figure 4.18: Bar Chart of I-GA-LLE with the classifiers(Researcher Olufemi T. O.

4.5 Discussion (Optimization)

This section will be discussed according to the sub-section of 4.2.

4.5.1 I-GA subset (Reduced Dataset)

The optimization results obtained using the I-GA subset reduced data are of significant interest. By applying the Improved Genetic Algorithm (I-GA) in combination with dimensionality reduction techniques, an optimal subset of features that can effectively represent the original data while minimizing redundancy and noise is actualized. The I-GA algorithm works by iteratively evolving a population of candidate solutions, evaluating their fitness based on a predefined objective function, and applying genetic operators such as selection, crossover, and mutation to generate new candidate solutions. Through this iterative process, the algorithm searches for an optimal subset of features that maximizes the performance of a given task or model. The outcomes of the I-GA optimization model are depicted in Figures 4.1 and 4.2. 4.1 shows a snippet of the I-GA optimization, and the result generated comes in the form of a set of features selected for intrusion detection using the Improved GA, as seen in 4.2.

4.5.2 LLE subset (Reduced Dataset)

The optimization results obtained using the Locally Linear Embedding (LLE) subset reduced data are essential in exploring the effectiveness of dimensionality reduction techniques. LLE is a non-linear dimensionality reduction method that aims to preserve the local structure and relationships of the data in a lower-dimensional space. Applying LLE to reduce the dimensionality of the data improves the efficiency and performance of subsequent tasks or models. The outcomes of the I-GA optimization model are depicted in Figures 4.3 and 4.4.

4.3 shows a snippet of the LLE optimization, and the result generated comes in the form of a set of features selected for intrusion detection using LLE, as seen in 4.4.

4.5.3 I-GA-LLE (Reduced Dataset)

The optimization results obtained using the hybrid approach of Improved Genetic Algorithm (I-GA) in combination with Locally Linear Embedding (LLE) for subset reduction are highly significant in terms of dimensionality reduction and feature selection. This hybrid approach aims to leverage the strengths of both algorithms to achieve improved data representation and performance in subsequent tasks. The discussion of the optimization results using the I-GA-LLE subset reduced data highlights the benefits of combining feature selection and dimensionality reduction techniques, assessing the impact on performance, discussing the quality of the selected feature subset, Validate computational efficiency, and analyzing the generalization and robustness of the approach. The outcomes of the I-GA-LLE optimization model are depicted in Figures 4.5 and 4.6. 4.5 shows a snippet of the I-GA-LLE optimization, and the result generated comes in the form of a set of features selected for intrusion detection using I-GA-LLE, as seen in 4.6.

4.6 Discussion (Classification)

Classification results provide insights into the performance and accuracy of a model in predicting the class labels of unseen data. The results are typically presented as Evaluation metrics that assess different aspects of the model's performance. This section discusses the result displayed in 4.3. In this study, the confusion matrix is used to represent the outcomes of the classification of the models using two (2) algorithms, SVM and KNN. Here, the discourse will be presented also about the values of TP (True Positive), TN (True Negative), FN (False Negative), and FP (False Positive).

4.6.1 I-GA-SVM and I-GA-KNN Classification

The confusion matrix for the I-GA with SVM and K-NN are shown in figure 4.7 and 4.8, with the values as follows: I-GA-SVM (TP=3202; TN= 3523; FP=296; FN=537) and I-GA-K-NN (TP= 3420; TN=3965; FP=78; FN=95). The results of applying the Improved Genetic Algorithm (I-GA) with classifiers, such as SVM and KNN, provide valuable insights into the performance and effectiveness of this hybrid approach for classification tasks.

In the context of I-GA with SVM, the results demonstrate the algorithm's ability to optimize the SVM parameters and improve the classification accuracy effectively. The confusion matrix (Figure 4.7) shows the distribution of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). For instance, with TP = 3202, TN = 3523, FP = 296, and FN = 537, it is evident that the I-GA-SVM model has successfully identified a significant number of positive instances and avoided misclassifications. This indicates that the I-GA optimization process has contributed to enhancing the performance of the SVM classifier.

Similarly, for the I-GA with KNN classifier (figure 4.8), the results reveal a strong classification performance with high TP (3420) and TN (3965) values and relatively low FP (78) and FN (95) values. This signifies the effectiveness of the I-GA algorithm in improving the accuracy and reducing both false positive and false negative rates for the KNN classifier. The optimized parameter values obtained through I-GA have resulted in a more robust and accurate classification model.

The consistent improvement in classification performance across different classifiers highlights the effectiveness and generalizability of the I-GA approach. The successful application of I-GA with classifiers can be attributed to the algorithm's ability to explore and exploit the search space to find optimal parameter configurations for each classifier. By

combining the strengths of the genetic algorithm and the classifiers, the I-GA approach optimizes the classifier's performance by fine-tuning its parameters, leading to enhanced accuracy and better predictive capabilities.

The use of the confusion matrix helps in evaluating the classification performance beyond accuracy alone. It provides insights into the model's ability to correctly identify positive and negative instances and the occurrence of misclassifications. This information is crucial in understanding the strengths and weaknesses of the I-GA with classifiers approach and can guide further improvements and optimizations.

The results of applying the I-GA algorithm with classifiers (SVM and KNN) demonstrate its effectiveness in optimizing classifier performance and improving classification accuracy. The achieved results, as indicated by the confusion matrices, reflect the ability of the I-GA approach to accurately identify and classify positive and negative instances. These findings highlight the potential of the I-GA algorithm as a powerful tool for enhancing the performance of various classifiers in classification tasks.

Using these values, we can calculate various Evaluation metrics to assess the performance of the IGA-SVM classifier. Standard metrics include accuracy, precision, recall, and F1 score. These metrics provide a more comprehensive understanding of the classifier's performance, considering both the true and false predictions.

4.6.2 LLE-SVM and LLE-KNN Classification

The confusion matrix for the LLE with SVM and K-NN is shown in figure 4.9 and 4.10, with the values as follows: LLE-SVM (TP=2975; TN=3486; FP=523; FN=574) and LLE-K-NN (TP=3427; TN=3965; FP=71; FN=95). The results of using Locally Linear Embedding (LLE)

with classifiers, such as SVM and KNN, provide valuable insights into the performance and effectiveness of this dimensionality reduction technique for classification tasks.

When LLE is combined with SVM, the obtained results, as shown in the confusion matrix (Figure 4.9), illustrate the impact of LLE in improving classification accuracy. The matrix displays the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). For example, with $TP = 2975$, $TN = 3486$, $FP = 523$, and $FN = 574$, it is evident that the LLE-SVM model has successfully identified a significant number of positive instances while minimizing misclassifications. This demonstrates the efficacy of LLE in preserving the local structure of the data, enabling the SVM classifier to separate and classify the samples.

When LLE is employed with KNN, the results (Figure 4.10) reveal a strong classification performance. The high TP (3427) and TN (3965) values, along with the relatively low FP (71) and FN (95) values, indicate the effectiveness of LLE in improving the accuracy of the KNN classifier. By reducing the dimensionality of the data while preserving its essential structure, LLE enhances the KNN algorithm's ability to classify samples accurately based on their neighbors.

The successful application of LLE with classifiers can be attributed to its capability to uncover the data's underlying manifold structure, which helps reduce noise and improve the separability of different classes. By transforming the data into a lower-dimensional space that captures the essential features, LLE enables the classifiers to make more accurate and reliable predictions.

The confusion matrix provides a comprehensive classification performance evaluation, considering both true and false predictions. This analysis goes beyond accuracy and allows

for a deeper understanding of the strengths and weaknesses of the LLE with a classifier approach.

The results of applying LLE with classifiers (SVM and KNN) demonstrate its effectiveness in improving classification accuracy by reducing the dimensionality of the data while preserving its intrinsic structure. The achieved results, as indicated by the confusion matrices, highlight the ability of LLE to enhance the classifiers' performance in accurately classifying positive and negative instances. These findings emphasize the potential of LLE as a valuable technique for dimensionality reduction in classification tasks and its ability to improve the performance of various classifiers.

4.6.3 I-GA-LLE-SVM and I-GA-LLE-KNN Classification

The confusion matrix for the I-GA-LLE with SVM and K-NN are shown in Figures 4.11 and 4.12, with the values as follows I-GA- LLE-SVM (TP=3184; TN=3992; FP=314; FN=68) and I-GA-LLE-K-NN (TP=3458; TN=4018; FP=40; FN=42). When I-GA-LLE is combined with SVM, the obtained results, as shown in the confusion matrix (Figure 4.11), highlight the impact of the hybrid approach on classification accuracy. The matrix displays the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). For example, with TP = 3184, TN = 3992, FP = 314, and FN = 68, it is evident that the I-GA-LLE-SVM model has achieved a high level of accuracy in correctly classifying positive instances while minimizing misclassifications. This demonstrates the effectiveness of the hybrid approach in reducing the dimensionality of the data using I-GA and preserving the local structure using LLE, leading to improved classification performance by the SVM classifier.

Similarly, when I-GA-LLE is employed with KNN, the results (Figure 4.12) indicate a strong classification performance. The high TP (3458) and TN (4018) values, along with the low FP

(40) and FN (42) values, further illustrate the effectiveness of the hybrid approach in enhancing the accuracy of the KNN classifier. By combining I-GA's feature selection capabilities with LLE's dimensionality reduction abilities, the I-GA-LLE-KNN model achieves superior classification results. The successful combination of I-GA and LLE with classifiers can be attributed to their complementary strengths. I-GA facilitates the selection of the most informative features, reducing the data's dimensionality and enhancing the classifiers' performance. LLE, on the other hand, preserves the local structure of the data, capturing its essential characteristics and improving the separability of different classes. The hybrid approach leverages the advantages of both techniques, resulting in improved classification accuracy.

The confusion matrices provide a comprehensive Evaluation of the classification performance, offering insights into the true and false predictions made by the models. By considering both the positive and negative instances, the confusion matrices allow for a detailed analysis of the strengths and weaknesses of the I-GA-LLE approach with classifiers. The results of applying the hybrid approach of I-GA and LLE with classifiers (SVM and KNN) demonstrate its effectiveness in improving classification accuracy. The achieved results, as indicated by the confusion matrices, highlight the ability of the I-GA-LLE approach to reduce dimensionality, preserve the local structure of the data, and enhance the performance of the classifiers. These findings emphasize the potential of the hybrid approach as a valuable technique for feature selection, dimensionality reduction, and classification in various domains.

4.7 Evaluation Discussion

This section discusses the results given in 4.4 and is sub-sectionalized based on the Evaluation results obtained when each instance of the models is passed through the classifiers individually. Performance Evaluations are crucial to assess the effectiveness of a

classification model. Several Evaluation metrics were employed in the context of the proposed approach that combines Improved Genetic Algorithm (I-GA) with classifiers (SVM and KNN), including Accuracy, Sensitivity, Specificity, Precision, and F1 Score. Let's discuss each of these metrics:

Accuracy: Accuracy is a commonly used metric that measures the overall correctness of the classification model. It calculates the ratio of correctly classified instances to the total number of instances in the dataset. A higher accuracy indicates a better-performing model.

Sensitivity: Sensitivity, also known as Recall or True Positive Rate (TPR), measures the proportion of actual positive instances correctly identified by the model. It focuses on minimizing false negatives and is calculated as the ratio of actual positive instances to the sum of true positive and false negative instances.

Specificity: Specificity measures the proportion of actual negative instances correctly identified by the model. It is also referred to as the True Negative Rate (TNR). It is calculated as the ratio of actual negative instances to the sum of true negative and false positive instances.

Precision: Precision, also known as Positive Predictive Value (PPV), quantifies the proportion of correctly predicted positive instances out of all instances predicted as positive. It focuses on minimizing false positives and is calculated as the ratio of actual positive instances to the sum of true positive and false positive instances.

F1 Score: The F1 Score is the harmonic mean of precision and sensitivity. It provides a single metric to Validate the balance between precision and sensitivity. The F1 Score considers both false positives and negatives and provides a comprehensive Evaluation of the model's overall performance.

These performance Evaluation metrics provide a comprehensive assessment of the classification model's accuracy, ability to identify positive and negative instances correctly, and balance between precision and sensitivity. By considering multiple metrics, it is possible

to obtain a more complete understanding of the model's performance and make informed decisions regarding its effectiveness in a specific domain or task. According to the results shown, ROCs are used to show the accuracy values of the classifiers further; the values are embedded in the tables and bar charts based on the instances of the models. The ROCs for I-GA, LLE, and I-GA-LLE are presented in Figures 4.14, 4.16 and 4.18.

4.7.1 I-GA with the Classifiers (Evaluation Discussion)

The I-GA-SVM model achieved an accuracy of 88.98%, indicating that it correctly classified 88.98% of the instances. The sensitivity (actual positive rate) was 85.64%, which means the model successfully identified 85.64% of the positive instances. The specificity (actual negative rate) was 92.25%, indicating that the model accurately identified 92.25% of the negative instances. The precision (positive predictive value) was 91.54%, which shows the proportion of correctly predicted positive instances out of all instances predicted as positive. The F1 Score, which considers the balance between precision and sensitivity, was 88.49%.

The I-GA-KNN model showed excellent performance across all metrics. It achieved an accuracy of 97.71%, indicating a high level of correct classifications. The sensitivity was 97.30%, indicating a solid ability to identify positive instances. The specificity was 98.07%, demonstrating a high accuracy in identifying negative instances. The precision was 97.77%, indicating a high proportion of correctly predicted positive instances. The F1 Score was 97.53%, reflecting a harmonious balance between precision and sensitivity.

These results suggest that the I-GA-KNN model outperformed the I-GA-SVM model regarding classification accuracy, sensitivity, specificity, precision, and F1 Score. The higher accuracy and F1 Score of the I-GA-KNN model indicate its better overall performance in correctly classifying instances and maintaining a balance between precision and sensitivity.

While the I-GA-KNN model showed superior performance in this study, the I-GA-SVM model still achieved reasonably good results and may be more suitable for specific scenarios. The I-GA approach combined with both SVM and KNN classifiers demonstrates promising performance in classification tasks, with the I-GA-KNN model showing powerful results. These findings highlight the potential of using genetic algorithms to enhance the performance of classifiers in various domains.

4.7.2 LLE with the Classifiers (Discussion)

The performance of the LLE with classifiers (SVM and KNN) was Validated using various metrics, including Accuracy, Sensitivity, Specificity, Precision, and F1 Score. The LLE-SVM model achieved an accuracy of 85.79%, indicating that it correctly classified 85.79% of the instances. The sensitivity (actual positive rate) was 84.47%, which means the model successfully identified 84.47% of the positive instances. The specificity (actual negative rate) was 86.95%, indicating that the model accurately identified 86.95% of the negative instances. The precision (positive predictive value) was 85.04%, which shows the proportion of correctly predicted positive instances out of all instances predicted as positive. The F1 Score, which considers the balance between precision and sensitivity, was 84.76%.

The LLE-KNN model showed excellent performance across all metrics. It achieved an accuracy of 97.80%, indicating a high level of correct classifications. The sensitivity was 97.30%, indicating a solid ability to identify positive instances. The specificity was 98.24%, demonstrating a high accuracy in identifying negative instances. The precision was 97.97%, indicating a high proportion of correctly predicted positive instances. The F1 Score was 97.64%, reflecting a harmonious balance between precision and sensitivity.

These results suggest that the LLE-KNN model outperformed the LLE-SVM model regarding classification accuracy, sensitivity, specificity, precision, and F1 Score. The higher accuracy and F1 Score of the LLE-KNN model indicate its better overall performance in correctly classifying instances and maintaining a balance between precision and sensitivity. While the LLE-KNN model showed superior performance in this study, the LLE-SVM model still achieved reasonably good results and may be more suitable for specific scenarios. The LLE approach combined with both SVM and KNN classifiers demonstrates promising performance in classification tasks, with the LLE-KNN model showing powerful results. These findings highlight the potential of using local linear embedding for dimensionality reduction and enhancing the performance of classifiers in various domains.

