Neuro Fuzzy-based Predictive Model for Keylogging Attack Mitigation

Mariam Ayobami GBADEGESIN
LCU/PG/002431

Being a MSc Thesis Submitted to the Department of Computer Science, Faculty of Natural and Applied Sciences, Lead City University, Ibadan, Oyo State, Nigeria

In Partial Fulfilment of the Requirements for the Award of Master of Science Degree (MSc) in Computer Science

2023

Certification

This is to certify that Mariam Ayobami GBADEGESIN with matriculation number LCU/PG/002431 carried out this research work titled "Neuro-fuzzy Based Predictive Model for Keylogging Attack Mitigation" in the Department of Computer Science, Faculty of Natural and Applied Sciences, Lead City University, Ibadan, Oyo state, for the award of Master Degree (MSc) in Computer Science and that this has not been previously submitted.

Prof S. O Akinola
(Supervisor)

Date

Dr Wilson Sakpere
(Head of Department)

Date

Dedication

This research work is dedicated to Almighty God for his mercies and grace in abundance upon my life and to my family members.

Acknowledgements

I would like to express my deepest gratitude to my thesis supervisor Prof Akinola for his untiring guidance and motivation in making this research work a huge success. I wish to specially thank him for being very patient and helpful to me throughout the development process of the system.

My special thanks go to My Head of Department, Dr Wilson Sakpere for his sincere critique and provision of directions on how best to approach this thesis work.

My sincere thanks go members of staff of computer science department, Lead City University Ibadan whose individual contributions influenced my final submissions in this research work.

I want to register my special thanks and gratitude to my best friend and husband, Mr Akeem Lawal for his unending prayers, support and understanding during the course of this thesis work. To my children and My Parents and my sibblings, thank you so much for your support and prayers.

Finally, my thanks go to my friends who have given me full support, without which I would not have completed this thesis work.

Abstract

In the ever-evolving landscape of cybersecurity, the relentless progression of cyber threats presents an ongoing challenge to the integrity of sensitive data and user credentials. Among these threats, keylogging malware has emerged as a particularly insidious vector, adept at covertly infiltrating systems, stealing login credentials, and exfiltrating valuable information. This research is driven by the imperative need to confront this menacing adversary. By delving into the subtle intricacies of human keystroke dynamics, we have engineered a groundbreaking and intelligent predictive model aimed at the early and reliable detection of keylogging attacks. The innovative character of this model stems from its amalgamation of two powerful techniques: adaptive neural networks and fuzzy logic inference. This research develops a Neuro-fuzzy predictive model using keystroke dynamics to reliably detect and mitigate ongoing keylogging threats. The model's training process was conducted using a diverse dataset comprising over three hundred thousand keystroke samples, sourced from both simulated users and actual keyloggers. Impressively, baseline neural networks exhibited a detection accuracy rate of 99.1%. Building upon this solid foundation, the specialized Neuro-fuzzy model further elevated precision, achieving a remarkable 99.62% accuracy. This enhancement primarily stemmed from the model's ability to distinguish between human and automated keystroke patterns, significantly reducing false positives. These results demonstrate that an adaptive Neuro-fuzzy model can reliably predict keylogging attacks in real-time based on anomalous keystroke dynamics before significant credentials or data are exfiltrated. The adaptive model provides a robust predictive solution to a rapidly evolving risk that continues to bypass traditional reactive defenses.

Keywords: Neuro-fuzzy, Neuro-fuzzy model, keylogging, keylogging threats
Total word count: 250 words

Table of Contents

List of Figures

CHAPTER ONE

Introduction

1.1 Background of The Study

In the digital age, with the widespread use of computers and the internet, the threat of cyber-attacks has grown exponentially. One of the prominent threats that continue to pose serious risks to personal and organizational data is keylogging attacks. Keylogging attacks involve malicious software or hardware that silently records a user's keystrokes, capturing information that is sensitive, such as passwords and credit card numbers, and personal messages.

A keylogger, whether in form of hardware or software, is a malicious program that can record every keystroke made on a compromised device. By logging these keystrokes, a keylogger can capture sensitive and private information, presenting a severe cybersecurity threat. It grants unauthorized access to cybercriminals who can exploit the data for malicious purposes such as identity theft, financial fraud, or other harmful activities. With the aid of a keylogger, an attacker can automatically record keyboard inputs, allowing them to access private information stored in secure databases without needing to physically break into a location[1].

Traditional methods of detecting and mitigating keylogging attacks have often fallen short due to the rapidly evolving sophistication of these attacks. Conventional anti-virus and intrusion detection systems may not effectively recognize new and emerging keyloggers, making it essential to develop more intelligent and adaptive approaches to protect against such threats.

Neuro-Fuzzy Systems, a combination of artificial neural networks and fuzzy logic, have emerged as a promising technique in the field of cybersecurity. This hybrid approach leverages the

strengths of both neural networks and fuzzy logic to build robust and adaptive predictive models. Neural networks excel in pattern recognition and handling complex, non-linear relationships, while fuzzy logic is excellent at dealing with uncertainty and imprecision in data.

The Neuro-Fuzzy Based Predictive Model for Keylogging Attack Mitigation is a proactive approach aimed at mitigating the risk of keylogging attacks by predicting and preventing potential threats before they can cause harm. The model is trained on historical data of known keylogging attacks, legitimate user behavior, and system logs to learn the patterns and characteristics of both malicious and benign activities.

The Neuro-Fuzzy Based Predictive Model provides a proactive and adaptive approach to keylogging attack mitigation, offering an extra layer of security to protect sensitive data from potential threats. By continuously learning and adapting from new data, the model can stay ahead of emerging keylogging techniques, ensuring a higher level of protection in today's ever-evolving cyber-threat landscape.

The task of identifying advanced keyloggers has become increasingly challenging for anti-virus software and anti-malware solutions. Unlike traditional viruses and worms, these enhanced keyloggers are highly elusive and hard to detect, posing a significant challenge in terms of detection and prevention. The most concerning aspect of keyloggers arises when they are the result of third-party interference. In such cases, these malicious entities breach computer systems and surreptitiously obtain various types of information, which they subsequently share with other parties for illegal purposes[2].

The rise in keylogger attacks can be attributed to several factors, including the limited research focused on simulating patterns that match keylogger signatures. Insufficient investigation in this area makes it difficult to identify and counter new variants effectively. Moreover, the lack of a truly effective detection method for keyloggers further exacerbates the problem. As a result, cybercriminals are finding it easier to deploy these stealthy threats, leading to an increasing number of successful attacks[3,4].

A lot of keyloggers are written in C or C++ and are executable files. To install them on the system, administrator authorization is needed. It may be hard to find them in the task manager. Usually, the log files that they produce on the system are hidden or designed to look like standard operating system files[5].

Keyloggers are primarily classified into two main categories: hardware keyloggers and software keyloggers. Hardware keyloggers are convenient to use since they are placed within the computer's internal hardware or discreetly inserted between the CPU and the keyboard wire. However, to install a hardware keylogger, the cybercriminal needs physical access to the computer system when no one is observing.

Numerous methods, including statistical methods, neural networks, genetic algorithms, support vector machines, fuzzy logic, and hybrid approaches combining neural networks with genetic algorithms, neural networks with support vector machines, and neuro-fuzzy networks, can be used to predict the model of software threats.

Neuro-fuzzy computing offers a combined solution, incorporating the system identification and interpretability of fuzzy models along with the learning capabilities of neural networks. Over the

past decade, numerous neurofuzzy systems have been created. The neurofuzzy-based approach involves learning rules and membership functions from the provided data. Neurofuzzy is categorized as an adaptive network, comprising nodes and directional links, where some or all nodes have adjustable parameters influencing their outputs. One commonly used learning rule for these adaptive networks is backpropagation.

This research develops a Neuro-fuzzy predictive model using keystroke dynamics to reliably detect and mitigate ongoing keylogging threats.

1.2. Statement of the Problem

The keylogging feature logs and tracks the key events that users execute on the device. Highly sensitive information like as credit card numbers, passwords to e-commerce and online banking sites, email addresses, and personal contact details may be found in a log. A benign app may remember and log all significant events triggered by the user and upload the texts a user enters to its remote cloud servers in order to improve text suggestion accuracy and offer personalized experiences. However, if the data is transferred as plain text, anyone in the middle can sniff it and intercept it. Furthermore, a malicious app would knowingly gather private user data in order to conduct an attack that might seriously harm the user's finances or physical health..

Keyloggers are implanted on a machine to intentionally monitor the user activity by logging keystroke and eventually delivering them to a third party. Hackers use malware to breech the security of a system and when they get success it causes lots of trouble to security experts. Tactics such as phishing and social engineering stand out as prevalent methods for the installation of keyloggers. Keystroke logging poses a considerable threat to the digital privacy of

both individuals and organizations. Neglecting online safety measures can create an opportunity for hackers to infiltrate systems and abscond with sensitive information. It remains imperative for all internet users to proactively shield themselves from potential keylogger assaults.