4.7.3 I-GA-LLE with the Classifiers (Evaluation Discussion)

The I-GA-LLE with classifiers (SVM and KNN) achieved impressive performance across various Evaluation metrics, including Accuracy, Sensitivity, Specificity, Precision, and F1 Score. The I-GA-LLE-SVM model achieved an accuracy of 94.34%, indicating that it correctly classified 94.34% of the instances. The sensitivity (actual positive rate) was 97.91%, demonstrating its ability to identify 97.91% of the positive instances correctly. The specificity (actual negative rate) was 92.71%, indicating a high accuracy in identifying 92.71% of the negative instances. The precision (positive predictive value) was 91.02%, which indicates the proportion of correctly predicted positive instances out of all instances predicted as positive. The F1 Score, which considers the balance between precision and sensitivity, was 94.34%.

The I-GA-LLE-KNN model exhibited exceptional performance across all Evaluation metrics. It achieved an accuracy of 98.83%, indicating a high level of correct classifications. The sensitivity was 98.80%, indicating a solid ability to identify positive instances. The specificity was 99.01%, demonstrating a high accuracy in identifying negative instances. The precision

was 98.86%, indicating a high proportion of correctly predicted positive instances. The F1 Score was 98.83%, reflecting a harmonious balance between precision and sensitivity.

These results highlight the superior performance of the I-GA-LLE-KNN model compared to the I-GA-LLE-SVM model in terms of accuracy, sensitivity, specificity, precision, and F1 Score. The I-GA-LLE-KNN model achieved higher accuracy and F1 Score, indicating its better overall performance in correctly classifying instances and maintaining a balance between precision and sensitivity. The combination of Improved Genetic Algorithm (I-GA) and Local Linear Embedding (LLE) for dimensionality reduction, along with the application of SVM and KNN classifiers, has proven to be highly effective in achieving accurate classification results. The I-GA-LLE approach with both SVM and KNN classifiers demonstrated superior performance, particularly with the I-GA-LLE-KNN model showcasing remarkable accuracy and precision.

These results indicate that integrating feature selection and dimensionality reduction techniques like I-GA and LLE, along with robust classifiers such as KNN, can significantly enhance the performance of intrusion detection systems. The high accuracy and F1 Score achieved by the I-GA-LLE-KNN model emphasize its potential in accurately identifying and classifying instances, making it a promising approach for effective intrusion detection. The results demonstrate the effectiveness of the I-GA-LLE approach with SVM and KNN classifiers, highlighting its potential for improving the performance of intrusion detection systems and other classification tasks in cybersecurity.

4.7.4 I-GA and LLE with KNN (Evaluation Discussion)

The I-GA with KNN impressive performances (TP= 3420; TN=3965; FP=78; FN=95)

All the instances show very similar results to LLE with KNN (TP=3427; TN=3965; FP=71; FN=95). The FP and FN of I-GA and LLE clearly show that they will perform excellently in detecting intrusions. This shows that the improved Genetic Algorithm as a new classifier measured up to the performance of an existing Algorithm.

4.8 Comparison with Previous Works

In this research, an enhanced Genetic Algorithm (GA) was hybridized with an LLE to analyze high-dimensional, high-complexity data with positive classification results. The study does numerous iterations to arrive at the best possible network architecture and then uses this architecture as an intrusion detection model to categorize attacks. This increases the classification accuracy and generalization of the model and reduces the complexity of the network structure by addressing the issue of selecting an appropriate network structure when utilizing classification methods for intrusion detection in the face of various threats. Among the many benefits of this approach are: In contrast to previous works such as J. Ren, 2019 and Mugabo, 2020, the classification accuracy of the network structure developed for a given type of assault might exceed 99%. Furthermore, our system has far higher classification accuracy than alternative approaches such as Saif S. Kareem, 2022. Training time can be shortened as model complexity decreases without sacrificing classification precision. Combining I-GA with the LLE model creates an algorithm that can be utilized for intrusion detection in the IoT and other purposes, such as classification and recognition. Appropriate network topology for classification is adaptively built for various training sets. Furthermore, excellent classification accuracy can be achieved even for tiny training sets, which aids in the identification of low-frequency attacks in intrusion detection systems. Several trials were carried out in this study, and the results are shown in the table. IGA-LLE-KNN outperformed the other developed

models with 98.83% accuracy and took the shortest execution time at 4 seconds. Table 4.4 shows the comparison of the results obtained from different works.

Table 4.5: Comparison with existing Results (Researcher Olufemi T. O, 2023)

Author	Technique	Results
Dharani K.et al, 2023	SAR-BSO	95.4%
EESA, 2021	Cuttlefish+KNN	95%
Mugabo 2020	SVM-IG	87%
Mugabo and Zhang 2020	Information Gain-SVM	87%
J. Ren 2019	GA-Random Forest	90%
Wei 2019	PSO-Deep Belief Network	84%

4.9 Table 4.6 Summary of the Results (Researcher Olufemi T. O., 2023)

SN	Research Questions	Objectives	Method	Results
1	What is the most efficient algorithm considered for the development of an improved novel approach?	Develop an improved machine model to fetch relevant information and reduce the curse-of-dimensionality	I-GA - LLE	IGA-LLE=KNN
2	How were the threats classified and what models were used?	Classifications of the threats with the developed model.	SVM AND KNN	Confusion Matrices
3	What Evaluation metrics were used on the developed model and which metric was the n?	Validate the developed model in relationships of performance Evaluation with the metrics.	Accuracy, Precision,	98.83 98.86

4	What results are better with regards to the existing works?	Compare the attained outcomes with existing advances.	Compared Accuracy of the developed model with that of existing works.	The accuracy of the existing work is 95.4 while the newly developed model gave 98.83
---	---	---	---	--

Do Not Copy, Lead City University, Nigeria

Chapter Five

Conclusion

5.1 Summary of Results

IoT systems are susceptible to being hacked or having their networks invaded by malicious actors. This research work presents a method for increasing the reliability of detecting unauthorized access to Internet of Things (IoT) systems. The suggested solution offers an integrated optimization method based on I-GA with LLE to improve the accuracy of network intrusion detection classifiers such as SVM and KNN. The I-GA-LLE extracts the most essential features of datasets and gets rid of noise, iteration, and duplication at the same time. Each feature vector is treated as a member of the I-GA-LLE population, and the values determined to be the most optimal are the ones utilized. At the same time, classifiers are being trained to identify network intrusion.

An intrusion detection system is an essential component of any wireless sensor network's security toolkit because it helps guard against sneaky and unanticipated attacks, hence protecting the network's services and infrastructure. There are just a few approaches for relatively effective machine learning to identify intrusions in wireless sensor networks, and those proposals are few and far between. The problem of uneven data distribution in network traffic still requires a higher level of precision and efficiency to be resolved. In this research, we built a unique model to detect potential intrusion risks by combining an improved Genetic Algorithm with LLE using SVM and KNN models for classification purposes. This model was developed to assist traditional models in improving their ability to recognize attacks from underrepresented groups in the exceedingly asymmetrical data flows typical of modern networks. Experiments using holdout and 10-fold cross-evaluation procedures were carried out on a dataset containing information about intrusion network-detection systems. According

to the findings of a 10-fold cross-evaluation test, the suggested approach obtained a higher detection rate (98.83%) than the state-of-the-art methods and other ensemble learning classifiers. The study's findings reveal important insights into the performance of the Intrusion Detection System (IDS) model utilizing various techniques such as feature selection, dimensionality reduction, and classification algorithms.

The study demonstrates that the combination of feature selection techniques, such as the Improved Genetic Algorithm (I-GA), and dimensionality reduction techniques, such as Local Linear Embedding (LLE), can significantly enhance the performance of the IDS model. These techniques effectively select relevant features and reduce the dataset's dimensionality, improving the intrusion detection accuracy. Furthermore, the study Validates different classifiers, including Support Vector Machine (SVM) and K-nearest neighbors (KNN). The results indicate that both classifiers show promising performance in detecting intrusions. The combination of I-GA, LLE, and KNN yields the highest accuracy, indicating its potential for real-world implementation.

5.1.1 Conclusion

As suggested in various existing research works, multiple existing algorithms can be combined, or any of them can be improved since they all have strengths in one area. It has been proved once again in this research that such suggestions should be embraced continuously. The introduction of Levy flight into the Genetic Algorithm shows that improving any of the Algorithm is a great idea and significantly led to a very reduced dataset of very relevant extracted features, while the second optimization using the LLE further optimized the initially extracted features, which gave more relevant features closely related to intrusions. These more relevant features had a robust impact on the classification phase of the model. The results clearly show that the developed model and processes worked better

compared with the existing works. The intrusion detection-related datasets (CICIDS2017) also helped the model perform closely to the desired expectations. The findings indicate the ideal combination of attributes.

5.2 Contribution to Knowledge

The contributions to knowledge of the research work are enumerated as follows:

- i. Genetic Algorithm has problems falling prematurely into local Optimum and continuous iterations, among others. In this research, the Introduction of Levy Flight, which works as a silent member but successfully selects the optimal solution by avoiding it falling into the local optimum, is excellent. It is a contribution to knowledge. GA is known to be a valuable Algorithm capable of getting Optimal Solutions; the combination of Levy Flight is an excellent contribution, as earlier said.
- ii. The problem of continuous iterations is also arrested as a result of Levy Flight taking a flight with what it sees as the best solution when the repeated iterations are compared, and there is an almost insignificant difference in the latest recorded results. It takes a flight with the latest result and presents it as optimal. This puts a stop to the continuous iterations. It thereby reduces computational time and the complexities of the processes.
- iii. The hybrid optimization model of IGA-LLE is, on its own, another Contribution to Knowledge. The features that resulted from this hybrid model are more relevant to the ones needed for the accurate classification phase. This shows in the accuracy and other metrics compared with existing works. It is worth knowing that some hybrid works have been done, but to the best of my knowledge, in one way or another, they have not combined and improved like the ones used in this research work.

- iv. When the results of the classifications are compared with the existing works, the performance of the newly developed hybrid optimized model with regards to Accuracy is better.

5.3 Suggestions for Further Studies

Since the suggested data optimization may successfully lessen the impact of the imbalanced sample distribution on IDS and has demonstrated encouraging performance, it might be further extended to other domains of anomaly detection, such as fraud detection, also using deep learning techniques. More improvements and combinations can be made on other existing models to continue the upward trends of inaccurate classifications of threats. The search approach could also be improved because training classifiers takes a lot of time.

Bibliography

Chapter of Book

- Dhiman, A., Gupta, K., & Sharma, D. K. “*An Introduction to Deep Learning Applications in Biometric Recognition*”. In **Trends in Deep Learning Methodologies Elsevier**, 2021 (pp. 1–36). <https://doi.org/10.1016/B978-0-12-822226-3.00001-5>
- DiPietro, R., & Hager, G. D. “*Deep Learning: RNNs and LSTM.*” In **Handbook of Medical Image Computing and Computer Assisted Intervention Elsevier**. 2020, pp. 503–519. <https://doi.org/10.1016/B978-0-12-816176-0.00026-0>
- Faircloth, J. “*Hacking Database Services*”. In **Penetration Tester’s Open Source Toolkit Elsevier**. 2017, pp. 215–241. <https://doi.org/10.1016/B978-0-12-802149-1.00006-3>

Conference Proceedings

- Ajay Kumara, M. A., & Jaidhar, C. D. “*Hypervisor and Virtual Machine Dependent Intrusion Detection and Prevention System for Virtualized Cloud Environment*”. **International Conference on Telematics and Future Generation Networks, TAFGEN** , 2015, 28–33. <https://doi.org/10.1109/TAFGEN.2015.7289570>
- Barhate, K., & Jaidhar, C. “*Automated Digital Forensic Technique with Intrusion Detection Systems*”. **3rd IEEE International Advance Computing Conference (IACC)**, 2013, 185–189. <https://doi.org/10.1109/IAdCC.2013.6514218>
- Cai, M., Huang, H., & Huang, J. “*Understanding Security Vulnerabilities in File Systems*”. **Proceedings of the 10th ACM SIGOPS Asia-Pacific Workshop on Systems, APSys**, 2019, 8–15. <https://doi.org/10.1145/3343737.3343753>
- Chabathula, K. J., Jaidhar, C. D., & Ajay Kumara, M. A. “*Comparative Study of Principal Component Analysis Based Intrusion Detection Approach using Machine Learning Algorithms*”. **3rd International Conference on Signal Processing, Communication and Networking (ICSCN)**, 2015, 1–6. <https://doi.org/10.1109/ICSCN.7219853>
- Chen Junli, & Jiao Licheng. “*Classification Mechanism of Support Vector Machines*”. **5th International Conference on Signal Processing Proceedings. 16th World Computer Congress**, 2000, 3, 1556–1559. <https://doi.org/10.1109/ICOSP.2000.893396>
- Di Lucca, G. A., Fasolino, A. R., Mastoianni, M., & Tramontana, P. “*Identifying Cross Site Scripting Vulnerabilities in Web Applications*”. **Web Site Evolution, Sixth IEEE International Workshop on (WSE’04)**, 2004, 71–80. <https://doi.org/10.1109/WSE.2004.10013>

Journal

- Abbas, A., Khan, M. A., Latif, S., Ajaz, M., Shah, A. A., & Ahmad, J.” *A New Ensemble-Based Intrusion Detection System for Internet of Things*”. **Arabian Journal for** 183

Science and Engineering". 2021, <https://doi.org/10.1007/s13369-021-06086-5>

- Abbas, A., Khan, M. A., Latif, S., Ajaz, M., Shah, A. A., & Ahmad, J. "A New Ensemble-Based Intrusion Detection System for Internet of Things". **Arabian Journal for Science and Engineering**, 47(2), 2022, 1805–1819. <https://doi.org/10.1007/s13369-021-06086-5>
- Abdallah, E. E., Eleisah, W., & Otoom, A. F. "Intrusion Detection Systems using Supervised Machine Learning Techniques: A Survey". **Procedia Computer Science**, 2022, 205–212. <https://doi.org/10.1016/j.procs.2022.03.029>
- Abdel-Basset, M., Abdullallah G, Karam M. S, Ibrahim E, Kumudu M, and Abbas J. "An Optimization Model for Appraising Intrusion-Detection Systems for Network Security Communications: Applications, Challenges, and Solutions." **Sensors** 22, no. 11, 2022, 4123. <http://dx.doi.org/10.3390/s22114123>.
- Abdul Rahim, N., Paulraj, M. P., & Adom, A. H. "Adaptive Boosting with SVM Classifier for Moving Vehicle Classification". **Procedia Engineering**, 53, 2013, 411–419. <https://doi.org/10.1016/j.proeng.2013.02.054>
- Abdulhammed, R., Musafer, H., Alessa, A., Faezipour, M., & Abuzneid, A. "Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection". **Electronics**, 8(3), 2019, 322. <https://doi.org/10.3390/electronics8030322>
- Abiodun, A. "Identifying Phishing As A form of Cybercrime in Nigeria". **Nnamdi Azikiwe University Journal of International Law and Jurisprudence**, 12(2), 2021, 176–186.
- Abulaish, M., Halder, N. A. H., & Jahiruddin, J. "P2DF: A Privacy-Preserving Digital Forensics Framework". **International Journal of Digital Crime and Forensics**, 13(6), 2021, 1–15. <https://doi.org/10.4018/IJDCF.288547>
- Adeniyi, D. A., Wei, Z., & Yongquan, Y. "Automated Web Usage Data Mining and Recommendation System using K-Nearest Neighbor (KNN) Classification Method". **Applied Computing and Informatics**, 12(1), 2016, 90–108. <https://doi.org/10.1016/j.aci.2014.10.001>
- Agarwal, N., & Hussain, S. Z. "A Closer Look at Intrusion Detection System for Web Applications" **Security and Communication Networks**, 2018, 1–27. <https://doi.org/10.1155/2018/9601357>
- Ahmad, I., Ul Haq, Q. E., Imran, M., Alassafi, M. O., & AlGhamdi, R. A. "An Efficient Network Intrusion Detection and Classification System". **Mathematics**, 2022, 530. <https://doi.org/10.3390/math10030530>
- Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. "Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches" **Transactions on Emerging Telecommunications Technologies**, 10(3), 2021. <https://doi.org/10.1002/ett.4150>
- Ahsan, M., Nygard, K. E., Gomes, R., Chowdhury, M. M., Rifat, N., & Connolly, J. F. "Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning—A Review" **Journal of Cybersecurity and Privacy**, 2(3), 2022, 527–555. <https://doi.org/10.3390/jcp2030027>