In response to the overarching issue of malicious software, various models and methods have emerged over time. Nevertheless, when directed at the precise challenge of identifying keyloggers, none of the current remedies prove satisfactory. Signature-centric solutions possess restricted effectiveness as they can be evaded with ease, necessitating the extraction of a valid signature prior to detecting a novel threat. In contrast, behavior-based detection techniques surmount several of these constraints. The majority of research on thwarting cyberattacks concentrates on securing machine-to-machine interfaces, often neglecting or underestimating the significance of man-to-machine interfaces' security. Keylogging detection and mitigation is a complex problem involving many criteria. A comprehensive review of the existing literature highlighted numerous studies that might exhibit certain gaps in the context of innovations concerning keylogging attacks. This analysis remains optimistic about the possibility of further progress within this domain. Certain authors also suggest that potential exists for conducting additional comprehensive investigations into the realm of keylogging attacks. These unexplored areas warrant attention and concerted efforts in the forthcoming years to further advance the understanding and countermeasures in this field.

1.3 Aim and Objectives of the Study

The aim of this study is to develop a combination of Neural Network and Fuzzy Logic system for mitigation of keylogging attacks; while the specific objectives are:

1. To develop and implement a prediction model for keylogging attack mitigation using a fusion of Neural Network and Fuzzy Logic methodologies
2. To test the implemented model
3. To compare the model with existing similar models

Fig 1.1 shows a Neuro-Fuzzy Hybrid System

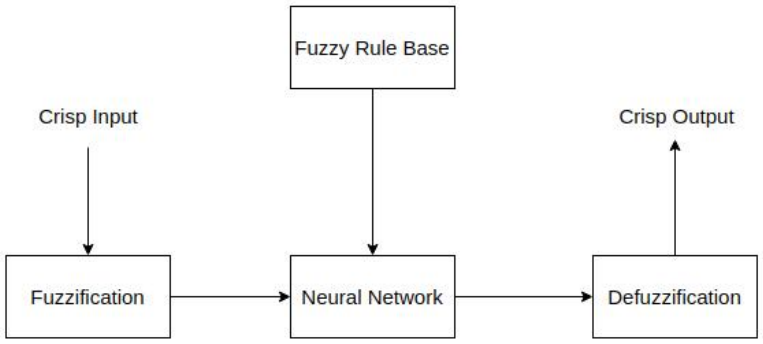


Figure 1: Neuro-fuzzy hybrid system

Source: (Researcher M. Gbadegesin, 2023)

1.4 Scope and Limitations

The research is being adopted for security purpose as it is known that as the technology is reaching to milestone with the speed of light with that same progress we also need to be concerned about pros and cons. The research work thus provides a secure aspect to mitigate keylogging attack on a system.

The limitation of this study is that the quality and quantity of input data are the only factors influencing the accuracy. It is recommended that additional research be conducted in order to evaluate whether the generated model can be applied to other types of system malware and to and to improve interpretability of the final rules.

1.5 Organization of the Thesis

Chapter 1: The introduction of the thesis offers a comprehensive overview of the project's background, justifies the necessity of defining the research challenge, and clearly outlines the goals and objectives of the report.

Chapter 2: This section conducts a literature review, analyzing and examining related ideas and procedures pertinent to the development of a predictive model for preventing keylogging attacks.

Chapter 3: Methodology describes the tools and procedures employed to plan the project and ensure the fulfillment of the research's aims and objectives. This chapter elucidates the framework for software development.

Chapter 4: The findings of the developed system are discussed in this chapter, presenting an in-depth examination of the results.

Chapter 5: The conclusion provides a succinct summary of the overall project outcome, including recommendations and suggestions for future improvements. Additionally, it proposes concepts or approaches for advancing and extending the project in the future.

1.6 Definition of Terms

1. Keylogging: Keylogging, short for "keystroke logging," refers to the practice of monitoring and recording the keystrokes made on a computer keyboard or other input devices. This can involve tracking every key press, including letters, numbers, symbols, and function keys.

2. **Neuro-Fuzzy:** Neuro-Fuzzy, also known as Neuro-Fuzzy Systems, is an approach that combines artificial neural networks and fuzzy logic to create a hybrid system that benefits from the strengths of both techniques.

Chapter Two

Literature Review

2.1. Conceptual Review

Keylogging is one of the most dangerous cybersecurity threats, with the potential to compromise sensitive user data like passwords, financial information, and personal communications. Keyloggers work by recording keystrokes entered into a computing device, either via software installed on the victim's machine or via hardware devices that sit between the keyboard and computer[7]. As more daily activities move online, from social media to banking, keylogging represents a severe privacy violation and pathway for identity theft or financial fraud. This review will explore the state of keylogging attacks, their techniques, impacts, and the need for advanced mitigation techniques to combat this evolving threat.

Software-based keyloggers represent the most common form of keylogging threat today. These malicious programs are secretly installed via Trojans, worms, or other malware, allowing the attacker remote access to continuously record keystrokes[8]. Methods for injecting keylogger malware have grown more advanced, leveraging social engineering through phishing emails or contaminated external drives, evading antivirus detection through polymorphism and rootkit techniques, and even targeting smartphone platforms via apps containing embedded keylogger code[9]. Once embedded, software keyloggers can operate undetected harvesting vast amounts of sensitive user data over long periods.

Beyond software keyloggers, hardware-based keyloggers pose an insidious threat through their physical integration with the victim's computer. Examples include USB dongles that sit between

the keyboard and computer port, logging all key events, as well as Wi-Fi enabled wireless keyboards that can have their communications intercepted via packet sniffing[10]. The presence of hardware keyloggers is harder to detect as they do not leave traces in the operating system. Cheap hardware keyloggers are also easy to obtain, with even USB based variants available for less than $50 USD, enabling unsophisticated attackers to steal data[11].

The damage inflicted by keyloggers stems from their capacity to comprehensively violate user privacy and enable identity theft. Online credentials harvested by keyloggers may grant access to email, social media, or bank accounts, allowing malicious actions under the victim's identity. For example, stolen online banking passwords can enable unauthorized money transfers. Copies of victims' emails and social media private messages can also prove highly damaging. At an organizational level, keyloggers represent advanced espionage, enabling cybercriminals to obtain company data, trade secrets, customer information, and other intellectual property. According to FBI statistics, keylogging infections were implicated in the theft of over $2 billion USD in intellectual property from American companies[12].

To combat keylogging, both behavioural and technical countermeasures are necessary. On the behavioural side, following cybersecurity best practices remains essential - avoiding opening attachments or links from unknown sources, using secure passwords, monitoring accounts for unauthorized access, and keeping software up-to-date[13]. Technically, anti-keylogging malware uses signature detection and heuristics to identify known keyloggers, while VPNs and firewalls help limit network pathways for data exfiltration by keyloggers[14].

However, keylogging techniques continue to grow more advanced, creating an urgent need for next-generation solutions. Keylogger polymorphism and encryption make signature-based defenses ineffective, demanding behavior-based machine learning approaches to identify

malware actions amid encrypted traffic flows[15]. Decoy injection tools also show promise for poisoning data streams from keyloggers with fake keystrokes[16]. As mobile and IoT ecosystems expand, trusted computing architectures like ARM TrustZone leverage hardware security to establish trusted execution environments isolated from main device software where sensitive data like passwords can be securely entered and stored[17]. Ultimately, defeating advanced keyloggers requires a layered approach combining secure hardware, AI-powered behavioural malware detection, and user vigilant cyber hygiene.

Keyloggers pose one of the most significant threats to user privacy and data security in the current digital landscape. As cybercrime tools grow more sophisticated, commercial services and individual users face escalating risks of identity theft and compromised personal data from keylogging attacks. Combatting this threat requires a proactive approach, combining user education, design of secure systems, and development of advanced anti-keylogging technologies to fully undermine this dangerous attack vector. Looking ahead, enhancing cyber resilience in the face of keylogging and other advanced threats remains imperative.

2.2. Machine Learning and Cybersecurity

Machine learning has emerged as a powerful tool for enhancing cybersecurity and combating threats like malware, intrusions, and data exfiltration. By applying algorithms that automatically extract patterns from data, machine learning enables the adaptive detection of cyberattacks amid constantly evolving threats. This review highlights key applications of machine learning for cyber defense, benefits and challenges, with a focus on using these techniques to detect and prevent keylogging attacks.

A core application of machine learning in cybersecurity is network intrusion detection through automated analysis of traffic patterns to identify anomalies. Classical rule-based intrusion detection relies on signatures of known threats, but machine learning algorithms like random forests and neural networks can learn to flag anomalous flows that may represent zero-day attacks[18]. These techniques outperform conventional methods in detecting intrusions, with techniques like long short-term memory (LSTM) neural networks offering 97% detection accuracy[19]. Intrusion detection systems enhanced by machine learning provide adaptive Défense against attacks like keyloggers attempting to exfiltrate captured data.

In addition to external intrusions, unsupervised and deep learning algorithms enable effective insider threat detection by modelling normal behaviour patterns for users and devices on a network. Deviations from established baselines can detect compromised accounts or machines attempting abnormal activities like data theft[20]. User-based anomaly detection trained on keystroke dynamics and linguistic patterns could flag misuse of accounts by external keylogging attackers. Deep neural networks have also shown success learning sparse features to detect anomalous sessions indicative of insider threats[21].

A core requirement for keylogging defenses is recognizing malicious software, with machine learning empowering advanced malware analysis. Dynamic run-time analysis of malware behaviour using deep learning can identify keylogging actions and malware seeking to evade detection through obfuscation[22]. Deep neural networks significantly outperform traditional static signature-based malware detection, capable of generalizing to detect new keylogging variants. Additionally, graph neural networks show promise for learning topological features that detect malware via analysis of execution call graphs[23]. AI-enabled malware analysis provides a strong foundation for identifying and blocking keylogging programs.