- Nitu D, Sujata C, & Amiya K. R. “Intrusion Detection System Based on Principal Component Analysis and Machine Learning Techniques” **Ijedr**, 6(3), 2018, 359–367.
- Albulayhi, K., Smadi, A. A., Sheldon, F. T, & Abercrombie, R. K. “IoT Intrusion Detection Taxonomy, Reference Architecture, and Analyses” **Sensors**, 21(19), 2021, 6432. <https://doi.org/10.3390/s21196432>
- Aldwairi, M., Abu-Dalo, A. M., & Jarrah, M. “Pattern Matching of Signature-Based IDS using Myers Algorithm under MapReduce Framework”. **EURASIP Journal on Information Security**, (1), 2017, 9. <https://doi.org/10.1186/s13635-017-0062-7>
- Alex, M. E., & Kishore, R. “Forensics Framework for Cloud Computing” **Computers and Electrical Engineering**, 60, 2017, 193–205. <https://doi.org/10.1016/j.compeleceng.2017.02.006>
- Alkahtani, H., & Aldhyani, T. H. H. “Intrusion Detection System to Advance Internet of Things Infrastructure-Based Deep Learning Algorithms” **Complexity**, 2021, 1–18. <https://doi.org/10.1155/2021/5579851>
- Alkhalil, Z., Hewage, C., Nawaf, L., & Khan, I. “Phishing Attacks: A Recent Comprehensive Study and a New Anatomy” **Frontiers in Computer Science**, 2021, 3. <https://doi.org/10.3389/fcomp.2021.563060>
- Almutairi, S., Mahfoudh, S., Almutairi, S., & Alowibdi, J. S. “Hybrid Botnet Detection Based on Host and Network Analysis” **Journal of Computer Networks and Communications**, 2020, 1–16. <https://doi.org/10.1155/2020/9024726>
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. “Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions) **Journal of Big Data**, 8(1), 2021, 53. <https://doi.org/10.1186/s40537-021-00444-8>
- Alzubi, J., Nayyar, A., & Kumar, A. “Machine Learning from Theory to Algorithms: An Overview”. **Journal of Physics: Conference Series**, 1142(1). 2018, <https://doi.org/10.1088/1742-6596/1142/1/012012>
- Andress, J., & Winterfeld, S. “Non-State Actors in Computer Network Operations”. In **Cyber Warfar Elsevier**. <https://doi.org/10.1016/B978-0-12-416672-1.00012-X>, 2014, pp. 207–219.
- Arshad, D., Asim, M., Tariq, N., Baker, T., Tawfik, H., & Al-Jumeily OBE, D. “THC-RPL: A Lightweight Trust-Enabled Routing in RPL-Based IoT Networks against Sybil Attack” **PLOS ONE**, 17(7), 2022, e0271277. <https://doi.org/10.1371/journal.pone.0271277>
- Asperti, A., & Trentin, M. “Balancing Reconstruction Error and Kullback-Leibler Divergence in Variational Autoencoders”. **IEEE Access**, 8, 2020, 199440–199448. <https://doi.org/10.1109/ACCESS.2020.3034828>
- Ávila, R., Khoury, R., Khoury, R., & Petrillo, F. “Use of Security Logs for Data Leak Detection: A Systematic Literature Review”. **Security and Communication Networks**, 2021, 1–29. <https://doi.org/10.1155/2021/6615899>
- Badenhop, C. W., Graham, S. R., Ramsey, B. W., Mullins, B. E., & Mailloux, L. O. “The Z-

- Wave Routing Protocol and its Security Implications*” **Computers & Security**, 68, 2017, 112–129. <https://doi.org/10.1016/j.cose.2017.04.004>
- Banerjee, M., Lee, J., & Choo, K.-K. R. “*A Blockchain Future for Internet of Things Security: A Position Paper*” **Digital Communications and Networks**, 4(3), 2018, 149–160. <https://doi.org/10.1016/j.dcan.2017.10.006>
- Bansal, P., Lamba, R., Jain, V., Jain, T., Shokeen, S., Kumar, S., Singh, P. K., & Khan, B. *GGA-MLP: A Greedy Genetic Algorithm to Optimize Weights and Biases in Multilayer Perceptron*” **Contrast Media & Molecular Imaging**, 2022, 1–14. <https://doi.org/10.1155/2022/4036035>
- Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., & Jenssen, R. “*Properties and Training in Recurrent Neural Networks*” **SpringerBriefs in Computer Science book series (BRIEFSCOMPUTER)**, 2017.
- Bijone, M. “*A Survey on Secure Network: Intrusion Detection & Prevention Approaches*”. **American Journal of Information Systems**, 4(3), 2016, 69–88. <https://doi.org/10.12691/ajis-4-3-2>
- Borsboom, D., Deserno, M. K., Rhemtulla, M., Epskamp, S., Fried, E. I., McNally, R. J., Robinaugh, D. J., Perugini, M., Dalege, J., Costantini, G., Isvoranu, A. M., Wysocki, A. C., van Borkulo, C. D., van Bork, R., & Waldorp, L. J. “*Network Analysis of Multivariate Data in Psychological Science*”. **Nature Reviews Methods Primers**, 1(1). 2021, <https://doi.org/10.1038/s43586-021-00055-w>
- Brereton, R. G., & Lloyd, G. R. “*Support Vector Machines for Classification and Regression*”. **The Analyst**, 135(2), 2010, 230–267. <https://doi.org/10.1039/B918972F>
- Casino, F., Dasaklis, T. K., Spathoulas, G. P., Anagnostopoulos, M., Ghosal, A., Borocz, I., Solanas, A., Conti, M., & Patsakis, C. “*Research Trends, Challenges, and Emerging Topics in Digital Forensics: A Review of Reviews*”. **IEEE Access** 10, 2022, 25464–25493. <https://doi.org/10.1109/ACCESS.2022.3154059>
- Castillo, A. C. “*Network Forensics: Concepts and Challenges*”. **Journal of Forensics Sciences and Criminal Investigation**, 13(1), 2019, 11–15.
- Cateni, S., Vannucci, M., Vannocci, M., & Colla, V. “*Variable Selection and Feature Extraction Through Artificial Intelligence Techniques*”. **Multivariate Analysis in Management, Engineering and the Science**, 6, 2012, 103–118.
- Cazorla, L., Alcaraz, C., & Lopez, J. “*Cyber Stealth Attacks in Critical Information Infrastructures*”. **IEEE Systems Journal**, 12(2), 2018, 1778–1792. <https://doi.org/10.1109/JSYST.2015.2487684>
- Chandol, M. K., & Rao, M. K. “*Border Collie Cat Optimization for Intrusion Detection System in Healthcare IoT Network Using Deep Recurrent Neural Network*”. **The Computer Journal**. <https://doi.org/10.1093/comjnl/bxab136>, 2021.
- Chen, R.-C., Dewi, C., Huang, S.-W., & Caraka, R. E. “*Selecting Critical Features for Data Classification Based on Machine Learning Methods*” **Journal of Big Data**, 7(1), 2020, 52. <https://doi.org/10.1186/s40537-020-00327-4>

- Cheng, H., Tan, P. N., & Jin, R. "Efficient Algorithm for Localized Support Vector Machine". **IEEE Transactions on Knowledge and Data Engineering**, 22(4), 2010, 537–549. <https://doi.org/10.1109/TKDE.2009.116>
- Cheng, S., Lu, H., Lei, X., & Shi, Y. "A Quarter Century of Particle Swarm Optimization". **Complex & Intelligent Systems**, 4(3), 2018, 227–239.
- Cheng, S., Shi, Y., & Qin, Q. "Population Diversity of Particle Swarm Optimizer Solving Single and Multi-Objective Problems". **International Journal of Swarm Intelligence Research (IJSIR)**, 3(4), 2012, 23–60.
- Corizzo, R., Zdravevski, E., Russell, M., Vagliano, A., & Japkowicz, N. "Feature Extraction Based on Word Embedding Models for Intrusion Detection in Network Traffic" **Journal of Surveillance, Security and Safety**, 2020
- Creech, G., & Hu, J. "A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguous and Discontiguous System Call Patterns". **IEEE Transactions on Computers**, 63(4), 2014, 807–819.
- Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. M., Adetunmbi, A. O., & Ajibuwa, O. E. "Machine Learning for E-mail Spam Filtering: Review, Approaches and Open Research Problems" **Heliyon**, 5(6), 2019, e01802.
- Dai, H.-P., Chen, D.-D., & Zheng, Z.-S. "Effects of Random Values for Particle Swarm Optimization Algorithm" **Algorithms**, 11(2), 2018, 23. <https://doi.org/10.3390/a11020023>
- Das, S., & Nayak, T. "Impact of Cyber Crime: Issues and Challenges". **International Journal of Engineering Sciences & Emerging Technologies**, 6(2), 2013, 142–153.
- Dash, S. K., Rawat, S., & Pujari, A. K. "Use of Dimensionality Reduction for Intrusion Detection. In *Information Systems Security*": **Third International Conference, ICISS 2007, Delhi, India, Proceedings, Springer Berlin Heidelberg**. 2007, pp. 306–320.
- Díaz-Verdejo, J., Muñoz-Calle, J., Estepa Alonso, A., Estepa Alonso, R., & Madinabeitia, G. "On the Detection Capabilities of Signature-Based Intrusion Detection Systems in the Context of Web Attacks". **Applied Sciences**, 12(2), 2022, 852. <https://doi.org/10.3390/app12020852>
- Dina, A. S., & Manivannan, D. "Intrusion Detection Based on Machine Learning Techniques in Computer Networks". **Internet of Things**, 16, 2021, 100462. <https://doi.org/10.1016/j.iot.2021.100462>
- Eesa, A. S., Sadiq, S., Hassan, M., & Orman, Z. "Rule Generation Based on Modified Cuttlefish Algorithm for intrusion detection system". **Uludağ University Journal of The Faculty of Engineering**, 2021, 253–268. <https://doi.org/10.17482/uumfd.747078>
- El Miedany, Y. "Artificial Intelligence. In *Rheumatology Teaching*" **Springer International Publishing**. 2019, pp. 347–378. https://doi.org/10.1007/978-3-319-98213-7_18
- Fair, T., Nordfelt, M., Ring, S., & Cole, E. "Advanced Techniques. In *Cyber Spying*" **Elsevier**. 2005, pp. 343–376. <https://doi.org/10.1016/B978-193183641-1>

- Faisal, H., Syed, G. A., Muhammad, H., Ubaid, U. F., Farrukh, S., & Ghalib, A. S. "IOT DOS and DDOS attack dataset". **IEEE Dataport**. 2021. <https://doi.org/10.21227/0s0p-s959>
- Fang, L., Wu, H., Qian, K., Wang, W., & Han, L. "A Comprehensive Analysis of DDoS Attacks Based on DNS". **Journal of Physics: Conference Series**, 2024(1). 2021, <https://doi.org/10.1088/1742-6596/2024/1/012027>
- Fatani, A., Abd Elaziz, M., Dahou, A., Al-Qaness, M. A. A., & Lu, S. "IoT Intrusion Detection System Using Deep Learning and Enhanced Transient Search Optimization". **IEEE Access**, 9, 2021, 123448–123464. <https://doi.org/10.1109/ACCESS.2021.3109081>
- Figuerola Barraza, J., López Droguett, E., & Martins, M. R. "Towards Interpretable Deep Learning: A Feature Selection Framework for Prognostics and Health Management Using Deep Neural Networks". **Sensors** 21, (17), 2021, 5888. <https://doi.org/10.3390/s21175888>
- Foody, G. M., & Mathur, A. "Toward Intelligent Training of Supervised Image Classifications: Directing Training Data Acquisition for SVM Classification". **Remote Sensing of Environment**, 93(1–2), 2004, 107–117. <https://doi.org/10.1016/j.rse.2004.06.017>
- George, G., & Uppin, C. "A Proactive Approach To Network Forensics Intrusion (Denial of Service Flood Attack) Using Dynamic Features, Selection and Convolution Neural Network". **Open Journal of Physical Science (ISSN: 2734-2123)**, 2(2), 2021, 01–09. <https://doi.org/10.52417/ojps.v2i2.237>
- Gill, N. S. 2021. "Artificial Neural Networks Applications and Algorithms". **XenonStack**, 2021, 1–24. <https://www.xenonstack.com/blog/artificial-neural-network-applications/>
- Gopika, P., Sowmya, V., Gopalakrishnan, E. A., & Soman, K. P. "Transferable Approach for Cardiac Disease Classification using Deep Learning". In **Deep Learning Techniques for Biomedical and Health Informatics**. Elsevier. 2020, pp. 285–303
- Griffiths, E., & Chavez, J. M. "Longitudinal Crime Trends at Places. In *Encyclopedia of Criminology and Criminal Justice*" **Springer New York**. 2014, pp. 2953–2960. https://doi.org/10.1007/978-1-4614-5690-2_211
- Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. "A Survey of Methods for Explaining Black Box Models" **ACM Computing Surveys**, 51(5), 2019, 1–42. <https://doi.org/10.1145/3236009>
- Gür, G., Bahtiyar, Ş., & Alagöz, F. "Security Analysis of Computer Networks. In *Modeling and Simulation of Computer Networks and Systems*" **Elsevier**, 2015, pp. 861–898. <https://doi.org/10.1016/B978-0-12-800887-4.00030-4>
- Haley, C. B., Laney, R., Moffett, J. D., & Nuseibeh, B. "Security Requirements Engineering: A Framework for Representation and Analysis". **IEEE Transactions on Software Engineering**, 34(1), 2008, 133–153. <https://doi.org/10.1109/TSE.2007.70754>

- Han, L. "Password Cracking and Countermeasures in Computer Security: A Survey". **arXiv preprint arXiv:1411.7803**. , 2014.
- Hanan Hindy, Christos Tachtatzis, Robert Atkinson, Ethan Bayne, X. B. "MQTT Internet of Things Intrusion Detection Dataset". **IEEE Dataport**. 2020,
- Harimoorthy, K., & Thangavelu, M. "Multi-Disease Prediction Model using Improved SVM-Radial Bias Technique in Healthcare Monitoring System". **Journal of Ambient Intelligence and Humanized Computing**, 12(3), 2021, 3715–3723. <https://doi.org/10.1007/s12652-019-01652-0>
- Hina, M., Ali, M., Javed, A. R., Ghabban, F., Khan, L. A., & Jalil, Z. "SeFACED: Semantic-Based Forensic Analysis and Classification of E-Mail Data Using Deep Learning" **IEEE Access**, 9, 2021, 98398–98411. <https://doi.org/10.1109/ACCESS.2021.3095730>
- Hira, Z. M., & Gillies, D. F. "A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data". **Advances in Bioinformatics**, 2015, 1–13. <https://doi.org/10.1155/2015/198363>
- Hölldobler, K., Rumpe, B., & Wortmann, A. "Software Language Engineering in the Large: Towards Composing and Deriving Languages". **Computer Languages, Systems & Structures**, 54, 2018, 386–405. <https://doi.org/10.1016/j.cl.2018.08.002>
- Holm, H. "Signature Based Intrusion Detection for Zero-Day Attacks: (Not) A Closed Chapter?" **2014 47th Hawaii International Conference on System Sciences**, 2014, 4895–4904. <https://doi.org/10.1109/HICSS.2014.600>
- Hosseinzadeh, H., & Einalou, Z. "Logistic Regression Projection-Based Feature Representation for Visual Domain Adaptation". **Signal, Image and Video Processing**, 14(6), 2020, 1115–1123. <https://doi.org/10.1007/s11760-020-01649-9>
- Hyde, K. F. "Recognising Deductive Processes in Qualitative Research. Qualitative Market Research" **An International Journal**, 3(2),2000, 82–90. <https://doi.org/10.1108/13522750010322089>
- I. Alghamdi, M. " Alghamdi, M. I. 2021. "Digital Forensics in Cyber Security—Recent Trends, Threats, and Opportunities. Cybersecurity Threats with New Perspectives". 2021
- Iqbal, S., & Abed Alharbi, S. "Advancing Automation in Digital Forensic Investigations Using Machine Learning Forensics". **In Digital Forensic Science. IntechOpen**. 2020, <https://doi.org/10.5772/intechopen.90233>
- Jamadar, R. A. "Network Intrusion Detection System Using Machine Learning". **Indian Journal of Science and Technology**, 11(48),2018, 1–6. <https://doi.org/10.17485/ijst/2018/v11i48/139802>
- Jang-Jaccard, J., & Nepal, S. "A Survey of Emerging Threats in Cybersecurity". **Journal of Computer and System Sciences**, 80(5), 2014, , 973–993.
- Javaid, M., Haleem, A., Pratap Singh, R., Suman, R., & Rab, S. "Significance of Machine