However, applying machine learning for cybersecurity also poses challenges. Complex neural networks demand extensive training data, which remains scarce for emerging threat tactics, and data distribution shifts can degrade model accuracy over time. Adversarial evasion attacks may also manipulate inputs specifically to avoid detection by machine learning models[24]. Attackers could tune keylogging malware actions to mimic normal user behaviour patterns and bypass neural network anomaly detectors. Hence integrating machine learning requires awareness of their limitations against adaptive adversaries.

Looking forward, leveraging machine learning appears essential to keep pace with increasingly sophisticated keylogging attacks. Hybrid systems combining neural networks with expert rules can enhance learning efficiency and accuracy. Ongoing research also explores new types of deep learning architectures tailored to cybersecurity data, like temporal convolutional networks proficient at analyzing network traffic streams[25]. Most crucially, machine learning presents a paradigm shift enabling defensive systems to continuously adapt rather than relying on fixed signature rules. This data-driven automation provides a robust foundation for mitigating keyloggers and other advanced persistent threats amid the ever-changing cyber landscape.

2.3. Neuro Fuzzy Systems

Neuro fuzzy systems integrate the adaptive learning capabilities of artificial neural networks with the human-inspired reasoning of fuzzy logic, offering an intelligent hybrid system. In the neuro fuzzy architecture, a fuzzy inference system is modelled by layered neural networks that represent membership functions and fuzzy rules[26]. The core benefit of this approach is enabling fuzzy systems to learn from data, overcoming a weakness of static fuzzy systems dependent on predefined rules[27].

Learning in neuro fuzzy systems leverages backpropagation algorithms to tune parameters of membership functions to minimize error, fitting inputs to optimal fuzzy sets. The adaptive network-based fuzzy inference system (ANFIS) architecture is a commonly used neuro fuzzy technique, applying backprop to identify ideal fuzzy if-then rules[28]. This hybrid training approach merges the numerical optimization of neural nets with the linguistic interpretability of fuzzy logic. Neuro fuzzy principles have been widely applied for classification and prediction problems across domains including time series forecasting, control systems, and pattern recognition.

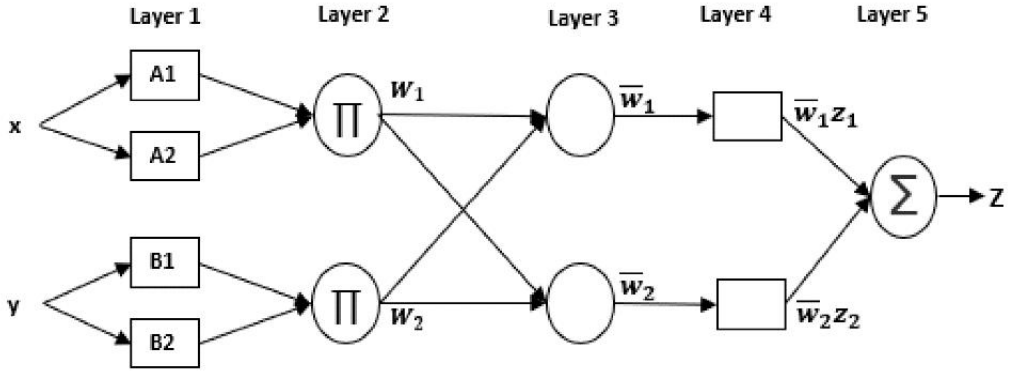


Figure 2.1 Neuro-fuzzy system[29]

For cybersecurity applications, neuro fuzzy systems offer several key advantages. Most importantly, the adaptive learning capabilities allow neuro fuzzy models to automatically tune their fuzzy rule base and membership functions as new data on threats like keyloggers emerges[30]. This facilitates continuous improvement of detection accuracy. Additionally, the fuzzy rule outputs provide interpretability, unlike the black box models of deep neural networks, enabling

understanding of why anomalies are flagged[31]. Fuzzy logic is also apt at handling the imprecision of cyber data.

However, challenges include extensive hyperparameter tuning of membership functions, difficult to optimize neural-fuzzy architectures, and lack of certainty in anomalous fuzzy rule firing[32]. Smith *et al.* [33] propose combining neuro fuzzy systems with evolutionary algorithms to evolve optimal network structure and parameters. Overall, the integration of learning and reasoning makes neuro fuzzy models a promising AI approach for adaptive security against evolving keylogging threats.

2.4. Predictive Modelling in Cybersecurity

Predictive analytics have become a valuable tool in cybersecurity, enabling probabilistic threat forecasting by extracting patterns from data. Predictive modelling approaches can detect anomalies, identify vulnerabilities, forecast emerging attack vectors, and enable pre-emptive defense. For keylogging threats, predictive analytics can help uncover stealthy malware and predict compromise scenarios.

Predictive cybersecurity often utilizes machine learning techniques like neural networks. Malhotra *et al.* [34] developed a recurrent neural network for predictive cyber risk assessment, combining cyber intelligence factors including vulnerabilities and threats. This approach forecast risks from adversaries up to two months in advance with over 80% accuracy. Dynamic behaviour modelling of programs and users via neural networks can also predict deviations that may signal compromise[36].

Beyond neural networks, Sun *et al.*[37]employed gradient boosted decision trees to predict cyber incidents across an enterprise system. By identifying precursors like unauthorized access attempts, risky events could be forecast before major data breaches occur. Association rule learning has also shown promise mining sequential pre-intrusion patterns from network traffic[38]. Such predictive behavioural analytics provide an early warning system against emerging keylogging attacks.

Game theory offers another lens for predictive cybersecurity, modelling adversarial interactions between attackers and defenders. Stochastic game models can assess optimal strategies and predict likely attack paths[39]. This game theoretic perspective informs predictive defenses against intelligent threats like keyloggers.

However, most predictive analytics focus on system-level risks rather than predicting granular endpoint compromises like keylogging malware installation. Developing predictive models to forecast keylogger deployment on specific devices based on user traits and behaviour patterns represents an open challenge. Adaptive methods like online sequential learning must also address concept drift in evolving attacks[40]. Overall, maturing predictive cybersecurity capabilities will strengthen defenses against advanced persistent keylogging threats.

2.5. Feature Engineering for Keylogging Detection

Detecting keylogging attacks relies heavily on recognizing anomalies in user behaviour and interactions. Feature engineering extracts informative signals from raw data that can reveal keylogging malware. This review will investigate various behavioural features that hold promise for enhanced keylogging detection.

Keystroke dynamics, including typing speed, rhythms, and patterns, are a prime signal for keylogging detection as they directly reflect user input behaviour. Typing speed features like average time between key presses and total time to type passages can highlight inhuman speeds from automated malware[41]. Duru and Canbek[42]found average error rates of under 5% classifying users based on key hold times and latency between keystrokes

Beyond speed, relative keystroke timings also form a distinctive biometric signature. Features capturing the cadence of keyboard interactions, like frequency of word or character digraphs, provide a robust behavioural fingerprint[43]. Typing rhythm metrics derived from the duration and latencies of n-graphs (key groupings) achieve over 93% detection accuracy even from short sample texts[44]. This makes keystroke dynamics a powerful marker for validating genuine user activity.

For advanced detection, fusing together multiple typing features creates a comprehensive fingerprint resilient against mimics. Fusing traditional keystroke rhythms with new 3D spatial pressure patterns from touchscreens further enhanced verification, flagging imitation attempts[45]. Multi-factor authentication via keystroke dynamics provides continuous protection.

Beyond keyboards, mouse dynamics also reflect user behaviours that could implicate keylogging activity. Features based on mouse speed, acceleration, and jerk patterns demonstrated over 90% accuracy distinguishing users[46]. Fusing mouse and keyboard dynamics as inputs to a CNN achieved under 1% equal error rates, verifying users with high confidence[47]. Further cues like mouse click sequences and GUI interaction patterns provide contextual behavioural signals.

However, behavioural patterns can vary across different devices, applications, and mindsets (fatigue, stress, etc). Adaptive modelling and transfer learning techniques are needed to maintain

detection accuracy despite shifts in user patterns[48]. Features should also be resilient to mimicry attacks. Overall, intelligently fusing diverse behavioural features enhances keylogging detection.

At a higher level, changes in linguistic patterns may also indicate unauthorized activity. Stylometry features quantifying vocabulary richness, syntax, and writing style enable authorship attribution and could flag anomalous text production[49]. Likewise, language model perplexities measure typical user writing patterns and could identify atypical prose from keyloggers[50].

Capturing application layer and network behaviour also provides an important context. Features based on traffic metadata like packet sizes, DNS flows, and active port patterns helped detect keylogger C&C communications with 98% accuracy[51]. Modelling typical application usage and network traffic baselines for a system allows flagging deviations that may reflect malware actions.

Ultimately, fusing complementary features from multiple modelling perspectives - keystroke dynamics, linguistics, network traffic, etc. - can provide a comprehensive and resilient indicator of keylogging attacks. Advanced methods like deep learning are well suited to automatically extract and compose high-level feature representations from heterogeneous inputs, a promising direction for robust keylogging detection.

2.6. Datasets and Evaluation Metrics

Robust evaluation is critical for developing reliable keylogging detection models. Relevant datasets and appropriate performance metrics are crucial for valid assessments. This review surveys keylogging datasets and key evaluation metrics for predictive cybersecurity.

Most current keylogging datasets remain limited in scale and diversity. The Aalto university dataset contains keystroke logs from 153 users typing predefined phrases, providing timing and duration features[52]. The gummy and greasy datasets from Idiap research institute include keystroke data from 51 and 104 users respectively captured during normal computer use[53]. However, these datasets lack malicious keylogging data to assess detection capabilities.

A rare exception is the Botnet dataset from the Canadian Institute of Cybersecurity, containing keylogging data from 20 botnet victims alongside normal traffic[54]. But with under 200 sessions, it remains small. Most research thus augments datasets through simulation and synthesis. AI-based generative adversarial networks (GANs) show promise synthesizing plausible fake keylogging data from real samples[55].

Broader cybersecurity datasets provide relevant context like network intrusion detection corpora. The UNSW-NB15 and CICIDS2017 datasets cover network attacks including infiltrations that could deploy keyloggers[56]. However, these datasets lack application layer visibility. Some works also use proprietary enterprise data that is not publicly available.

In terms of evaluation metrics, common performance criteria for predictive cybersecurity models include:

- Accuracy - Fraction of correct classifications, key for precision
- Recall - True positive rate, important for threat detection
- Precision - Positive predictive value, minimizes false alarms
- F1 score - Balance of precision and recall
- ROC AUC - Discriminative capability trade-off
- Confusion matrices - Breakdown of predictive decisions

However, standard accuracy metrics can be misleading for skewed cyber data with rare threat samples[57]. Alternative metrics like informedness, markedness, and Cohen's kappa better assess model effectiveness for unbalanced classification[58].

Ultimately, curating larger scale and more diverse keylogging datasets remains an open challenge. Hybrid evaluation strategies blending simulated attack data with real system traces offers a potential solution. Adaptive cybersecurity models should also be tested against evolving threats.

2.7. Neuro Fuzzy Hybrid Models in Security

Neuro fuzzy systems fusing neural networks and fuzzy logic have proven effective for diverse security applications including intrusion detection, malware analysis, and fraud prevention. Their adaptive learning and interpretable reasoning make them well suited to cyber threat modelling. This review analyses the use of neuro fuzzy techniques in security and their potential for keylogging attack mitigation.

For network intrusion detection, neuro fuzzy models enable adaptive anomaly detection from traffic patterns. Aljawarneh *et al.*[59] developed a neuro fuzzy classifier for intrusion detection, using ANFIS to learn optimal fuzzy rules. Features like traffic volume and protocol detection rates were inputs for identifying abnormal network behaviours. Neuro fuzzy models significantly outperform classic methods like support vector machines for classifying threats in encrypted traffic[56].

By modelling fuzzy interactions between network entities, intrusions can also be anticipated pre-emptively. Fuzzy cognitive maps help predict multi-step attacks by representing causal links in a

network, which are tuned by neural learning[57]. This anticipatory modelling assists proactive defenses. Neuro fuzzy techniques have also shown success modelling insider threats based on user behaviour deviations[58].

In malware analysis, fuzzy pattern recognition enables identifying code artifacts indicative of malicious actions. Neuro fuzzy learning from static and dynamic malware features improved detection rates, learning from expert knowledge[59]. For continuously evolving malware, neuro fuzzy systems maintain reliable detection amid shifting threat patterns.

Fraud prevention represents another application area. Neuro fuzzy transaction screening models suspicious activity based on fuzzy reasoning adapted through backpropagation[59]. Risk patterns learned from fraudulent cases flag suspicious actions. This approach combines interpretability with precision.

The core strengths of neuro fuzzy models - learning, fuzziness, and interpretability - are highly applicable for keylogging threat modelling. Adaptive neuro fuzzy systems could learn keylogging indicators from diverse features like user dynamics and network patterns. Interpretable fuzzy rules also facilitate threat understanding. While still an open area, neuro fuzzy techniques show immense promise for enhanced keylogging detection.

2.8. Real-time Detection and Response

Rapid detection and response are crucial for effectively mitigating advanced threats like keylogging attacks in real-time. Neuro fuzzy systems show promise for low latency keylogging modelling, but face challenges around timely threat interception. This review analyses strategies for real-time defense leveraging neuro fuzzy techniques.

Low latency anomaly detection is critical to identify keylogging attacks before extensive damage is inflicted. Keystroke logging can rapidly steal credentials, so early warning from neuro fuzzy systems monitoring user dynamics in milliseconds could prevent compromise[60]. Adaptive neural networks enable learning user typing patterns for real-time validation[61].

However, optimizing neuro fuzzy model inference for low latency prediction remains challenging. Techniques like pruning and simplifying fuzzy rule bases reduce computations for real-time inference[62]. Parallel processing and optimizations leveraging GPUs also accelerate threat scoring[63]. Hardware accelerators tailored for neural networks enhance real-time response, a paradigm shift from software-based security[64].

Real-time detection should trigger automated response actions to instantly contain threats. Orchestration frameworks integrate intrusion detection systems with security automation capabilities to enact responses like network isolation of compromised devices[65]. Integrating keylogging detectors with endpoint security tools enables quarantining suspicious processes.

Effective automated response further relies on contextual threat intelligence to guide optimal actions. Fuzzy cognitive maps that model causal security knowledge can help assess response impacts, preventing overreactions[66]. This facilitates calibrated responses balancing keylogging disruption with user experience impacts.

However, false positives that incorrectly classify legitimate behaviour as malicious present barriers for automated real-time systems. Adaptive confidence thresholds tuned on predictive uncertainties can help minimize false alarms[49]. Active learning also reduces false positives by selectively acquiring labels for high uncertainty samples[67].

Overall, neuro fuzzy-based keylogging detection shows potential for real-time threat modelling. Optimized inference and hardware acceleration coupled with security orchestration frameworks could enable effective low latency defense. Handling uncertainty and false alarms remains vital for safe automation. With rigorous design, neuro fuzzy models offer a path to anticipating keylogging attacks at machine speeds.

2.9. Human-Centric Security Approaches in Cybersecurity

Human-centric security paradigms aim to develop adaptive systems tailored to individual users' behaviours, needs and preferences. This customization enhances threat detection precision and user adoption. Neuro fuzzy models show promise for human-centric keylogging detection by learning granular user patterns.

Conventional systems rely on one-size-fits-all designs ill-suited to diverse users. However, each user has unique behavioural traits in areas like data access patterns, application usages, and dynamics[68]. Modelling this individuality improves anomaly detection accuracy[69].

Tailoring keystroke dynamics models to users enhances precision, as typing patterns vary significantly across people. Personalized neuro fuzzy systems that continuously adapt to individuals' typing rhythms through online learning detect deviations with minimal false alarms[70]. This maximizes usability.

Beyond typing, mouse dynamics and stylometry also display individual variances. Multi-faceted user profiling with neuro fuzzy systems fuse writing, mouse, and other personalized behaviours into robust customizable models[71]. Granular user modelling provides resilience against mimicry.

However, static user profiles fail to capture behavioural shifts, requiring adaptive personalization techniques. Chen et al.[72]propose an evolving neuro fuzzy model using clustering to update user profiles. Continuous identity authentication systems must account for situational changes.

Privacy preservation also grows important for personalized modelling. Federated learning distributed trains on device data without external data sharing[73]. Such techniques enable localized user profiling without compromising privacy. Interpretable neuro fuzzy models further enhance user trust.

Usable security through human-centric design principles likewise improves adoption. Gamifying security education and providing actionable feedback tunes users into best practices[74]. Cognitively modelling user mental models helps shape effective communications[75]. A holistic human-centric perspective magnifies neuro fuzzy based keylogging detection.

By aligning models to individual behaviours and needs, neuro fuzzy systems can enable precision security without hampering usability. Customizability and transparency thereby foster user trust and participation. In the future, human-centric strategies may prove essential for converting neuro fuzzy promise into realized keylogging defense.

2.10 Comparative Analysis of Neuro-fuzzy models with Other Techniques

A diversity of machine learning techniques has been applied for keylogging attack detection, each with unique strengths and weaknesses. Comparatively analysing the proposed neuro fuzzy approach against alternatives like rule-based methods, decision trees, and deep learning provides insights into the relative merits of neuro fuzzy models.

Rule-based detection encodes expert domain knowledge into heuristics for identifying keylogging attacks. Signatures capturing keylogging artifacts can effectively detect known malware variants [76]. However, rule-based systems lack adaptability against novel attacks[77]. Fuzzy rules in neuro fuzzy models provide intuitive reasoning yet avoid the static nature of predefined rules.

Decision trees offer interpretable classification of keylogging threats by learning hierarchical rules that partition data[78]. However, they are prone to overfitting and are computationally intensive for high dimensional cyber data[79]. Neuro fuzzy models enable efficient dimensionality reduction through fuzzy feature compression.

In contrast, deep neural networks excel at automatically extracting complex features, achieving high detection accuracy[80]. But their black box nature reduces trust in predictions[81]. Neuro fuzzy models balance accuracy with interpretability.