Learning in Healthcare: Features, Pillars and Applications". International Journal of Intelligent Networks, 3,2022, 58–73. <https://doi.org/10.1016/j.ijin.2022.05.002>

Jaw, E., & Wang, X. "A Novel Hybrid-Based Approach of Snort Automatic Rule Generator and Security Event Correlation" (SARG-SEC). *PeerJ Computer Science*, 8,2022, e900. <https://doi.org/10.7717/peerj-cs.900>

Jayalaxmi, P. L. S., Saha, R., Kumar, G., & Kim, T.-H. "Machine and Deep Learning Amalgamation for Feature Extraction in Industrial Internet-of-Things". *Computers & Electrical Engineering*, 2021,

Jombart, T., Devillard, S., & Balloux, F. "Discriminant Analysis of Principal Components: a New Method for the Analysis of Genetically Structured Populations". *BMC Genetics*, 11(1), 2010, 94. <https://doi.org/10.1186/1471-2156-11-94>

Kakareka, A. "Detecting System Intrusions. In *Network and System Security*". Elsevier. 2014, pp. 1–27 <https://doi.org/10.1016/B978-0-12-416689-9.00001-0>

Kaliappan, J., Srinivasan, K., Mian Qaisar, S., Sundararajan, K., Chang, C. Y., & Suganthan, C. "Performance Evaluation of Regression Models for the Prediction of the COVID-19" **Reproduction Rate. Frontiers in Public Health**, 2021, pp. 9. <https://doi.org/10.3389/fpubh.2021.729795>

Khan, A. R., Kashif, M., Jhaveri, R. H., Raut, R., Saba, T., & Bahaj, S. A. "Deep Learning for Intrusion Detection and Security of Internet of Things (IoT): Current Analysis, Challenges, and Possible Solutions". *Security and Communication Networks*, 2022, 1–13. <https://doi.org/10.1155/2022/4016073>

Khraisat, A., & Alazab, A. "A Critical Review of Intrusion Detection Systems in the Internet of Things: Techniques, Deployment Strategy, Evaluation Strategy, Attacks, Public Datasets and Challenges". *Cybersecurity*, 4(1), 2021, 18. <https://doi.org/10.1186/s42400-021-00077-7>

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. "Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges". *Cybersecurity*, 2(1),2019, 20. <https://doi.org/10.1186/s42400-019-0038-7>

Kigwana, I., Kebande, V. R., & Venter, H. . "A Proposed Digital Forensic Investigation Framework for an eGovernment Structure for Uganda". **IST-Africa Week Conference (IST-Africa)**, 2017, 1–8. <https://doi.org/10.23919/ISTAFRICA.2017.8102348>

Kim, J. Y. "FPGA Based Neural Network Accelerators". In *Advances in Computers Elsevier*. 2021, Vol. 122, pp. 135-165.

Kim, J., Kim, J., Kim, H., Shim, M., & Choi, E. "CNN-Based Network Intrusion Detection against Denial-of-Service Attacks". *Electronics*, 9(6), 2020, 916. <https://doi.org/10.3390/electronics9060916>

Knapp, E. D., & Langill, J. T. "Exception, Anomaly, and Threat Detection". In *Industrial Network Security. Elsevier*. 2015, pp. 323–350

- Kononenko, I., & Kukar, M. "Artificial Neural Networks. In *Machine Learning and Data Mining*" Elsevier. 2007, pp. 275–320. <https://doi.org/10.1533/9780857099440.275>
- Kumar, D. A. "Intrusion Detection Systems: A Review". **International Journal of Advanced Research in Computer Science**, 8(8), 2017, 356–370. <https://doi.org/10.26483/ijarcs.v8i8.4703>
- Kumar, S., Sunanda, & Arora, S. "A Statistical Analysis on KDD Cup'99 Dataset for the Network Intrusion Detection System. Applied" **Soft Computing and Communication Networks: Proceedings of CAN**, 2020, 131-157.
- Le, T.-T.-H., Kim, Y., & Kim, H. "Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks". **Applied Sciences**, 9(7),2019, 1392. <https://doi.org/10.3390/app9071392>
- Lee, J.-G., Jun, S., Cho, Y.-W., Lee, H., Kim, G. B., Seo, J. B., & Kim, N. "Deep Learning in Medical Imaging: General Overview". **Korean Journal of Radiology**, 18(4),2017, 570.
- Leevy, J. L., & Khoshgoftaar, T. M. "A Survey and Analysis of Intrusion Detection Models Based on CSE-CIC-IDS2018 Big Data". **Journal of Big Data**, 7(1), 2020, 104. <https://doi.org/10.1186/s40537-020-00382-x>
- Li, Y., Ghoreishi, S., & Issakhov, A. "Improving the Accuracy of Network Intrusion Detection System in Medical IoT Systems through Butterfly Optimization Algorithm" **Wireless Personal Communications**. 2021,
- Liang, C., Shanmugam, B., Azam, S., Karim, A., Islam, A., Zamani, M., Kavianpour, S., & Idris, N. B. "Intrusion Detection System for the Internet of Things Based on Blockchain and Multi-Agent Systems". **Electronics**, 9(7),2020, 1120. <https://doi.org/10.3390/electronics9071120>
- Liao, Y., & Vemuri, V. R. "Use of K-Nearest Neighbor Classifier for Intrusion Detection". **Computers and Security**, 21(5),2002, 439–448. [https://doi.org/10.1016/S0167-4048\(02\)00514-X](https://doi.org/10.1016/S0167-4048(02)00514-X)
- Lin, J., Sadeh, N., Amini, S., Lindqvist, J., Hong, J. I., & Zhang, J. "Expectation and Purpose". **Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp**, 12, 2012, 501. <https://doi.org/10.1145/2370216.2370290>
- Liu, G., Zhao, H., Fan, F., Liu, G., Xu, Q., & Nazir, S. "An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs". **Sensors**, 22(4), 2022, 1407. <https://doi.org/10.3390/s22041407>
- Lyko, K., Nitzschke, M., & Ngonga Ngomo, A.-C. 2016. "Big Data Acquisition. In *New Horizons for a Data-Driven Economy*" Springer International Publishing. 2016, pp. 39–61. https://doi.org/10.1007/978-3-319-21569-3_4
- Ma, X., Zhang, Y., Cao, H., Zhang, S., & Zhou, Y. "Nonlinear Regression with High-Dimensional Space Mapping for Blood Component Spectral Quantitative Analysis". **Journal of Spectroscopy**, 2018, 1–8. <https://doi.org/10.1155/2018/2689750>

- Mahdaveinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., & Sheth, A. P. "Machine Learning for Internet of Things Data Analysis: A Survey". **Digital Communications and Networks**, 4(3), 2018, 161–175. <https://doi.org/10.1016/j.dcan.2017.10.002>
- Mahdi, M. N., Mohamed Zabil, M. H., Ahmad, A. R., Ismail, R., Yusoff, Y., Cheng, L. K., Azmi, M. S. B. M., Natiq, H., & Happala Naidu, H. "Software Project Management Using Machine Learning Technique—A Review". **Applied Sciences**, 11(11), 2021, 5183. <https://doi.org/10.3390/app11115183>
- Makrakis, G. M., Kolias, C., Kambourakis, G., Rieger, C., & Benjamin, J. "Vulnerabilities and Attacks against Industrial Control Systems and Critical Infrastructures". **arXiv preprint arXiv**, 2021,
- Mallik, A., Ahsan, A., Shahadat, M. M. Z., & Tsou, J. C. "Man-In-The-Middle-Attack: Understanding in Simple Words". **International Journal of Data and Network Science**, 3(2), 2019, 77–92.
- Mallikarjunan, K. N., Muthupriya, K., & Shalinie, S. M. "A Survey of Distributed Denial of Service Attack". **2016 10th International Conference on Intelligent Systems and Control (ISCO)**, 2016, 1–6. <https://doi.org/10.1109/ISCO.2016.7727096>
- McHugh, M. L. "The Chi-Square Test of Independence". **Biochemia Medica**, 23(2), 2012, 143–149. <https://doi.org/10.11613/BM.2013.018>
- Mcilroy, R., Sevcik, J., Tebbi, T., Titzer, B. L., & Verwaest, T. "Spectre is here to stay: An Analysis of Side-Channels and Speculative Execution". <http://arxiv.org/abs/1902.05178>, 2019.
- McInnes, L., Healy, J., & Melville, J. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". <http://arxiv.org/abs/1802.03426>, 2018.
- Menardi, G., & Torelli, N. "Evaluating Enterprise Risk of Default using Boosting Procedures." 2008, 1–4.
- Miyamoto, D., Hazeyama, H., & Kadobayashi, Y. "An Evaluation of Machine Learning-Based Methods for Detection of Phishing Sites", 2009, pp. 539–546. https://doi.org/10.1007/978-3-642-02490-0_66
- Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., & Rajarajan, M. "A Survey of Intrusion Detection Techniques in Cloud". **Journal of Network and Computer Applications**, 36(1), 2012, 42–57.
- Mohamed, T. S., & Aydin, S. "IoT-Based Intrusion Detection Systems: A Review". **Smart Science**, 2021, 1–18. <https://doi.org/10.1080/23080477.2021.1972914>
- Mohammadpour, L., Ling, T. C., Liew, C. S., & Aryanfar, A. "A Mean Convolutional Layer for Intrusion Detection System". **Security and Communication Networks**, 2020, 1–16. <https://doi.org/10.1155/2020/8891185>

- Morales-Molina, C. D., Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, L. K., Perez-Meana, H., Olivares-Mercado, J., Portillo-Portillo, J., Sanchez, V., & Garcia-Villalba, L. J. "A Dense Neural Network Approach for Detecting Clone ID Attacks on the RPL Protocol of the IoT". **Sensors**, 21(9), 2021, 3173. <https://doi.org/10.3390/s21093173>
- Moustakidis, S., & Karlsson, P. "A Novel Feature Extraction Methodology using Siamese Convolutional Neural Networks for Intrusion Detection". **Cybersecurity**, 3(1), 2020, 16. <https://doi.org/10.1186/s42400-020-00056-4>
- Mugabo, E., & Zhang, Q. Y. "Intrusion Detection Method Based on Support Vector Machine and Information Gain for Mobile Cloud Computing". **Int. J. Netw. Secur.**, 22(2), 2020, 231-241.
- N. Ahmed, M., Abdullah, A. H., & Kaiwartya, O. "FSM-F: Finite State Machine Based Framework for Denial of Service and Intrusion Detection in MANET". **PLOS ONE**, 11(6), 2016, e0156885. <https://doi.org/10.1371/journal.pone.0156885>
- Nagarajan, A., Varadharajan, V., & Tarr, N. "Trust Enhanced Distributed Authorisation for Web Services". **Journal of Computer and System Sciences**, 80(5), 2014, 916–934. <https://doi.org/10.1016/j.jcss.2014.02.002>
- Nascimento, G., & Correia, M. "Anomaly-Based Intrusion Detection in Software as a Service". **2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)**, 2011, 19–24. <https://doi.org/10.1109/DSNW.2011.5958858>
- Navamani, T. M. "Efficient Deep Learning Approaches for Health Informatics". In **Deep Learning and Parallel Computing Environment for Bioengineering Systems**, Elsevier, 2019, pp. 123–137. <https://doi.org/10.1016/B978-0-12-816718-2.00014-2>
- Ndatinya, V., Xiao, Z., Manepalli, V. R., Meng, K., & Xiao, Y. "Network Forensics Analysis using Wireshark". **International Journal of Security and Networks**, 2015, 10(2), 91. <https://doi.org/10.1504/IJSN.2015.070421>
- Nehinbe, J. O. "A Critical Evaluation of Datasets for Investigating IDSs and IPSs Researches". **IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS)**, 2011, 92–97. <https://doi.org/10.1109/CIS.2011.6169141>
- Nguyen, L. H., & Holmes, S. "Ten Quick Tips for Effective Dimensionality Reduction". **PLOS Computational Biology**, 15(6), 2019, e1006907. <https://doi.org/10.1371/journal.pcbi.1006907>
- Obasuyi, G. C., & Sari, A. "Security Challenges of Virtualization Hypervisors in Virtualized Hardware Environment". **International Journal of Communications, Network and System Sciences**, 08(07), 2015, 260–273. <https://doi.org/10.4236/ijcns.2015.87026>
- Hambali, A. H., Yakub, K. S., Oladele, T. O., & Gbolagade, M. D. "ADABOOST Ensemble Algorithms for Breast Cancer Classification". 2019
- Olatosi, B., Sun, X., Chen, S., Zhang, J., Liang, C., Weissman, S., & Li, X. "Application of

- Machine-Learning Techniques in Classification of HIV Medical Care Status for People Living with HIV in South Carolina". AIDS, 35(Supplement 1), 2021, S19–S28. <https://doi.org/10.1097/QAD.0000000000002814>*
- Ormandy, T. "An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments". 2007.
- Ozkan-Okay, M., Samet, R., Aslan, O., & Gupta, D. "A Comprehensive Systematic Literature Review on Intrusion Detection Systems". **IEEE Access**, 9, 2021, 157727–157760. <https://doi.org/10.1109/ACCESS.2021.3129336>
- Pan, Y., Sun, F., Teng, Z., White, J., Schmidt, D. C., Staples, J., & Krause, L. "Detecting Web Attacks with End-to-End Deep Learning". **Journal of Internet Services and Applications**, 10(1), 2019, 16. <https://doi.org/10.1186/s13174-019-0115-x>
- Patcha, A., & Park, J.-M. "Network Anomaly Detection with Incomplete Audit Data". **Computer Networks**, 51(13), 2007, 3935–3955. <https://doi.org/10.1016/j.comnet.2007.04.017>
- Pinkas, B., & Sander, T. "Securing Passwords against Dictionary Attacks". **Proceedings of the ACM Conference on Computer and Communications Security**, 2002, 161–170. <https://doi.org/10.1145/586110.586133>
- Pinto da Costa, J. F., & Cabral, M. "Statistical Methods with Applications in Data Mining: A Review of the Most Recent Works". **Mathematics**, 10(6), 2022, 993. <https://doi.org/10.3390/math10060993>
- Podgorelec, V., & Zorman, M. "Decision Tree Learning. In Encyclopedia of Complexity and Systems Science" **Springer Berlin Heidelberg**. 2015, pp. 1–28. https://doi.org/10.1007/978-3-642-27737-5_117-2
- Qadeer, M. A., Zahid, M., Iqbal, A., & Siddiqui, M. R. "Network Traffic Analysis and Intrusion Detection using Packet Sniffer". **2nd International Conference on Communication Software and Networks, ICCSN**, 2010, 313–317. <https://doi.org/10.1109/ICCSN.2010.104>
- Qureshi, S., Li, J., Akhtar, F., Tunio, S., Khand, Z. H., & Wajahat, A. "Analysis of Challenges in Modern Network Forensic Framework". **Security and Communication Networks**, 2021, 1–13. <https://doi.org/10.1155/2021/8871230>
- Rakotondravony, N., Taubmann, B., Mandarawi, W., Weishäupl, E., Xu, P., Kolosnjaji, B., Protsenko, M., de Meer, H., & Reiser, H. P. "Classifying Malware Attacks in IaaS Cloud Environments". **Journal of Cloud Computing**, 6(1). 2017, <https://doi.org/10.1186/s13677-017-0098-8>
- Ramos, V., & Abraham, A. "ANTIDS: Self Organized Ant-Based Clustering Model for Intrusion Detection System" **Advances in Soft Computing, AISC**, 2005, 977–986. https://doi.org/10.1007/3-540-32391-0_101
- Ren, J., Guo, J., Qian, W., Yuan, H., Hao, X., & Jingjing, H. "Building an Effective Intrusion Detection System by Using Hybrid Data Optimization Based on Machine Learning Algorithms". **Security and Communication Networks**, 2019, 1–11.