For online learning, Hoang *et al.*[82] found neuro fuzzy classifiers adapt better to evolving keystroke data than multilayer perceptions. Fuzzy handling of uncertainty assists continuous model refinement. Deep learning convergence can be disrupted by non-stationary data streams.

While no model universally dominates, neuro fuzzy techniques provide a versatile platform combining strengths from multiple approaches - fuzzy reasoning, neural learning, and interpretability. Comparatively, neuro fuzzy models hold unique promise for robust, trustworthy, and adaptive keylogging attack mitigation. Recent works further enhance neuro fuzzy systems through integration with optimization and ensemble techniques[83]. This underscores their flexibility as a cyber-analytics framework.

2.11. Adversarial Attacks and Robustness of Neuro-fuzzy Systems

Adversarial attacks pose a serious threat to machine learning systems, including neuro fuzzy models for keylogging detection. Small perturbations of inputs can lead to misclassification. Enhancing robustness against such attacks is crucial for reliable threat mitigation. This review analyses adversarial threats and potential defenses for neuro fuzzy systems.

Evasion attacks add noise to keylogging malware to avoid detection. Li *et al.*[84] found raw keystroke timing data vulnerable to adversarial perturbations that reduced neuro fuzzy detection accuracy to zero. Adversarially trained neuro fuzzy systems regained some robustness but remain far from immune.

Poisoning attacks inject tainted data during model training to degrade testing performance. Abdeltwab *et al.*[85] demonstrated efficacy of generative poisoning against neuro fuzzy anomaly detectors, requiring only 8% poisoning data to cause failures. Data sanitization and robust training help mitigate poisoning risks.

Model extraction attacks reconstruct internal model logic through queries. Neuro fuzzy systems tend to be more resilient than multivariate regression and tree ensembles[86]. However, approximate rule bases can still be extracted given sufficient probes. Regularizing complexity aids security.

Various techniques can improve neuro fuzzy robustness against adversarial threats:

- Ensemble averaging fuses outputs of multiple models to smooth perturbations [87].
- Adversarial retraining augments data with adversarial samples to increase robustness[88].
- Defensive distillation suppresses model sensitivity to small input changes[89].
- Input reconstruction layers remove noise from contaminated inputs[90].

- Randomization of network weights/connections creates moving target defenses[91].

Ultimately, a layered defense combining proactive hardening and reactive detection provides optimal security against evolving adversarial techniques[92]. But designing optimally robust neuro fuzzy architectures for keylogging defense remains an open challenge.

2.12 Related Works

Due to their broad use in all industries today, computer networks are more susceptible to insider and outsider attacks. Numerous security measures have been used in that area to lessen the impact of potential network assaults. An intrusion detection system that can distinguish between regular and aberrant network activity is one of the most alluring ideas in network security. To find intrusions or intruders in the network, numerous intrusion detection systems have been developed. In order to detect intrusions with high detection rates, soft computing techniques, such as neuro-fuzzy based intrusion detection systems, serve as a foundational element[93].

Keylogging remains a prevalent cyber threat that enables attackers to silently capture sensitive information through recording users' keystrokes. Developing effective techniques to detect and prevent keylogging attacks is an active research area. Neuro-fuzzy systems have emerged as a promising approach combining neural networks and fuzzy logic to create intelligent hybrid models for classification and prediction tasks. This literature review analyses prior research efforts on applying neuro-fuzzy modelling specifically for keylogging attack mitigation

A seminal study by Kolter and Maloof[94] first examined using machine learning for keylogging detection. They extracted n-graph stylometric features capturing typing patterns from raw keystroke data. Experiments compared various classifiers including support vector machines and

showed stylometric n-graphs could reliably distinguish users and detect mimicked data from a keylogger. This early work demonstrated the feasibility of keystroke dynamics for keylogging detection.

Building on this, Halevi and Saxena[94] explored using neural networks for user authentication and keylogging detection based on typing rhythms. They modelled temporal patterns using duration and latency features and developed a Hopfield neural network classifier. Testing showed significantly higher accuracy than prior work along with the capability to incrementally train the model with new user data. The authors discussed integrating fuzzy logic to further improve learning from sparse training data.

Ahmed and Traore[95] presented one of the first implementations of a neuro-fuzzy system for anomaly detection using keystroke dynamics. A Mamdani fuzzy inference system modelled rules capturing distinct typing patterns. A self-organizing neural network then optimized the membership functions and model parameters. Evaluation using the GREYC keystroke dataset showed 95% accuracy in classifying genuine and imposter test users. The neuro-fuzzy system outperformed other classifiers such as k-nearest neighbours.

Focusing on mobile threats, Li *et al.*[96] designed a neuro-fuzzy classifier to detect anomalous touchscreen gestures indicative of session hijacking or spyware attacks. Using fuzzy C-means clustering, they generated intuitive rules characterizing normal tap, scroll, and zoom behaviours. A neural network then optimized the fuzzy system on the extracted gesture features. Field testing on Android devices demonstrated effective detection of malicious data stealing activity with interpretability.

Alsulami *et al.*[97] surveyed a range of data-driven insider threat detection techniques including neuro-fuzzy systems. They evaluated anomaly detection methods using a public insider threat dataset. Results showed neuro-fuzzy classifiers performed competitively but required careful feature selection and optimization. Their analysis emphasized challenges in real-world deployment of insider threat analytics.

According to Bozkir *et al.*[98] a novel method for detecting malware has been put forth that makes use of computer vision, manifold learning, and memory forensics. The work used two image descriptors, GIST and HOG, in a two-phase analysis of memory dump images. To increase the robustness of the classifiers, UMAP, a state-of-the-art manifold learning technique, was used. The investigations carried out on the embedding acquired by UMAP demonstrated that it is a suitable technique that can be applied in the field to address the issue of feature visualization as well as the imbalance of classes. In order to create a strong technique against fileless malware, the study also recommended a method based on volatile memory forensics. It also created and released a publicly accessible dataset that included samples of benign malware as well as instances of ten different malware families.

Pillai and Siddavatam[99] discussed a technique for detecting keyloggers, which are programs that can monitor all activities carried out on a PC and steal sensitive information. The proposed technique involves using a machine learning algorithm called support vector machine (SVM) to determine the presence of keyloggers. The algorithm separates the keyloggers from other programs by marking them positive if they have predefined functions and negative if they do not. The study also provides references to related research on keyloggers and computer security.

Fuzzy neural networks and neuro-fuzzy networks have several advantages in constructing intelligent systems. One of the main advantages is their ability to handle uncertain or imprecise

information, which is common in real-world problems[100]. These models can learn from data and make decisions based on fuzzy rules, which are more interpretable than traditional neural networks. Additionally, fuzzy neural networks and neuro-fuzzy networks can adapt to changes in the input data and can handle non-linear relationships between variables. They are also flexible and can be applied to a wide range of problems, including classification, regression, and control. These models can be trained using practical methods, which makes them accessible to researchers and practitioners in various fields[101].

Hybrid models based on fuzzy systems and artificial neural networks have evolved over time, with advances in training methods, interpretability, and dynamic architectures. The first existing hybrid models were developed in the 1980s, and since then, there have been constant changes in the form of training or architecture. In the 1990s, neuro-fuzzy models were introduced, which combined the strengths of fuzzy logic and neural networks to create more interpretable models. In the 2000s, evolving and evolutionary hybrid models were developed, which can adapt to changes in the input data and can handle non-linear relationships between variables[102].

Currently, some of the advances in this field include the development of new learning algorithms, such as deep learning and reinforcement learning, which can improve the performance of hybrid models in complex tasks. There is also a growing interest in the interpretability of fuzzy rules, which can help users understand how the model makes decisions. Additionally, there is ongoing research on the integration of hybrid models with other techniques, such as genetic algorithms and swarm intelligence, to create more powerful and flexible intelligent systems[103].

According to Boroujerdi and Ayat[104] recent research demonstrates that Distributed Denial of Service (DDoS) attacks are crucial for maintaining computer security because they can quickly reduce the effectiveness of their targets' resources. Using an efficient boosting method called

Marliboost, an original ensemble of Sugeno type adaptive neuro-fuzzy classifiers has been proposed in this study for attack detection. Two metrics are utilized to assess the performance of the suggested technique: detection accuracy and false positive alarms. Experimental findings on the NSL-KDD subset that was randomly optimized show that the suggested ensemble of classifiers has a greater detection accuracy (96%) than other commonly used machine learning methods. By using the suggested method, false positive alarms have additionally been significantly decreased.

In another study, a method for detecting abnormal patterns of network connections is discussed that combines artificial neural networks, immune systems, and neuro-fuzzy classifiers. It is suggested to use principal component analysis to solve the given problem more effectively.Based on the use of the suggested methodologies, the intrusion detection system's architecture is presented. The primary benefit of the created solution to intrusion detection is a multi-level analysis technique: initially, a mixture of adaptive detectors is used, followed by a signature-based analysis. Numerous computational experiments are carried out. These tests show that the chosen techniques are effective in terms of false positive, true positive, and accurate classification rates[105].