<https://doi.org/10.1155/2019/7130868>

- Ren, Y., Zhang, L., & Suganthan, P. N. "Ensemble Classification and Regression-Recent Developments, Applications and Future Directions [Review Article]". **IEEE Computational Intelligence Magazine**, 11(1), 2016, 41–53. <https://doi.org/10.1109/MCI.2015.2471235>
- Roa, J., Pividori, M., Gutiérrez, M. D. los M., & Stegmayer, G. "How to Develop Intelligent Agents in an Easy Way with FAIA". In **Quality and Communicability for Interactive Hypermedia Systems**, 2010, pp. 121–141. IGI Global. <https://doi.org/10.4018/978-1-61520-763-3.ch006>
- Rosenberg, J. "Embedded Security. In *Rugged Embedded Systems*" Elsevier. 2017, pp. e1–e74. <https://doi.org/10.1016/B978-0-12-802459-1.00011-7>
- Rutkowski, L., Jaworski, M., Pietruczuk, L., & Duda, P. "The CART Decision Tree for Mining Data Streams". **Information Sciences**, 266, 2014, 1–15. <https://doi.org/10.1016/j.ins.2013.12.060>
- Saadatfar, H., Khosravi, S., Joloudari, J. H., Mosavi, A., & Shamshirband, S. "A New K-Nearest Neighbors Classifier for Big Data Based on Efficient Data Pruning". **Mathematics**, 8(2), 2020, 286. <https://doi.org/10.3390/math8020286>
- Sadeghpour, S., & Vlajic, N. "Click Fraud in Digital Advertising: A Comprehensive Survey". **Computers**, 10(12), 2021, <https://doi.org/10.3390/computers10120164>
- Saha, D., & Manickavasagan, A. "Machine Learning Techniques for Analysis of Hyperspectral Images to Determine Quality of Food Products: A Review". **Current Research in Food Science**, 2021, 4, 28–44. <https://doi.org/10.1016/j.crfs.2021.01.002>
- Santosh, K., Das, N., & Ghosh, S. "Deep learning: A Review. In *Deep Learning Models for Medical Imaging*" Elsevier. 2022, pp. 29–63. <https://doi.org/10.1016/B978-0-12-823504-1.00012-X>
- Saputra, M. F. A., Widiyaningtyas, T., & Wibawa, A. P. "Illiteracy Classification Using K Means-Naïve Bayes Algorithm". **JOIV: International Journal on Informatics Visualization**, 2(3), 2018, 153. <https://doi.org/10.30630/joiv.2.3.129>
- Sarhan, M., Layeghy, S., Moustafa, N., Gallagher, M., & Portmann, M. "Feature Extraction for Machine Learning-Based Intrusion Detection in IoT Networks". **Digital Communications and Networks**. 2022.
- Sarker, I. H. "Machine Learning: Algorithms, Real-World Applications and Research Directions". **SN Computer Science**, 2(3), 2021, 160. <https://doi.org/10.1007/s42979-021-00592-x>
- Sarker, I. H. "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions". **SN Computer Science**, 2(6), 2021, 420. <https://doi.org/10.1007/s42979-021-00815-1>
- Sarker, I. H., Kayes, A. S. M., Badsha, S., Alqahtani, H., Watters, P., & Ng, A. "Cybersecurity Data Science: An Overview from Machine Learning Perspective".

- Sathya, M., Jeyaselvi, M., Joshi, S., Pandey, E., Pareek, P. K., Jamal, S. S., Kumar, V., & Atiglah, H. K. "Cancer Categorization Using Genetic Algorithm to Identify Biomarker Genes". **Journal of Healthcare Engineering**, 2022, 1–12. <https://doi.org/10.1155/2022/5821938>
- Schaffer, A. L., Dobbins, T. A., & Pearson, S.-A. "Interrupted Time Series Analysis using Autoregressive Integrated Moving Average (ARIMA) Models: A Guide for Evaluating Large-Scale Health Interventions". **BMC Medical Research Methodology**, 2021, 21(1), 58.
- Schmidt, J., Marques, M. R. G., Botti, S., & Marques, M. A. L. "Recent Advances and Applications of Machine Learning in Solid-State Materials Science". **Npj Computational Materials**, 5(1), 2019, 83. <https://doi.org/10.1038/s41524-019-0221-0>
- Schneider, P., & Xhafa, F. "Anomaly Detection, Classification and CEP with ML Methods. In *Anomaly Detection and Complex Event Processing over IoT Data Streams*" **Elsevier**. 2022, pp. 193–233. <https://doi.org/10.1016/B978-0-12-823818-9.00020-1>
- Selvapandian, D., & Santhosh, R. "Deep Learning Approach for Intrusion Detection in IoT-Multi Cloud Environment". **Automated Software Engineering**, 28(2), 2021, 19. <https://doi.org/10.1007/s10515-021-00298-7>
- Seungjin, L., Abdullah, A., & Jhanjhi, N. Z. "A Review on Honeypot-Based Botnet Detection Models for Smart Factory". **International Journal of Advanced Computer Science and Applications**, 11(6), 2020, 418–435.
- Seyfollahi, A., & Ghaffari, A. "A Review of Intrusion Detection Systems in RPL Routing Protocol Based on Machine Learning for Internet of Things Applications". **Wireless Communications and Mobile Computing**, 2021, 1–32.
- Shah, N., Bhagat, N., & Shah, M. "Crime Forecasting: A Machine Learning and Computer Vision Approach to Crime Prediction and Prevention". **Visual Computing for Industry, Biomedicine, and Art**, 4(1), 2021, 9. <https://doi.org/10.1186/s42492-021-00075-z>
- Sharafaldin, I., Gharib, A., Lashkari, A. H., & Ghorbani, A. A. "Towards a Reliable Intrusion Detection Benchmark Dataset". **Software Networking**, (1), 2018, 177-200.
- Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". **ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy**, 2018 108–116. <https://doi.org/10.5220/0006639801080116>
- Sharif, W., Yanto, I. T. Y., Samsudin, N. A., Deris, M. M., Khan, A., Mushtaq, M. F., & Ashraf, M. "An Optimised Support Vector Machine With Ringed Seal Search Algorithm for Efficient Text Classification". **Journal of Engineering Science and Technology**, 14(3),2019, 1601–1613.
- Shi, H., Schmidt, B., Liu, W., & Müller-Wittig, W. "Parallel Mutual Information Estimation

- for Inferring Gene Regulatory Networks on GPUs". **BMC Research Notes**, 4(1), 2011, 189. <https://doi.org/10.1186/1756-0500-4-189>
- Shi, Y. "Brain Storm Optimization Algorithm in Objective Space". **2015 IEEE Congress on Evolutionary Computation (CEC)**, 2015, 1227–1234. <https://doi.org/10.1109/CEC.2015.7257029>
- Shiravi, A., Shiravi, H., Tavallae, M., & Ghorbani, A. A. "Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection". **Computers & Security**, 31(3), 2012, 357–374.
- Sindian, S., & Samer, S. "An Enhanced Deep Autoencoder-Based Approach for DDoS Attack Detection". **Wseas Trans. Syst. Control**, 15, 2020, 716–725.
- Singh, K. N., Devi, S. D., Devi, H. M., & Mahanta, A. K. "A Novel Approach for Dimension Reduction using Word Embedding: An Enhanced Text Classification Approach". **International Journal of Information Management Data Insights**, 2(1), 2022, 100061.
- Sivamohan, S., Veeramani, R., Liza, K., Krishnaveni, S., & Jothi, B. "Data Mining Technique for DDoS Attack in Cloud Computing". **International Journal of Control Theory and Applications**, 9(28), 2016, 149–156.
- Souza, R., Rittner, L., & Lotufo, R. "A comparison between k-Optimum Path Forest and K-Nearest Neighbors Supervised Classifiers". **Pattern Recognition Letters**, 2014, 39, 2–10.
- Stephenson, F. H. "Forensics and Paternity. In Calculations for Molecular Biology and Biotechnology" **Elsevier**. 2010, pp. 423–446.
- Sun, L., & Gu, W. "Pavement Condition Assessment Using Fuzzy Logic Theory and Analytic Hierarchy Process". **Journal of Transportation Engineering**, 137(9), 2011, 648–655.
- Sun, P., Liu, P., Li, Q., Liu, C., Lu, X., Hao, R., & Chen, J. "DL-IDS: Extracting Features Using CNN-LSTM Hybrid Network for Intrusion Detection System". **Security and Communication Networks**, 2020, 1–11.
- Tan, S., Xie, P., Guerrero, J. M., Vasquez, J. C., Li, Y., & Guo, X. "Attack detection design for dc microgrid using eigenvalue assignment approach". **Energy Reports**, 7, 2021, 469–476.
- Talapula, D. K., Ravulakollu, K. K., Kumar, M., & Kumar, A. "SAR-BSO Meta-Heuristic Hybridization for Feature Selection and Classification using DBNover Stream Data". **Artificial Intelligence Review**, 2023, 1-39.
- Thakare, A., Bhende, M., Deb, N., Degadwala, S., Pant, B., & Kumar, Y. P. "Classification of Bioinformatics EEG Data Signals to Identify Depressed Brain State Using CNN Model". **BioMed Research International**, 2022, 1–9.
- Thirimanne, S. P., Jayawardana, L., Yasakethu, L., Liyanaarachchi, P., & Hewage, C. "Deep Neural Network Based Real-Time Intrusion Detection System". **SN Computer Science**, 3(2), 2022, 145.

- Thomas, C., Sharma, V., & Balakrishnan, N. "Usefulness of DARPA dataset for intrusion detection system Evaluation". **Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security** 2008, 6973, 69730G.
- Tian, J., Dong, D., Liu, Z., & Wei, J. "Key technologies and software platforms for radiomics. In *Radiomics and Its Clinical Application*" **Elsevier**. pp. 2021, 19–98.
- Ullah, I., & Mahmoud, Q. H. "Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks". **IEEE Access**, 9, 2021, 103906–103926.
- Ulven, J. B., & Wangen, G. "A Systematic Review of Cybersecurity Risks in Higher Education". **Future Internet**, 13(2), 2021, 1–40. <https://doi.org/10.3390/fi13020039>
- Vens, C. "Random Forest. In *Encyclopedia of Systems Biology*", **Springer New York**. 2013, pp. 1812–1813.
- Verma, P., Dumka, A., Singh, R., Ashok, A., Gehlot, A., Malik, P. K., Gaba, G. S., & Hedabou, M. "A Novel Intrusion Detection Approach Using Machine Learning Ensemble for IoT Environments". **Applied Sciences**, 11(21), 2021, 10268.
- Wall, D. S. 2012. "Enemies within: Redefining the insider threat in organizational security policy". **Security Journal**, 26(2), 2012, 107–124. <https://doi.org/10.1057/sj>.
- Wang, S., Di, J., Wang, D., Dai, X., Hua, Y., Gao, X., Zheng, A., & Gao, J. *State-of-the-Art Review of Artificial Neural Networks to Predict, Characterize and Optimize Pharmaceutical Formulation*. **Pharmaceutics**, 14(1), 2022, 183. <https://doi.org/10.3390/pharmaceutics14010183>
- Watkins, M. W. "Exploratory Factor Analysis: A Guide to Best Practice". **Journal of Black Psychology**, 44(3), 2018, 219–246. <https://doi.org/10.1177/0095798418771807>
- Wei, P., Li, Y., Zhang, Z., Hu, T., Li, Z., & Liu, D. "An Optimization Method for Intrusion Detection Classification Model Based on Deep Belief Network". **IEEE Access**, 7, 2019, 87593–87605. <https://doi.org/10.1109/ACCESS.2019.2925828>
- Wellmann, J. "Information Theory for Correlation Analysis and Estimation of Uncertainty Reduction in Maps and Models." **Entropy**, 15(12), 2013, 1464–1485. <https://doi.org/10.3390/e15041464>
- Wiles, J., Gudaitis, T., Jabbusch, J., Rogers, R., & Lowther, S. "Low Tech Hacking: Street Smarts for Security Professionals". **Elsevier**. 2011.
- Xiao, Y., Xing, C., Zhang, T., & Zhao, Z. "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks". **IEEE Access**, 7, 2019, 42210–42219. <https://doi.org/10.1109/ACCESS.2019.2904620>
- Xu, L., Song, B., & Cao, M. "An Improved Particle Swarm Optimization Algorithm with Adaptive Weighted Delay Velocity". **Systems Science & Control Engineering**, 9(1), 2021, 188–197. <https://doi.org/10.1080/21642583.2021.1891153>
- Xuan, J., Jiang, H., & Ren, Z. "Pseudocode of Genetic Algorithm and Multi-Start Strategy

Based Simulated Annealing Algorithm for Large Scale Next Release Problem". Dalian University of Technology, 2011

- Yaacoub, J.-P. A., Noura, H. N., Salman, O., & Chehab, A. "A Survey on Ethical Hacking: Issues and Challenges". **arXiv preprint** arXiv:2103.15072 , 2021
- Yaacoub, J.-P., Noura, H., Salman, O., & Chehab, A. "Security Analysis of Drones Systems: Attacks, Limitations, and Recommendations". **Internet of Things**, 11, 2020, 100218. <https://doi.org/10.1016/j.iot.2020.100218>
- Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. "Convolutional Neural Networks: An Overview and Application in Radiology". **Insights into Imaging**, 9(4), 2018, 611–629. <https://doi.org/10.1007/s13244-018-0639-9>
- Yang, D., Yu, Z., Yuan, H., & Cui, Y. "An Improved Genetic Algorithm and its Application in Neural Network Adversarial Attack". **PLOS ONE**, 17(5), 2022, e0267970. <https://doi.org/10.1371/journal.pone.0267970>
- Youssef, A., & Emam, A. "Network Intrusion Detection Using Data Mining and Network Behaviour Analysis". **International Journal of Computer Science and Information Technology**, 3(6), 2011, 87–98. <https://doi.org/10.5121/ijcsit.2011.3607>
- Zalaghi, H., & Khazaei, M. "The Role of Deductive and Inductive Reasoning in Accounting Research and Standard Setting". **Asian Journal of Finance & Accounting**, 8(1), 2016, 23. <https://doi.org/10.5296/ajfa.v8i1.8148>
- Zhang, B., Liu, Z., Jia, Y., Ren, J., & Zhao, X. "Network Intrusion Detection Method Based on PCA and Bayes Algorithm". **Security and Communication Networks**, 2018, 1–11. <https://doi.org/10.1155/2018/1914980>
- Zhang, C., Zhong, P., Liu, M., Song, Q., Liang, Z., & Wang, X. "Hybrid Metric K-Nearest Neighbor Algorithm and Applications". **Mathematical Problems in Engineering**, 2022, 1–15. <https://doi.org/10.1155/2022/8212546>
- Zhang, Ying, Li, P., & Wang, X. "Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network". **IEEE Access**, 7, 2019, 31711–31722. <https://doi.org/10.1109/ACCESS.2019.2903723>
- Zhang, Yudong, Wang, S., & Ji, G. "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications". **Mathematical Problems in Engineering**, 2015, 1–38.
- Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., & Garant, D. "Botnet detection based on traffic behavior analysis and flow intervals". **Computers & Security**, 39, 2013, 2–16.
- Zlatanov, N. "Computer Security and Mobile Security Challenges". In **Conference: Tech Security Conference At**. 2015
- Zuech, R., Khoshgoftaar, T. M., Seliya, N., Najafabadi, M. M., & Kemp, C. "A new intrusion detection benchmarking system". **Proceedings of the 28th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2015**, 252–255.

Do Not Copy, Lead City University, Nigeria

Appendix

11/15/22, 4:31 PM data_lle_classifiers2.ipynb - Colaboratory

```
#!/ pip install tqdm update_checker tqdm

#!/ pip install tpot

import matplotlib

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import seaborn as sns

import sklearn

import imblearn

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.feature_selection import RFE

import itertools

from sklearn.model_selection import train_test_split

#from tpot import TPOTRegressor

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
roc_auc_score, roc_curve

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.manifold import LocallyLinearEmbedding, MDS

from sklearn.svm import SVC

#Load Data

train = pd.read_csv(r'/content/Train_data.csv')
```

```

test = pd.read_csv(r'/content/Test_data.csv')

print("Training data has {} rows & {} columns".format(train.shape[0],train.shape[1]))

print(test.head(4))

print("Testing data has {} rows & {} columns".format(test.shape[0],test.shape[1]))

train.describe()

#print(train['num_outbound_cmds'].value_counts())

#print(test['num_outbound_cmds'].value_counts())

# 'num_outbound_cmds' is a redundant column so remove it from both train & test datasets
train.drop(['num_outbound_cmds'], axis=1, inplace=True) test.drop(['num_outbound_cmds'],
axis=1, inplace=True)

train['class'].value_counts()

Training data has 25192 rows & 42 columns

```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	\0
0	0	tcp	private	REJ	0	0	0	0
1	0	tcp	private	REJ	0	0	0	0
2	2	tcp	ftp_data	SF	12983	0	0	0
3	0	icmp	eco_i	SF	20	0	0	0

	wrong_fragment	urgent	hot	...	dst_host_count	dst_host_srv_count	\0
0	0	0	0	...	255	10	0
1	0	0	0	...	255	1	0
2	0	0	0	...	134	86	0
3	0	0	0	...	3	57	0

	dst_host_same_srv_rate	dst_host_diff_srv_rate	\0
0	0.04	0.06	0
1	0.00	0.06	0

2 0.61 0.04 0

3 1.00 0.00 0

dst_host_same_src_port_rate dst_host_srv_diff_host_rate \0

0 0.00 0.00 0

1 0.00 0.00 0

2 0.61 0.02 0

3 1.00 0.28 0

dst_host_serror_rate dst_host_srv_serror_rate dst_host_rerror_rate 0

0 0.0 0.0 1.0 0

1 0.0 0.0 1.0 0

2 0.0 0.0 0.0 0

3 0.0 0.0 0.0 0

dst_host_srv_rerror_rate 0

0 1.0 0

1 1.0 0

2 0.0 0

3 0.0 0

[4 rows x 41 columns]0

Testingdata has 22544 rows & 41 columns0

normal 134490

anomaly 117430

Name: class, dtype: int64

scaler = StandardScaler()0

extract numerical attributes and scale it to have zero mean and unit variance 0 cols =

train.select_dtypes(include=['float64','int64']).columns0

```

sc_train = scaler.fit_transform(train.select_dtypes(include=['float64','int64']))
sc_test = scaler.fit_transform(test.select_dtypes(include=['float64','int64']))

# turn the result back to a dataframe
sc_traindf = pd.DataFrame(sc_train, columns = cols)
sc_testdf = pd.DataFrame(sc_test, columns = cols)

encoder = LabelEncoder()

# extract categorical attributes from both training and test sets
cattrain = train.select_dtypes(include=['object']).copy()
cattest = test.select_dtypes(include=['object']).copy()

# encode the categorical attributes
traincat = cattrain.apply(encoder.fit_transform)
testcat = cattest.apply(encoder.fit_transform)

# separate target column from encoded data
enctrain = traincat.drop(['class'], axis=1)
cat_Ytrain = traincat[['class']].copy()

#Union of processed numerical and categorical data
train_x = pd.concat([sc_traindf,enctrain],axis=1)
train_y = cat_Ytrain

train_x.shape
(22544, 40)

test_df = pd.concat([sc_testdf,testcat],axis=1)
test_df.shape
(22544, 40)

rfc = RandomForestClassifier()

# fit random forest classifier on the training set
rfc.fit(train_x, train_y)

# extract important features
score = np.round(rfc.feature_importances_,3)

importances = pd.DataFrame({'feature':train_x.columns,'importance':score})

```



```

importances = importances.sort_values('importance',ascending=False).set_index('feature')
# plot importances
plt.rcParams['figure.figsize'] = (11, 4)
importances.plot.bar();

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DataConversionWarn after
removing the cwd from sys.path.

# create the RFE model and select 10 attributes
rfe = RFE(rfc,
n_features_to_select=10)

rfe = rfe.fit(train_x, train_y)

# summarize the selection of the attributes

feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), train_x.columns)]
selected_features = [v for i, v in feature_map if i==True]

selected_features

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)

```

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector

estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector

estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector

estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector

estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector

estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector

estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector

estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector

estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0

```

estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector
a = [i[0] for i in feature_map]0
train_x = train_x.iloc[:,a]0
test_df = test_df.iloc[:,a]0
train_x.shape0
(25192, 10)
#implementation of the locally linear embedding algorithm0
embedding = LocallyLinearEmbedding(n_components=7, eigen_solver = 'dense')0
Train_x_lle = embedding.fit_transform(train_x)0

```

```

X_train,X_test,Y_train,Y_test      =      train_test_split(Train_x_lle,train_y,train_size=0.70,
random_state=2)0 input_shape = [X_train.shape[1]]0

#SVM MODEL0

model_svm = SVC(random_state=1)0

model_svm.fit(X_train, Y_train)0

y_pred_svm = model_svm.predict(X_test)0

print(classification_report(Y_test, y_pred_svm))0

/usr/local/lib/python3.7/dist-packages/sklearn/utils/evaluation.py:993:
DataConversionWarning: A column-vector y was passed as a column or 1d array, which may be
misinterpreted as a scalar when fitting. Please use the np.ndarray or array interface.
DataConversionWarning: A column-vector y was passed as a column or 1d array, which may be
misinterpreted as a scalar when fitting. Please use the np.ndarray or array interface.

              precision    recall  f1-score   support

0               0.84      0.85      0.84      34980
1               0.87      0.86      0.86      40600

 accuracy               0.85      0.85      0.85      75580
 macro avg              0.85      0.85      0.85      75580
weighted avg              0.86      0.85      0.85      75580

#=====CONSTRUCTING THE CONFUSION
MATRIX=====#0

def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',
cmap=plt.cm.Reds):0 0

plt.figure(figsize=(10,10))0

plt.imshow(cm, interpolation='nearest', cmap=cmap)0

plt.title(title)0

plt.colorbar()0

tick_marks = np.arange(len(classes))0

plt.xticks(tick_marks, classes, rotation=45)0

plt.yticks(tick_marks, classes)0

```

```

if normalize:0

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]0

cm = np.around(cm, decimals=2)0

cm[np.isnan(cm)] = 0.00

thresh = cm.max() / 2.0

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):0 plt.text(j, i, cm[i, j],0
horizontalalignment="center",0
color="white" if cm[i, j] > thresh else "black")0

plt.tight_layout()0

plt.ylabel('True label')0

plt.xlabel('Predicted label')0

cm = confusion_matrix(Y_test, y_pred_svm)0

target_names = ["Intrusion detected", "Intrusion Not Detected"]0

plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')0

accuracy_score(Y_test, y_pred_svm)0

0.8548557819528976

# plot for SVM0

false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(Y_test, y_pred_svm)0

print('roc_auc_score for SVM: ', roc_auc_score(Y_test, y_pred_svm))0

plt.subplots(1, figsize=(10,10))0

plt.title('Receiver Operating Characteristic - SVM')0

plt.plot(false_positive_rate1, true_positive_rate1)0

plt.plot([0, 1], ls="--")0

plt.plot([0, 0], [1, 0], c=".7"), plt.plot([1, 1], c=".7")0

plt.ylabel('True Positive Rate')0

plt.xlabel('False Positive Rate')0

```

```

plt.show()

roc_auc_score for SVM:      0.85455334082529920

#KNN model

model_KNN = KNeighborsClassifier()

model_KNN.fit(X_train, Y_train)

y_pred_KNN = model_KNN.predict(X_test)

print(classification_report(Y_test, y_pred_KNN))

/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198:
DataConversionWarning: A column-vector return self._fit(X, y)

precision    recall  f1-score   support

0         0.97    0.98    0.98    34980
1         0.98    0.98    0.98    40600

               accuracy                   0.98    75580
               macro avg              0.98    0.98    0.98    75580
               weighted avg  0.98    0.98    0.98    75580

#=====CONSTRUCTING THE CONFUSION
MATRIX=====#

def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',
cmap=plt.cm.Greens):
    plt.figure(figsize=(10,10))

    plt.imshow(cm, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick_marks = np.arange(len(classes))

    plt.xticks(tick_marks, classes, rotation=45)

```

```

plt.yticks(tick_marks, classes)

if normalize:

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    cm = np.around(cm, decimals=2)

    cm[np.isnan(cm)] = 0.00

    thresh = cm.max() / 2.0

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

        plt.text(j, i, cm[i, j],

            horizontalalignment="center",

            color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

cm = confusion_matrix(Y_test, y_pred_KNN)

target_names = ["Intrusion Detected", "Intrusion Not Detected"]

plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')

accuracy_score(Y_test, y_pred_KNN)

0.978036517597248

# plot for KNN

false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(Y_test, y_pred_KNN)

print('roc_auc_score for KNN: ', roc_auc_score(Y_test, y_pred_KNN))

plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic - SVM')

plt.plot(false_positive_rate1, true_positive_rate1)

plt.plot([0, 1], ls="--")

plt.plot([0, 0], [1, 0], c=".7"), plt.plot([1, 1], c=".7")

plt.ylabel('True Positive Rate')

```



```
plt.xlabel('False Positive Rate')0
```

```
plt.show()0
```

```
roc_auc_score for KNN: 0.9781518362357660
```

```
model_DT = DecisionTreeClassifier(criterion = 'entropy', random_state = 100, max_depth = 3)0  
model_DT.fit(X_train, Y_train)0
```

```
y_pred_DT = model_DT.predict(X_test)0
```

```
print(classification_report(Y_test, y_pred_DT))0
```

	precision	recall	f1-score	support
0	0.68	0.67	0.68	34980
1	0.72	0.72	0.72	40600
accuracy			0.70	75580
macro avg	0.70	0.70	0.70	75580
weighted avg	0.70	0.70	0.70	75580

```
#=====CONSTRUCTING THE CONFUSION  
MATRIX=====#0
```

```
def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',  
cmap=plt.cm.Greens):0 0
```

```
plt.figure(figsize=(10,10))0
```

```
plt.imshow(cm, interpolation='nearest', cmap=cmap)0
```

```
plt.title(title)0
```

```
plt.colorbar()0
```

```
tick_marks = np.arange(len(classes))0
```

```
plt.xticks(tick_marks, classes, rotation=45)0
```

```
plt.yticks(tick_marks, classes)0
```

```
if normalize:0
```

```

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
cm = np.around(cm, decimals=2)
cm[np.isnan(cm)] = 0.00
thresh = cm.max() / 2.0
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j],
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')

cm = confusion_matrix(Y_test, y_pred_DT)
target_names = ["Intrusion Detected", "Intrusion Not Detected"]
plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')
accuracy_score(Y_test, y_pred_DT)
0.6999206139190262

# plot for KNN
false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(Y_test, y_pred_DT)
print('roc_auc_score for KNN: ', roc_auc_score(Y_test, y_pred_DT))
plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic - SVM')
plt.plot(false_positive_rate1, true_positive_rate1)
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0], c=".7"), plt.plot([1, 1], c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()

```

roc_auc_score for KNN: 0.69813348655248460

Colab paid products0 -0 Cancel contracts here

! pip install tqdm update_checker tqdm

! pip install tpot

import matplotlib

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import seaborn as sns

import sklearn

import imblearn

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.feature_selection import RFE

import itertools

from sklearn.model_selection import train_test_split

from tpot import TPOTRegressor

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
roc_auc_score, roc_curve

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/0>

Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (4.64.1)0

Collecting update_checker0

Downloading update_checker-0.18.0-py3-none-any.whl (7.0 kB)0

Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.7/dist-packages (from update_checker) (2.23.0)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->update_checker)

Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->update_checker)

Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->update_checker)

Requirement already satisfied: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->update_checker)

Installing collected packages: update-checker0

Successfully installed update-checker-0.18.00

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/0> Collecting tpot0

Downloading TPOT-0.11.7-py3-none-any.whl (87 kB)0

|| 87 kB 5.4 MB/s 0

Collecting xgboost>=1.1.00

Downloading xgboost-1.6.2-py3-none-manylinux2014_x86_64.whl (255.9 MB)0 || 255.9 MB 49 kB/s 0

Collecting stopit>=1.1.10

Downloading stopit-1.1.2.tar.gz (18 kB)0

Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.7/dist-packages (from tpot) (1.3.5)0

Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.7/dist-packages (from tpot) (1.1.0)0

Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.7/dist-packages (from tpot) (1.7.3)0

Requirement already satisfied: scikit-learn>=0.22.0 in /usr/local/lib/python3.7/dist-packages (from tpot) (1.0.2)0 Collecting deap>=1.20

Downloading deap-1.3.3-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl 139 kB 61.8 MB/s 0

Requirement already satisfied: update-checker>=0.16 in /usr/local/lib/python3.7/dist-packages (from tpot) (0.18.0)0

Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.7/dist-packages (from tpot) (4.64.1)0

Requirement already satisfied: numpy>=1.16.3 in /usr/local/lib/python3.7/dist-packages (from tpot) (1.21.6)0

Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.2->tpot) (2022.2.1)0

Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.2->tpot) (2.8.2)0

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas>=0.24.2->tpot) (1.16.0)0

Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.22.0->tpot) (2.2.1)0

Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.7/dist-packages (from update-checker>=0.16->tpot) (2.28.1)0

Requirement already satisfied: idna<3, >=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->update-checker>=0.16->tpot) (3.3)0

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages
(from requests>=2.3.0->upda

Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages
(from requests>=2.3.0->updat

Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from

Building wheels for collected packages: stopit0

Building wheel for stopit (setup.py) ... done0

Created wheel for stopit: filename=stopit-1.1.2-py3-none-any.whl size=11956
sha256=b601715034894cc7475294bd53c66dc7fb

Stored in directory:
/root/.cache/pip/wheels/e2/d2/79/eaf81edb391e27c87f51b8ef901ecc85a5363dc96b8b8d71e30

Successfully built stopit0

Installing collected packages: xgboost, stopit, deap, tpot0

Attempting uninstall: xgboost0

Found existing installation: xgboost 0.900

Uninstalling xgboost-0.90:0

Successfully uninstalled xgboost-0.900

Successfully installed deap-1.3.3 stopit-1.1.2 tpot-0.11.7 xgboost-1.6.20

#Load Data0

train = pd.read_csv(r'/content/Train_data.csv')0

test = pd.read_csv(r'/content/Test_data.csv')0

print("Training data has {} rows & {} columns".format(train.shape[0],train.shape[1]))0

print(test.head(4))0

print("Testing data has {} rows & {} columns".format(test.shape[0],test.shape[1]))0

train.describe()0

```
#print(train['num_outbound_cmds'].value_counts())0
```

```
#print(test['num_outbound_cmds'].value_counts())0
```

```
#'num_outbound_cmds' is a redundant column so remove it from both train & test datasets0
```

```
train.drop(['num_outbound_cmds'], axis=1, inplace=True)0 test.drop(['num_outbound_cmds'],
axis=1, inplace=True)0
```

```
train['class'].value_counts()0
```

Training data has 25192 rows & 42 columns0

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	\0
0	0	tcp	private	REJ	0	0	0	0
1	0	tcp	private	REJ	0	0	0	0
2	2	tcp	ftp_data	SF	12983	0	0	0
3	0	icmp	eco_i	SF	20	0	0	0
	wrong_fragment	urgent	hot	...	dst_host_count	dst_host_srv_count	\0	
0	0	0	0	...	255	10	0	
1	0	0	0	...	255	1	0	
2	0	0	0	...	134	86	0	
3	0	0	0	...	3	57	0	
	dst_host_same_srv_rate	dst_host_diff_srv_rate	\0					
0	0.04	0.06	0					
1	0.00	0.06	0					
2	0.61	0.04	0					
3	1.00	0.00	0					
	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	\0					
0	0.00	0.00	0					
1	0.00	0.00	0					

2 0.61 0.02 0

3 1.00 0.28 0

dst_host_serror_rate dst_host_srv_serror_rate dst_host_error_rate \0

0 0.0 0.0 1.0 0

1 0.0 0.0 1.0 0

2 0.0 0.0 0.0 0

3 0.0 0.0 0.0 0

dst_host_srv_error_rate 0

0 1.0 0

1 1.0 0

2 0.0 0

3 0.0 0

[4 rows x 41 columns]0

Testing data has 22544 rows & 41 columns0

normal 134490

anomaly 117430

Name: class, dtype: int64

scaler = StandardScaler()0

extract numerical attributes and scale it to have zero mean and unit variance 0 cols =

train.select_dtypes(include=['float64','int64']).columns0

sc_train = scaler.fit_transform(train.select_dtypes(include=['float64','int64']))0 sc_test =

scaler.fit_transform(test.select_dtypes(include=['float64','int64']))0

turn the result back to a dataframe0

sc_traindf = pd.DataFrame(sc_train, columns = cols)0

sc_testdf = pd.DataFrame(sc_test, columns = cols)0

encoder = LabelEncoder()0


```

# extract categorical attributes from both training and test sets
cattrain = train.select_dtypes(include=['object']).copy()
cattest = test.select_dtypes(include=['object']).copy()

# encode the categorical attributes
traincat = cattrain.apply(encoder.fit_transform)
testcat = cattest.apply(encoder.fit_transform)

# separate target column from encoded data
enctrain = traincat.drop(['class'], axis=1)
cat_Ytrain = traincat[['class']].copy()

# Union of processed numerical and categorical data
train_x = pd.concat([sc_traindf, enctrain], axis=1)
train_y = cat_Ytrain

train_x.shape
(22544, 40)

test_df = pd.concat([sc_testdf, testcat], axis=1)
test_df.shape
(22544, 40)

rfc = RandomForestClassifier()

# fit random forest classifier on the training set
rfc.fit(train_x, train_y)

# extract important features
score = np.round(rfc.feature_importances_, 3)
importances = pd.DataFrame({'feature': train_x.columns, 'importance': score})
importances = importances.sort_values('importance', ascending=False).set_index('feature')

# plot importances
plt.rcParams['figure.figsize'] = (11, 4)
importances.plot.bar()

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DataConversionWarning:
after removing the cwd from sys.path.