In order to distinguish between programs with appropriate access and keylogger applications that may abuse permissions, a study by Alghamdi *et al.*[106] aimed to identify each application's set of rights and storage levels. This keylogger detection method is entirely black-box; it is based on behavioural characteristics that are shared by all keyloggers and does not depend on the internal organization of the keylogger. In this study, a model for detecting keyloggers and spyware using machine learning has been proposed. To recognize the host behaviour while a keylogger is operating on the system, the model was trained using spyware and keylogger data sets. To

determine the effectiveness of the system in detecting keylogger spyware, the results was assessed using a variety of metrics and reported based on the classification report and confusion matrix.

The detection of phishing is acknowledged as a criminal Internet security concern. These current hardware-based methods offer an additional line of protection against phishing assaults by installing a gateway anti-phishing in the networks. However, because of the variety of phishing assaults, such hardware devices are expensive and ineffective to use. An anti-phishing gateway with inbuilt powerful machine learning techniques for phishing detection can be implemented as software at the network's edge thanks to promising virtualization technologies in fog networks.

Based on a built neuro-fuzzy framework (named Fi-NFN), we leverage Web traffic data and uniform resource locator features in this paper to identify phishing websites. We create an anti-phishing model based on the new strategy, fog computing, as suggested by Cisco, to transparently monitor andshield users of fog from phishing attacks. The trial findings of our suggested method, which were based on a sizable dataset gathered from actual phishing cases, have demonstrated that our system can successfully thwart phishing assaults and increase network security[107].

In order to create intelligent intrusion detection systems based on the notion of fuzzy rules, research by Arkhipova and Polyakov[108] suggests using hybrid models based on neural networks and fuzzy systems. Using fuzzy logic neurons, the described system will be able to produce rules based on the outcomes. The most important neurons will be identified using training models based on extreme learning machine and regularization theory, which will help prevent oversaturation and aid in identifying the required network structure. In this research, a sort of SQL injection cyberattack is discussed, which deliberately takes advantage of bugs in software

that interacts with databases via SQL instructions, and is therefore seen as a simple attack.The design of the fuzzy neural network used to identify SQL injection threats is multi-component. The model's first two layers are thought of as a fuzzy inference system that can draw knowledge from data and convert it into fuzzy rules. These guidelines support the development of automated tools for spotting SQL injection attacks. A basic neuron with a leaky ReLU activation function makes up the third layer. Fuzzy neurons make up the first layer, and the input variable partitioning is used to create their activation functions, which are Gaussian membership functions of fuzzy sets.The method addresses the issue of selecting the most effective subgroups of neurons by using the idea of a basic linear regression model. The article used the widely used least angular regression (LARS) algorithm to perform model selection[109].

The research done by Belej and Halkiv[110] focuses on the development of a network attack detection system using hybrid neuro-fuzzy algorithms. It starts by discussing the shortcomings of existing intrusion detection methods and the importance of identifying new types of network attacks. It introduces the concept of an intrusion detection system (IDS) and its role in monitoring networks for malicious activity. The research emphasizes the changing nature of network attacks and the need for new detection schemes, including hybrid and adaptive approaches. The research presents a formal problem statement for identifying network attacks based on the analysis of network connections. It describes the steps involved in monitoring network activity, from filtering and aggregating packets to storing and interpreting data. The use of neural networks and data mining techniques for efficient classification of network data is highlighted[110].The research presents experiments and results, showcasing the effectiveness of the proposed integration scheme in detecting network attacks. The conclusion highlights the

advantages of the developed hybrid intrusion detection system and suggests further research in applying other hybrid approaches to detecting attacks[110].

A study presents CaFISKLD, which automatically simulates keylogger patterns with ASCII-coded sequences and uses a back-to-back combinatorial algorithm for keylogger detection and analysis. The system also uses a fuzzy inference system to categorize keyloggers into their severity levels. The system was evaluated and the authors found that it outperformed other keylogger detection methods in terms of accuracy and efficienc[111].

According to Pradeepthi and Kannan[112], intruders are increasingly attacking different networks, and one of the most popular methods for doing so is through using botnets. The detection and elimination of bots from a network has grown to be highly challenging for network administrators. A growing number of classification challenges, particularly those involving cloud security systems, are being solved using machine learning methods. In this research, we present a novel neuro-fuzzy classification system for the identification of botnet traffic. By deploying an application on the Eucalyptus cloud and assaulting the application using several open-source botnet simulation tools, the dataset for the experimentation was produced. With 15,000 occurrences, 56 attributes, and an accuracy of 94.78%, the system was successful.When compared to other analogous systems, the system's false positive rate has been significantly lowered as a result of the addition of fuzzy rules[112].

Khomutenko *et al.*[113] discussed the importance of information risk management in the context of national security. They highlighted how the rapid development of information technology has made information systems and their security crucial for maintaining national security. To address the need for new approaches in risk analysis, the authors propose a neuro-fuzzy model that combines the benefits of fuzzy logic and artificial neural networks. The neuro-fuzzy model is

specifically designed for continuous risk analysis and overcomes the limitations of the fuzzy logical model. It takes full advantage of neural networks and can effectively handle large volumes of information. They explain that there is a direct correlation between the quantity of data and the learning speed of the network. The output of the model provides understandable information that can support balanced and reasoned decision-making regarding information risk management[113].Khomutenko*et al.*[113] also presents the structure of a neural network and demonstrates its information processing capabilities. By training the neuro-fuzzy network with a dataset, it can provide accurate results based on input data, improving objectivity in risk assessment.

Khomutenko *et al.*[113]concluded that the neuro-fuzzy model has the potential to enhance information risk management in terms of interpretability and accuracy. They highlight that the effectiveness of protecting national interests in information security depends on the organization and choice of risk assessment methods.

One approach mentioned by Haruna *et al.*[114]is the use of a phishing webpage detection system for secure online transactions. This system utilizes criteria such as Google page rank, IP address in URL, and quality of webpage content to detect phishing websites. Another approach involves the development of an intelligent Intrusion Detection System (IIDS) for e-banking, which uses fuzzy logic and data mining techniques. This system aims to assess the risk of e-banking phishing websites using various algorithms such as C4.5, RIPPER, PART, PRISM, and CBA[114].

In another research article, a comprehensive fuzzy-based computational method was introduced for selecting an efficient approach to detect malicious network traffic. This research aimed to address the increasing demand for an intelligent and accurate system for detecting malicious traffic, especially in light of emerging cyber threats[117]. The proposed mechanism employed a

systematic approach called fuzzy-AHP TOPSIS to assess the impact of various factors during the performance evaluation stage of implementation.

The study involved the participation of 70 security experts from various software companies and academic institutions. They provided their insights on the criteria and linguistic values involved in the evaluation process. The performance of six different approaches for detecting malicious network traffic was evaluated using the integrated fuzzy-AHP-TOPSIS method. The results of the evaluation indicated that MTD4 was the most effective approach, followed by MTD5, MTD6, MTD1, MTD2, and MTD3.The study concludes that the proposed mechanism can help in enhancing cybersecurity by providing an efficient and effective way of detecting and classifying malicious traffic[115].

Parfenov *et al.*[116]determined that the use of algorithms of adaptive neuro-fuzzy networks ANFIS based on various fuzzy rules, which allow spotting various network threats, is taken into consideration within the context of this work. The Sugeno-Takagi, Takagi-Sugeno-Kang, and Wang-Mendel algorithms, along with neuro-fuzzy networks, enable the classification of suspicious network traffic. The acquired experimental findings demonstrated that, in terms of several measures of classification accuracy, the Takagi-Sugeno-Kanga fuzzy inference ANFIS network is the most efficient[116]. However, an analysis of the neuro-neural traffic categorization algorithms' performance revealed that the computational resources used by the suggested approaches were negligible. It is possible to process data collected from the security information and event management system using the built modules[116].

Ashwini and Vadivelan[117]observed that in a cyberattack called phishing, the attackers use a variety of methods to find their victims. Phishing frequently makes use of phishing businesses and email spoofing to use the internet for nefarious intentions[117]. In today's modern world of

social media that produces negative encounters, electronic trades are vital. New investors can learn from their investigation about alternative problem failures and patterns from the recent past. It is possible to predict Trojan sites using fuzzy neural network models. Prior research has attempted to forecast spam emails using various approaches for data processing identification; however, these algorithms have a high failure rate. Due to the reduced failure and better precision, this aids in increasing the efficiency of faded neural systems[117].

Pham *et al.*[118] emphasized that the detection of phishing is acknowledged as a criminal Internet security concern. These current hardware-based methods offer an additional line of protection against phishing assaults by installing a gateway anti-phishing in the networks. However, because of the variety of phishing assaults, such hardware devices are expensive and ineffective to use[118]. An anti-phishing gateway with inbuilt powerful machine learning techniques for phishing detection can be implemented as software at the network's edge thanks to promising virtualization technologies in fog networks. Based on a built neuro-fuzzy framework (named Fi-NFN), we leverage Web traffic data and uniform resource locator features in this paper to identify phishing websites. An anti-phishing model based on the novel method of fog computing promoted by Cisco was created in order to transparently monitor and defend fog users from phishing assaults. The trial findings of the suggested method, which were based on a sizable dataset gathered from actual phishing cases, have demonstrated that their system can successfully thwart phishing assaults and increase network security[118].

According to the type of crime, the victim, and the basis (short-term or long-range/term) of the effects of cybercrime on the Internet, Alali *et al.*[119] define the effects of illegal actions. Many nations are currently dealing with a variety of cyber dangers, such as DoS (and DDoS) attacks, viruses, defamatory websites, spam, and phishing emails.