```

```

# create the RFE model and select 10 attributes0 rfe = RFE(rfc,
n_features_to_select=15)0
rfe = rfe.fit(train_x, train_y)0
# summarize the selection of the attributes0
feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), train_x.columns)]0
selected_features = [v for i, v in feature_map if i==True]0
selected_features0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

```


/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:

DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:310:

DataConversionWarning: A column-vector self.estimator_.fit(X[:, features], y, **fit_params)0

['src_bytes',0

'dst_bytes',0

'hot',0

'count',0

'srv count',

a = [i[0] for i in feature_map]0

train_x = train_x.iloc[:,a]0

test_df = test_df.iloc[:,a]0

train_x.shape0

(25192, 15)

tpot = TPOTRegressor(generations=5, population_size=50, verbosity=2)0

tpot.fit(train_x, train_y)0

#print(tpot.score(x_test, y_test))0

/usr/local/lib/python3.7/dist-packages/sklearn/utils/evaluation.py:993: DataConversionWarning: A column-vector y =
column_or_1d(y, warn=True)0

Generation 1 - Current best internal CV score: -0.0034427046312751040

Generation 2 - Current best internal CV score: -0.00302676440162130430

Generation 3 - Current best internal CV score: -0.00298706413909575940

Generation 4 - Current best internal CV score: -0.0028306168618364110

Generation 5 - Current best internal CV score: -0.0028306168618364110

Best pipeline: ExtraTreesRegressor(ExtraTreesRegressor(MaxAbsScaler(input_matrix),

X_train,X_test,Y_train,Y_test = train_test_split(train_x,train_y,train_size=0.70,

random_state=2)0 input_shape = [X_train.shape[1]]0

from sklearn.svm import SVC0

#SVM MODEL0

model_svm = SVC(random_state=1)0

model_svm.fit(X_train, Y_train)0

y_pred_svm = model_svm.predict(X_test)0

print(classification_report(Y_test, y_pred_svm))0

/usr/local/lib/python3.7/dist-packages/sklearn/utils/evaluation.py:993:

DataConversionWarning: A column-vector y was passed as a column or 1d array, which may be致使 misinterpretation, please use y = column_or_1d(y, warn=True)0

		precision	recall	f1-score	support
--	--	-----------	--------	----------	---------

0	0.98	0.91	0.94	34980
---	------	------	------	-------

1	0.93	0.98	0.95	40600
---	------	------	------	-------

	accuracy			0.95	75580
--	----------	--	--	------	-------

	macro avg	0.95	0.95	0.95	75580
--	-----------	------	------	------	-------

	weighted avg	0.95	0.95	0.95	75580
--	--------------	------	------	------	-------

#=====CONSTRUCTING THE CONFUSION

MATRIX=====#0

def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',

cmap=plt.cm.Reds):0 0

plt.figure(figsize=(10,10))0

plt.imshow(cm, interpolation='nearest', cmap=cmap)0

```

plt.title(title)

plt.colorbar()

tick_marks = np.arange(len(classes))

plt.xticks(tick_marks, classes, rotation=45)

plt.yticks(tick_marks, classes)

if normalize:

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    cm = np.around(cm, decimals=2)

    cm[np.isnan(cm)] = 0.00

    thresh = cm.max() / 2.0

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

        plt.text(j, i, cm[i, j],

            horizontalalignment="center",

            color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

cm = confusion_matrix(Y_test, y_pred_svm)

target_names = ["Intrusion detected", "Intrusion Not Detected"]

plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')

accuracy_score(Y_test, y_pred_svm)

0.949457528446679

# plot for SVM

false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(Y_test, y_pred_svm)

print('roc_auc_score for SVM: ', roc_auc_score(Y_test, y_pred_svm))

plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic - SVM')

```

```

plt.plot(false_positive_rate1, true_positive_rate1)0
plt.plot([0, 1], ls="--")0
plt.plot([0, 0], [1, 0], c=".7"), plt.plot([1, 1], c=".7")0
plt.ylabel('True Positive Rate')0
plt.xlabel('False Positive Rate')0
plt.show()0

roc_auc_score for SVM:    0.94674282559773770

#KNN model0
model_KNN = KNeighborsClassifier()0
model_KNN.fit(X_train, Y_train)0
y_pred_KNN = model_KNN.predict(X_test)0
print(classification_report(Y_test, y_pred_KNN))0

/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198:
DataConversionWarning: A column-vector return self._fit(X, y)0

      precision    recall  f1-score   support0

0   0.99    0.99    0.99    34980
1   0.99    0.99    0.99    40600

 accuracy          0.99    75580

 macro avg         0.99    0.99    0.99    75580

weighted avg    0.99    0.99    0.99    75580

#=====CONSTRUCTING THE CONFUSION
MATRIX=====#0

def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',
cmap=plt.cm.Greens):0 0

plt.figure(figsize=(10,10))0

plt.imshow(cm, interpolation='nearest', cmap=cmap)0

```

```

plt.title(title)

plt.colorbar()

tick_marks = np.arange(len(classes))

plt.xticks(tick_marks, classes, rotation=45)

plt.yticks(tick_marks, classes)

if normalize:

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    cm = np.around(cm, decimals=2)

    cm[np.isnan(cm)] = 0.00

    thresh = cm.max() / 2.0

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

        plt.text(j, i, cm[i, j],

            horizontalalignment="center",

            color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

cm = confusion_matrix(Y_test, y_pred_KNN)

target_names = ["Intrusion Detected", "Intrusion Not Detected"]

plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')

accuracy_score(Y_test, y_pred_KNN)

0.9891505689335803

# plot for KNN

false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(Y_test, y_pred_KNN)

print('roc_auc_score for KNN: ', roc_auc_score(Y_test, y_pred_KNN))

plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic - SVM')

```



```

plt.plot(false_positive_rate1, true_positive_rate1)0
plt.plot([0, 1], ls="--")0
plt.plot([0, 0], [1, 0], c=".7"), plt.plot([1, 1], c=".7")0
plt.ylabel('True Positive Rate')0
plt.xlabel('False Positive Rate')0
plt.show()0

roc_auc_score for KNN:    0.98911003331953230

model_DT = DecisionTreeClassifier(criterion = 'entropy', random_state = 100, max_depth =
3)0 model_DT.fit(X_train, Y_train)0
y_pred_DT = model_DT.predict(X_test)0
print(classification_report(Y_test, y_pred_DT))0

      precision    recall  f1-score   support0

0       0.97     0.93     0.95     34980
1       0.94     0.98     0.96     40600

accuracy                   0.95     75580
macro avg       0.95     0.95     0.95     75580
weighted avg    0.95     0.95     0.95     75580

#=====CONSTRUCTING THE CONFUSION
MATRIX=====#0

def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',
cmap=plt.cm.Greens):0 0

plt.figure(figsize=(10,10))0

plt.imshow(cm, interpolation='nearest', cmap=cmap)0

plt.title(title)0

plt.colorbar()0

tick_marks = np.arange(len(classes))0

```

```

plt.xticks(tick_marks, classes, rotation=45)

plt.yticks(tick_marks, classes)

if normalize:

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    cm = np.around(cm, decimals=2)

    cm[np.isnan(cm)] = 0.00

    thresh = cm.max() / 2.0

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

        plt.text(j, i, cm[i, j],

            horizontalalignment="center",

            color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

cm = confusion_matrix(Y_test, y_pred_DT)

target_names = ["Intrusion Detected", "Intrusion Not Detected"]

plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')

accuracy_score(Y_test, y_pred_DT)

0.9523683514157184

# plot for KNN

false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(Y_test, y_pred_DT)

print('roc_auc_score for KNN: ', roc_auc_score(Y_test, y_pred_DT))

plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic - SVM')

plt.plot(false_positive_rate1, true_positive_rate1)

plt.plot([0, 1], ls="--")

plt.plot([0, 0], [1, 0], c=".7"), plt.plot([1, 1], c=".7")

```

```
plt.ylabel('True Positive Rate')0
```

```
plt.xlabel('False Positive Rate')0
```

```
plt.show()0
```

```
roc_auc_score for KNN: 0.9504810630705230
```

Colab paid products0 -0 Cancel contracts here

```
! pip install tqdm update_checker tqdm
```

```
! pip install tpot
```

```
import matplotlib
```

```
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

```
import numpy as np
```

```
import seaborn as sns
```

```
import sklearn
```

```
import imblearn
```

```
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.feature_selection import RFE
```

```
import itertools
```

```
from sklearn.model_selection import train_test_split
```

```
from tpot import TPOTRegressor
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
```

```
roc_auc_score, roc_curve
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.manifold import LocallyLinearEmbedding, Isomap
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (4.64.1)

Collecting update_checker

Downloading update_checker-0.18.0-py3-none-any.whl (7.0 kB)

Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.7/dist-packages (from update_checker) (2.23.0)

Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->update_checker)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->update_checker)

Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->update_checker)

Requirement already satisfied: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->update_checker)

Installing collected packages: update-checker

Successfully installed update-checker-0.18.0

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Collecting tpot

Downloading TPOT-0.11.7-py3-none-any.whl (87 kB)

| 87 kB 5.8 MB/s

Requirement already satisfied: scikit-learn>=0.22.0 in /usr/local/lib/python3.7/dist-packages (from tpot) (1.0.2)

Collecting xgboost>=1.1.0

Downloading xgboost-1.6.2-py3-none-manylinux2014_x86_64.whl (255.9 MB)

| 255.9 MB 60 kB/s

Requirement already satisfied: numpy>=1.16.3 in /usr/local/lib/python3.7/dist-packages (from tpot) (1.21.6)0

Requirement already satisfied: update-checker>=0.16 in /usr/local/lib/python3.7/dist-packages (from tpot) (0.18.0)0

Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.7/dist-packages (from tpot) (1.7.3)0

Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.7/dist-packages (from tpot) (4.64.1)0 Collecting stopit>=1.1.10

Downloading stopit-1.1.2.tar.gz (18 kB)0

Collecting deap>=1.20

Downloading deap-1.3.3-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl || 139 kB 85.7 MB/s 0

Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.7/dist-packages (from tpot) (1.3.5)0

Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.7/dist-packages (from tpot) (1.1.0)0

Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.2->tpot)

Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.2->tpot) (2022)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas>=0.24.2->tpot)

Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.22.0->tpot)

Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.7/dist-packages (from update-checker>=0.16->tp

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->upda

Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->update-che

Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.3.0->updat

Requirement already satisfied: urllib3!=1.25.0,!>=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from

Building wheels for collected packages: stopit0

Building wheel for stopit (setup.py) ... done0

Created wheel for stopit: filename=stopit-1.1.2-py3-none-any.whl size=11956 sha256=d2773ecc8a947ccd30f566f319422ecd0c

Stored in directory: /root/.cache/pip/wheels/e2/d2/79/eaf81edb391e27c87f51b8ef901ecc85a5363dc96b8b8d71e30

Successfully built stopit0

Installing collected packages: xgboost, stopit, deap, tpot0

Attempting uninstall: xgboost0

Found existing installation: xgboost 0.900

Uninstalling xgboost-0.90:0

Successfully uninstalled xgboost-0.900

Successfully installed deap-1.3.3 stopit-1.1.2 tpot-0.11.7 xgboost-1.6.20

#Load Data0

train = pd.read_csv(r'/content/Train_data.csv')0

test = pd.read_csv(r'/content/Test_data.csv')0

```

print("Training data has {} rows & {} columns".format(train.shape[0],train.shape[1]))0

print(test.head(4))0

print("Testing data has {} rows & {} columns".format(test.shape[0],test.shape[1]))0

train.describe()0

#print(train['num_outbound_cmds'].value_counts())0

#print(test['num_outbound_cmds'].value_counts())0

# 'num_outbound_cmds' is a redundant column so remove it from both train & test datasets0

train.drop(['num_outbound_cmds'], axis=1, inplace=True)0 test.drop(['num_outbound_cmds'],
axis=1, inplace=True)0

train['class'].value_counts()0

Training data has 25192 rows & 42 columns0

```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	\0
0	0	tcp	private REJ	0	0	0	0	
1	0	tcp	private REJ	0	0	0	0	
2	2	tcp	ftp_data	SF	12983	0	0	0
3	0	icmp	eco_i SF	20	0	0	0	
	wrong_fragment	urgent	hot ...	dst_host_count	dst_host_srv_count	\0		
0	0	0	0 ...	255	10	0		
1	0	0	0 ...	255	1	0		
2	0	0	0 ...	134	86	0		
3	0	0	0 ...	3	57	0		
	dst_host_same_srv_rate	dst_host_diff_srv_rate	\0					
0	0.04	0.06	0					
1	0.00	0.06	0					
2	0.61	0.04	0					

3 1.00 0.00 0

dst_host_same_src_port_rate dst_host_srv_diff_host_rate \0

0 0.00 0.00 0

1 0.00 0.00 0

2 0.61 0.02 0

3 1.00 0.28 0

dst_host_serror_rate dst_host_srv_serror_rate dst_host_rerror_rate \0

0 0.0 0.0 1.0 0

1 0.0 0.0 1.0 0

2 0.0 0.0 0.0 0

3 0.0 0.0 0.0 0

dst_host_srv_rerror_rate 0

0 1.0 0

1 1.0 0

2 0.0 0

3 0.0 0

[4 rows x 41 columns]0

Testing data has 22544 rows & 41 columns0

normal 134490

anomaly 117430

Name: class, dtype: int64

scaler = StandardScaler()0

extract numerical attributes and scale it to have zero mean and unit variance 0 cols =

train.select_dtypes(include=['float64','int64']).columns0

sc_train = scaler.fit_transform(train.select_dtypes(include=['float64','int64']))0 sc_test =

scaler.fit_transform(test.select_dtypes(include=['float64','int64']))0


```

# turn the result back to a dataframe
sc_traindf = pd.DataFrame(sc_train, columns = cols)
sc_testdf = pd.DataFrame(sc_test, columns = cols)

encoder = LabelEncoder()

# extract categorical attributes from both training and test sets
cattrain = train.select_dtypes(include=['object']).copy()
cattest = test.select_dtypes(include=['object']).copy()

# encode the categorical attributes
traincat = cattrain.apply(encoder.fit_transform)
testcat = cattest.apply(encoder.fit_transform)

# separate target column from encoded data
enctrain = traincat.drop(['class'], axis=1)
cat_Ytrain = traincat[['class']].copy()

# Union of processed numerical and categorical data
train_x = pd.concat([sc_traindf, enctrain], axis=1)
train_y = cat_Ytrain

train_x.shape

test_df = pd.concat([sc_testdf, testcat], axis=1)
test_df.shape

(22544, 40)

rfc = RandomForestClassifier();

# fit random forest classifier on the training set
rfc.fit(train_x, train_y);

# extract important features

score = np.round(rfc.feature_importances_, 3)

importances = pd.DataFrame({'feature': train_x.columns, 'importance': score})

importances = importances.sort_values('importance', ascending=False).set_index('feature')

# plot importances
plt.rcParams['figure.figsize'] = (11, 4)
importances.plot.bar();