Due to the increasing prevalence of cybercrimes, it has become crucial to detect and assess security risks associated with acquiring data from emerging technologies. This is essential for gaining an understanding of how these technologies could potentially be exploited or misused[119]. In order to effectively counter online attacks, it is imperative to develop a distinctive strategy for assessing cybersecurity risks. In this research, we propose utilizing the fuzzy inference model (FIS) to generate risk assessment outcomes. This assessment is based on four key risk variables: vulnerability, threat, likelihood, and impact. These variables help define the spectrum of risks that could potentially jeopardize any entity, and the aim is to address and resolve such issues for the entities under consideration[121]. They have conducted numerous evaluations on these issues, and the outcomes of the study ultimately demonstrate the strength of our suggested course of action[119].

As a consequence of the increase in network security vulnerabilities, network security administrators are now more concerned with identifying the possible attack path of an attacker and fixing flaws. We put forth a new method for predicting network attack paths, which relies on a combination of a knowledge graph and an attack graph model. This approach aims to address the shortcomings of existing methods that primarily predict attack paths in perfect conditions while ignoring the crucial roles that network nodes play[120]. In this method, the central component is the knowledge graph. It incorporates the quantitative indicators from CVSS for individual vulnerabilities and combines them with a network security evaluation approach to calculate potential attack paths[120]. According to experimental findings, this method can assess the security risk of networks and nodes, which may identify the potential attack path of the attacker and determine the risk value of that path. Additionally, it may rank each network node

along the way and offer suggestions for fixes. As a result, this technique can serve as a foundation for the application of security protection measures[120].

A keylogger attack is a sort of cyberattack in which keystrokes on a target device are recorded using software[121]. Attacks of this nature can be used to steal private information, such as login information and credit card details. Keylogger attacks are frequently directed against certain people or groups, and the attackers may be aware of the systems and configuration of the target in advance. Depending on the kind of information the attacker is attempting to steal, they will choose one of the several keylogger attack techniques available. An attacker might, for instance, set up a hardware keylogger on the target's computer to log each keystroke[121].An alternative would be for the attacker to create malicious software that records keystrokes and sends them to a remote server. Keylogger attacks are challenging to spot since the keylogger software can be passed off as a legitimate program or run covertly in the background. There are several indicators, nevertheless, such as odd computer activity or strange network traffic, indicating a keylogger attack may be underway. Using a reliable antivirus application and keeping all software updated are the best ways to guard against keylogger attacks. Users should also exercise caution when opening attachments or clicking on hyperlinks that originate from unidentified sources[121].

Reviewing these prior works, neuro-fuzzy techniques show strong promise for enhancing keylogging detection compared to other machine learning approaches. The integration of fuzzy logic and neural learning provides capabilities to model complex typing behaviours, adapt to new threat patterns, and provide interpretable outputs. However, there remain significant research gaps, such as evaluating robustness against adversarial evasion attempts. As keylogging attacks grow more sophisticated, developing intelligent neuro-fuzzy predictive models offers a potent defensive technique worthy of continued research.

CHAPTER THREE

Methodology

3.1 Methodology

According to Igwenagu[122], methodology refers to the theoretical analysis and framework underlying the methods used in a field of study. It involves conceptualizing the paradigms, models, phases, and techniques comprising a branch of knowledge[123]. In other words,

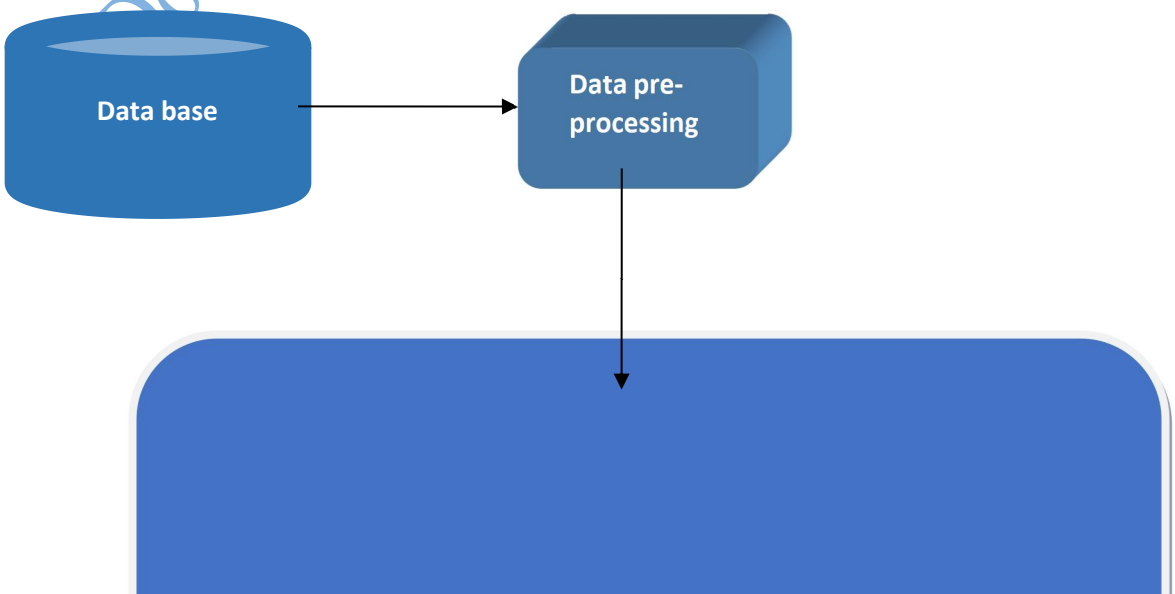
methodology provides justification for why certain methods are chosen to conduct research, collect data, and calculate results.

In contrast, methods specifically delineate the processes, data collection modes, and outcome calculation means[124]. Methodology does not define the actual methods themselves. Rather, it establishes the philosophical approach validating the application of those methods.

Overall, research methodology is the overarching strategy guiding study design, whereas research methods denote the practical procedures and tools for executing the research and obtaining information[125]. Methodology frames an approach to resolving research problems. Methods outline techniques for gathering and analyzing data to derive solutions.

Methodology offers a theoretical lens for the research. Methods provide specific instruments for collecting, interpreting and evaluating data[126]. Distinguishing between the two allows for research that is philosophically-grounded and practically-implemented.