```

```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DataConversionWarning: A
column-vector y was passed whe after removing the cwd from sys.path.0

# create the RFE model and select 10 attributes0 rfe = RFE(rfc,
n_features_to_select=15)0

rfe = rfe.fit(train_x, train_y)0

# summarize the selection of the attributes0

feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), train_x.columns)]0
selected_features = [v for i, v in feature_map if i==True]0

selected_features0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0

```



```

/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:283:
DataConversionWarning: A column-vector estimator.fit(X[:, features], y, **fit_params)0
/usr/local/lib/python3.7/dist-packages/sklearn/feature_selection/_rfe.py:310:
DataConversionWarning: A column-vector self.estimator_.fit(X[:, features], y,
**fit_params)0
['src_bytes',0
'dst_bytes',0
'logged_in',0
'count',0
'srv count'
a = [i[0] for i in feature_map]0
train_x = train_x.iloc[:,a]0
test_df = test_df.iloc[:,a]0
train_x.shape0
(25192, 15)
tpot = TPOTRegressor(generations=5, population_size=50, verbosity=2)0
tpot.fit(train_x, train_y)0
#print(tpot.score(x_test, y_test))0
/usr/local/lib/python3.7/dist-packages/sklearn/utils/evaluation.py:993:
DataConversionWarning: A column-vector y was passed as a column or 1d(y, warn=True)0

```

Generation 1 - Current best internal CV score: -0.00326564277539670430

Generation 2 - Current best internal CV score: -0.00324256318779018050

Generation 3 - Current best internal CV score: -0.00298497929819368680

Generation 4 - Current best internal CV score: -0.0024884421766277030

Generation 5 - Current best internal CV score: -0.0024884421766277030

Best pipeline: RandomForestRegressor(input_matrix, bootstrap=False, max_features=0.35000000000000003, min_samples_leaf= TPOTRegressor(generations=5, population_size=50, verbosity=2)

embedding = LocallyLinearEmbedding(n_components=8, eigen_solver = 'dense')0

Train_x_lle = embedding.fit_transform(train_x)0

X_train,X_test,Y_train,Y_test = train_test_split(Train_x_lle,train_y,train_size=0.70, random_state=2)0

input_shape = [X_train.shape[1]]0

from sklearn.svm import SVC

#SVM MODEL

model_svm = SVC(random_state=1)

model_svm.fit(X_train, Y_train)

y_pred_svm = model_svm.predict(X_test)

print(classification_report(Y_test, y_pred_svm))

/usr/local/lib/python3.7/dist-packages/sklearn/utils/evaluation.py:993:

DataConversionWarning: A column-vector y was passed as a column or 1d(y, warn=True)0

	precision	recall	f1-score	support
0	0.86	0.92	0.88	34980
1	0.92	0.87	0.89	40600
accuracy			0.89	75580
macro avg	0.89	0.89	0.89	75580

weighted avg 0.89 0.89 0.89 75580

```
#=====CONSTRUCTING THE CONFUSION  
MATRIX=====#
```

```
def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',  
cmap=plt.cm.Reds):
```

```
plt.figure(figsize=(10,10))
```

```
plt.imshow(cm, interpolation='nearest', cmap=cmap)
```

```
plt.title(title)
```

```
plt.colorbar()
```

```
tick_marks = np.arange(len(classes))
```

```
plt.xticks(tick_marks, classes, rotation=45)
```

```
plt.yticks(tick_marks, classes)
```

```
if normalize:
```

```
cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
```

```
cm = np.around(cm, decimals=2)
```

```
cm[np.isnan(cm)] = 0.0
```

```
thresh = cm.max() / 2.
```

```
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
```

```
plt.text(j, i, cm[i, j],
```

```
horizontalalignment="center",
```

```
color="white" if cm[i, j] > thresh else "black")
```

```
plt.tight_layout()
```

```
plt.ylabel('True label')
```

```
plt.xlabel('Predicted label')
```

```
cm = confusion_matrix(Y_test, y_pred_svm)
```

```
target_names = ["Intrusion detected", "Intrusion Not Detected"]
```

```

plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')

accuracy_score(Y_test, y_pred_svm)

0.8897856575813707

# plot for SVM0

false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(Y_test, y_pred_svm)0

print('roc_auc_score for SVM: ', roc_auc_score(Y_test, y_pred_svm))0

plt.subplots(1, figsize=(10,10))0

plt.title('Receiver Operating Characteristic - SVM')0

plt.plot(false_positive_rate1, true_positive_rate1)0

plt.plot([0, 1], ls="--")0

plt.plot([0, 0], [1, 0], c=".7"), plt.plot([1, 1], c=".7")0

plt.ylabel('True Positive Rate')0

plt.xlabel('False Positive Rate')0

plt.show()0

roc_auc_score for SVM:      0.89155710370739650

#KNN model

model_KNN = KNeighborsClassifier()

model_KNN.fit(X_train, Y_train)

y_pred_KNN = model_KNN.predict(X_test)

print(classification_report(Y_test, y_pred_KNN))

```

/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198:

DataConversionWarning: A column-vector return self._fit(X, y)0

		precision	recall	f1-score	support0
0	0.97	0.98	0.98	34980	
1	0.98	0.98	0.98	40600	
	accuracy			0.98	75580

```

macro avg    0.98    0.98    0.98    75580

weighted avg 0.98    0.98    0.98    75580

#=====CONSTRUCTING THE CONFUSION
MATRIX=====#

def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',
cmap=plt.cm.Greens):

plt.figure(figsize=(10,10))

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick_marks = np.arange(len(classes))

plt.xticks(tick_marks, classes, rotation=45)

plt.yticks(tick_marks, classes)

if normalize:

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

cm = np.around(cm, decimals=2)

cm[np.isnan(cm)] = 0.0

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, cm[i, j],

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

cm = confusion_matrix(Y_test, y_pred_KNN)

```



```

target_names = ["Intrusion Detected", "Intrusion Not Detected"]

plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')

accuracy_score(Y_test, y_pred_KNN)

0.9771103466525536

# plot for KNN

false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(Y_test, y_pred_KNN)

print('roc_auc_score for KNN: ', roc_auc_score(Y_test, y_pred_KNN))

plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic - SVM')

plt.plot(false_positive_rate1, true_positive_rate1)

plt.plot([0, 1], ls="--")

plt.plot([0, 0], [1, 0], c=".7"), plt.plot([1, 1], c=".7")

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

roc_auc_score for KNN: 0.97715126448047710

model_DT = DecisionTreeClassifier(criterion = 'entropy', random_state = 100, max_depth = 3)

model_DT.fit(X_train, Y_train)

y_pred_DT = model_DT.predict(X_test)

print(classification_report(Y_test, y_pred_DT))


```

	precision	recall	f1-score	support
0	0.70	0.70	0.70	34980
1	0.74	0.74	0.74	40600
accuracy			0.72	75580
macro avg	0.72	0.72	0.72	75580
weighted avg	0.72	0.72	0.72	75580

```

#=====CONSTRUCTING THE CONFUSION
MATRIX=====#0

def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',
cmap=plt.cm.Greens):0 0

plt.figure(figsize=(10,10))0

plt.imshow(cm, interpolation='nearest', cmap=cmap)0

plt.title(title)0

plt.colorbar()0

tick_marks = np.arange(len(classes))0

plt.xticks(tick_marks, classes, rotation=45)0

plt.yticks(tick_marks, classes)0

if normalize:0

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]0

cm = np.around(cm, decimals=2)0

cm[np.isnan(cm)] = 0.00

thresh = cm.max() / 2.0

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):0 plt.text(j, i, cm[i, j],0
horizontalalignment="center",0
color="white" if cm[i, j] > thresh else "black")0

plt.tight_layout()0

plt.ylabel('True label')0

plt.xlabel('Predicted label')0

cm = confusion_matrix(Y_test, y_pred_DT)0

target_names = ["Intrusion Detected", "Intrusion Not Detected"]0

plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')0

accuracy_score(Y_test, y_pred_DT)

```

0.7208256152421275

plot for Decision tree

false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(Y_test, y_pred_DT)

print('roc_auc_score for KNN: ', roc_auc_score(Y_test, y_pred_DT))

plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic - SVM')

plt.plot(false_positive_rate1, true_positive_rate1)

plt.plot([0, 1], ls="--")

plt.plot([0, 0], [1, 0], c=".7"), plt.plot([1, 1], c=".7")

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

roc_auc_score for KNN: 0.71917450365726240

Colab paid products -0 Cancel contracts here

Biodata

A. Personal Data

Full name: Tolulope Olushola OLUFEMI
Address: 7 Akinyele Street, Aguda Surulere, Lagos
Email Address: tolubiks67@gmail.com
Phone Number: 07035269331
Date of Birth: 20/05/1972
Place of Birth: Epe Ijero Ekiti
Nationality: Nigerian
Name of Next of kin: Olufemi Joy Ayomide

B. Educational Background

Primary education

✓ Baptist Primary School, Epe Ekiti, Ekiti State 1982

Secondary education

✓ Onike Girls High School, Iwaya Lagos 1989 WASSCE

Higher education

✓ Lead City University, Ibadan, PhD in Computer Science 2020 – Present

✓ Lead City University, Ibadan, M.Sc. in Computer Science 2020

✓ Abubakar Tafawa University, Bauchi, PGD Computer Science 1999

✓ Ondo State University (now EKSU), B.Sc. Computer Science 1997

China Europe International Business School, Shanghai China, Diploma in Leadership and International Business 2015

C. Membership Organization

✓ Computer Registration Council of Nigeria (MCPN) 008897/2023

✓ Nigeria Computer Society (MNCS) 14897

✓ International Society of Applied Computing (ISAC) RM2023103001

- ✓ Institute of Electrical & Electronics Engineers. (IEEE) International. Member
- ✓ Society for Multidisciplinary & Advanced Research Techniques Member. (SMART) Africa – Regional. Member.

D. Working Experience with Dates

- ✓ TechWorld Limited 34 Bukuru Bypass, Jos. August 1999 –November 2002
- ✓ AST Teletronics Limited 13 Association Avenue, Ilupeju Lagos. February 2003- March 2006
- ✓ Global Tech Services Limited plot 345 Kofo Abayomi Street, VI, Lagos. April 2006- December 2008
- ✓ HINET Limited 76 Opebi Road, Ikeja Lagos. January 2009- January 2013
- ✓ Landslide Technologies Services Limited. March 2013-Date

E. Publication

Thesis/Dissertation

- ✓ Design and Implementation of LAN on Campus (OSUA) (BSc Project)
- ✓ Comparative Analysis of Machine-Learning Based Intrusion Detection Systems.

Articles completed but not yet Published

- ✓ Enhancing Intrusion Detection Systems using Local Linear Embedding Algorithm: A Comparative Study with Machine Learning Classifiers.
- ✓ Optimized-Improved Genetic Algorithm and Advanced Classification Techniques for Efficient Big Data Analysis.

Conference Proceedings

- ✓ An improved Generic Algorithm for classifying Intrusion Detection. 4th International Conference of Applied ICT(ICAICT) Lead City University. Ibadan 2022
- ✓ Comparative Analysis of Machine-Learning Based Intrusion Detection
Air Force Institute of Technology Kaduna First International Conference On Data Science And Engineering. December, 2021. Journal of Software Engineering.
- ✓ On the Comparative Analysis of different Machine-Learning Based Intrusion Detection Algorithms for Training Intrusion Detection Systems. Proceedings of the 25th iSTEAMS Trans-

Atlantic Multidisciplinary Virtual Conference, Laboratoire Jean Kuntzmann, Universite Laboratoire Jean Kunzmann, Universite Grenoble, Alpes, France. June 2020.

- ✓ A Survey of Machine-Learning Based IDS.2nd International Conference of Applied ICT(ICAICT) Lead City University. Ibadan 2019.

Journal

- ✓ A Survey on Deep Learning Models for Cyber Threat Detection Systems, EasyChair, No. 9849, 2023.
- ✓ Development of Cardiac Monitoring Device Using Android Mobile Phone, Digital Innovations & Contemporary Research in Science, Engineering and Technology, 2023
https://www.isteam.net/_files/ugd/185b0a_e7b30a8e3f0c4a45b0a5f1e539622436.pdf
- ✓ Advances in Intrusion Detection Systems Using Feature Dimensionality and Convolutional-LSTM Network. Advances in Multidisciplinary & Scientific Research Journal. Vol 7. No. 1. 2021.
www.isteam.net/aimsjournal
- ✓ An Improved Approach for Generating Test Cases during Model-Based Testing Using Tree Traversal Algorithm. Journal of Software Engineering and Applications, 14, 2021.257-265.
<https://doi.org/10.4236/jsea.146015>
- ✓ An Analysis of the Limitations of Existing Data Classification Techniques Adopted in Designing Intrusion Detection Systems Journal of Digital Innovations & Contemporary Research in Science, Engineering & Technology. Journal Vol 9 No 2. 2021. www.isteam.net/digitaljournal
- ✓ Next generation Intrusion Detection and challenges in Intrusion Detection for Web Applications. Advances in Mathematical & Computational Sciences Journal. Vol 8, No. 3. 2020.
<https://www.isteam.net/mathematics-computationaljournal>
- ✓ Machine Learning and Classifier Techniques for Intrusion Detection. Computing, Information Systems & Development Informatics Vol 1 2020. www.isteam.net/cisdijournal

F. Professional Accomplishments

- ✓ CCNA 2003
- ✓ CCNP, Switching and Routing, 2007
- ✓ CCNP, Switching and Routing, Revalidated 2010,2013,2016,2019

- ✓ CCNP Implementing CISCO Enterprise Core Technologies, 2022

G. Major Conferences/Workshops Attended

- ✓ CISCO LIVE Conference London February 2011
- ✓ CISCO LIVE Conference Las Vegas July 2011
- ✓ CISCO LIVE Conference February London 2012
- ✓ CISCO LIVE Conference San Francisco February 2014
- ✓ CISCO LIVE Conference Las Vegas July 2017
- ✓ 2nd International Conference of Applied ICT (ICAICT), Lead City University, 2019.
- ✓ 25th iSTEAMS Trans-Atlantic Multidisciplinary Virtual Conference, Laboratoire Jean Kuntzmann, Universite Laboratoire Jean Kunzmann, Universite Grenoble, Alpes, France. June 2020.
- ✓ 1st International Conference on Data Science & Engineering, Air Force Institute of Technology Kaduna 2021.
- ✓ 4th International Faculty of Basic Medical and Applied Sciences Conference (FASCON 2022).

The University Compliance Form

This is to certify that this thesis by **Olufemi Tolulope Olushola** with Matriculation Number **LCU/PG/000138** in the Department of Computer Science, Faculty of Natural and Applied Sciences, Lead City University, Ibadan is in full compliance with the approval of the University's format and style.

Signature

Date

Olufemi_Tolulope_Olushola_Computer_Science_PhD_Post_F...
LCU LIBRARY

ORIGINALITY REPORT

5
%

SIMILARITY INDEX

6%

INTERNET SOURCES

4%

PUBLICATIONS

0%

STUDENT PAPERS

PRIMARY SOURCES

www.journaltoes.ac.uk

1

Internet Source

1%

www.researchgate.net

2

Internet Source

%

mdpi-res.com

3

Internet Source

1
%

www.hindawi.com

4

Internet Source

1
%

cybersecurity.springeropen.com

5

Internet Source

1
%

journalofbigdata.springeropen.com

6

Internet Source

1
%

Exclude quotes

On Exclude bibliography

On

Exclude matches

< 1%

Do Not Copy, Lead City University, Nigeria