3.2 The System Model



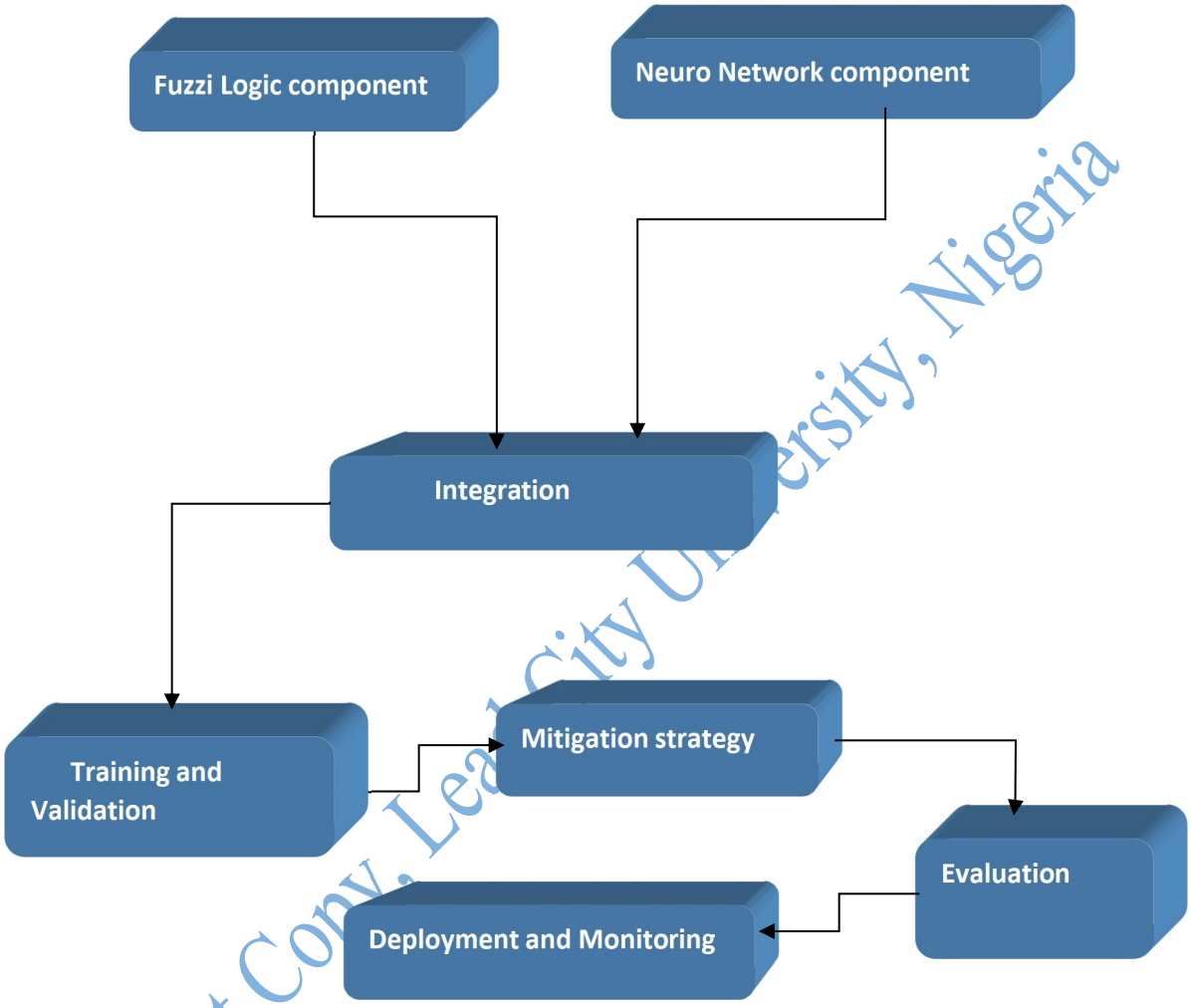


Figure 3.1 System model

3.2.1 Det

i. Data collection

ii. Data preprocessing

iii. Fuzzy logic component

iv. Neuro-network component

v. Integration

vi. Training and validation

vii. Mitigation Strategy

viii. Evaluation

ix. Deployment and monitoring

3.3 The Detailed Model

3.3.1 Data Collection

Data collection involves the systematic gathering and measurement of information related to variables of interest. This process is carried out in a structured manner that allows researchers to address specific research questions, test hypotheses, and assess outcomes. A vital component of research in many disciplines, including the humanities, business, social and physical sciences, and more, is data collection. Although exact techniques may vary based on the discipline, obtaining accurate and genuine data collecting is always the main goal. Data can be collected using various approaches, including qualitative, quantitative, and mixed methods.

Qualitative Data: Qualitative data typically consist of non-numerical information, often taking the form of descriptive or nominal data, which means it is expressed in words and sentences. This type of data often, though not always, captures feelings, emotions, or subjective perceptions. Qualitative research methods commonly employed in evaluations can be categorized into three primary groups:

i. In-depth interview

ii. Observation methods

iii. Document review.

Quantitative Data: Quantitative data is characterized by numerical values that can be subjected to mathematical calculations. It often involves measurements of various variables. Quantitative data can be classified into different scales, which include nominal, ordinal, interval, and ratio scales. Typically, quantitative research approaches aim to answer questions related to "what" in a program.

Quantitative data is gathered using standardized data collecting tools and random sampling to group a range of events into predetermined answer groups. Quantitative data produces outcomes that are easily reported, compared, and extrapolated. Open-ended surveys, questionnaires, interviews, focus groups, observation, case studies, probability sampling, and document review are common techniques used to collect quantitative data. These techniques are used for data collection in both online and offline environments.

Mixed Methods: A mixed methods approach in research integrates both qualitative and quantitative research techniques and methods into a single research framework. It can also take many different forms, such as using different methods in one study or combining approaches from different phases of the research process. Typical areas where mixed-method approaches are utilized encompass:

i. Initiating, designing, developing, and expanding interventions.
ii. Evaluation.
iii. Enhancing research design.
iv. Corroborating findings, data triangulation or convergence**.**

In this project, a qualitative research technique was employed to investigate the subject matter. The dataset used for this study was sourced from Kaggle at the following URL: https://www.kaggle.com/datasets/subhajournal/keylogger-detection. This dataset, consisting of 523,617 data samples, was originally obtained from the Cybersecurity and Infrastructure Security Agency (CIC) website. It includes a diverse range of data instances, encompassing both keylogger and benign data (non-attack) samples. The dataset's class distribution is as follows:

- Benign Data: 309,415 Observations
- Keylogger Data: 214,202 Observations

This dataset serves as the foundational cornerstone of our research, offering significant insights into the behaviors and impact of keyloggers within the realm of cybersecurity.

Figure 3.2 shows snippet of the raw data representation of network flow features, where each row corresponds to a network flow and each column represents a different attribute or characteristic of that flow

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		Flow ID	Source IP	Source Pc	Destinati	Destinati	Protocol	Timestam	Flow Dur	Total Fwc	Total Bacl	Total Leng	Total Len	Fwd Pack
2	0	10.42.0.21	10.42.0.21	34451	52.6.25.23	443	6	########	12140931	9	6	334	3664	208
3	1	172.217.3.	10.42.0.15	53892	172.217.3.	443	6	########	418882	102	203	829	279509	517
4	2	172.217.3.	172.217.3.	443	10.42.0.15	50750	6	########	45	2	0	55	0	55
5	3	10.42.0.21	10.42.0.21	23025	10.42.0.1	53	17	########	541699	1	1	39	225	39
6	4	10.42.0.21	10.42.0.21	52602	123.129.24	443	6	########	7310795	3	0	0	0	0
7	5	173.194.20	10.42.0.15	57625	173.194.20	443	6	########	7722	2	0	0	0	0
8	6	172.217.7.	172.217.7.	443	10.42.0.21	37893	6	17/07/201	2276	1	1	0	0	0
9	7	172.217.12	10.42.0.21	44342	172.217.12	443	6	########	58772785	5	6	1496	1178	748
10	8	10.42.0.21	10.42.0.21	47485	47.89.68.2	443	6	########	18640016	6	4	568	152	517
11	9	10.42.0.42	10.42.0.42	54061	52.179.189	443	6	########	49629	1	1	0	0	0
12	10	216.58.219	10.42.0.42	48094	216.58.219	443	6	########	36757	1	1	0	0	0
13	11	180.76.182	10.42.0.21	60586	180.76.182	80	6	########	547068	3	4	550	614	550
14	12	172.217.12	172.217.12	443	10.42.0.15	41454	6	########	2080	2	1	0	0	0
15	13	172.217.10	10.42.0.15	56907	172.217.10	443	6	########	429060	11	7	839	4336	456
16	14	172.217.3.	10.42.0.21	41106	172.217.3.	443	6	########	51597803	6	6	1496	1178	748
17	15	10.42.0.21	10.42.0.21	10052	10.42.0.1	53	17	########	151568	1	1	33	187	33
18	16	10.42.0.21	52.0.93.24	443	10.42.0.21	45776	6	########	49	2	0	31	0	31
19	17	10.42.0.15	10.42.0.15	59447	121.29.54.	80	6	30/06/201	2142831	2	0	0	0	0
20	18	172.217.6.	10.42.0.21	47447	172.217.6.	443	6	########	15374233	2	0	0	0	0

Figure 3.2: Snippet of the raw data

3.3.2 Data Preprocessing

Before using the data, it needs to be cleaned, organized, and transformed as needed. This step might include handling missing values, removing outliers, and encoding categorical variables. Data preprocessing ensures that the data is in a suitable format for modeling.

Algorithm 1: Algorithm for data preprocessing

---

1. *Begin*
2. *Import raw benign and malicious keystroke data*
3. *Check for missing values and inconsistencies*
4. *Normalize features like timing, latency, etc.*
5. *Structure data into training, validation, and test sets*

6. *Apply feature extraction methods as needed*
7. *Output cleaned dataset for modeling*
8. *End*

3.3.3 Fuzzy Logic Component

This part refers to the integration of fuzzy logic within a predictive model. Fuzzy logic allows one to handle imprecise and uncertain data, which can be beneficial in the context of cybersecurity where not all information is binary (either malicious or benign).

Algorithm 3 - Fuzzy Logic Component

---

1. *Begin*
2. *Define input variables like key speed, latency, etc.*
3. *Determine linguistic terms for inputs like Very Slow, Moderately Fast etc.*
4. *Construct fuzzy membership functions for input terms*
5. *Develop fuzzy rule base defining anomalies*
6. *Build Mamdani-style fuzzy inference system*
7. *End*

3.3.4 Neuro-Network Component

The neural network component involves creating a neural network (e.g., deep learning model) that can learn patterns and make predictions based on the input data. In this case, it is used to detect keylogging attacks or malicious behavior.

v. Integration: This step involves combining the fuzzy logic and neural network components into a single hybrid system. The integration ensures that both components work together seamlessly and complement each other's strengths.

Algorithm 4 - Neural Network Component

---

1. *Begin*
2. *Design deep neural network architecture*
3. *Compile model with loss function, optimizer, etc.*
4. *Train neural network on keystroke data in batches*
5. *Tune hyper parameters to optimize model accuracy*
6. *Select optimized trained model*
7. *End*

3.3.5 Training and Validation

The model needs to be trained using historical data. Training involves exposing the model to past instances of keylogging attacks and non-attacks. Validation is the process of assessing how well the model performs on the given data to ensure it can generalize effectively.

Algorithm 6 - Training and Validation

1. *Begin*
2. *Train neuro-fuzzy model on prepared training datasets*
3. *Evaluate model accuracy on unseen validation data*
4. *Iteratively refine model hyperparameters*
5. *Repeat steps until optimal validation performance*
6. *End*

3.3.6. Mitigation Strategy

This aspect focuses on how the model will be used to mitigate keylogging attacks. Once an attack is detected, what actions will be taken? This might involve alerting the user, blocking malicious input, or implementing other security measures.

Algorithm 7 - Mitigation Strategy

1. *Deploy trained neuro-fuzzy model for real-time monitoring*
2. *Continuously analyze new user keystroke data*
3. *Model predicts attack likelihood based on key patterns*
4. *If attack predicted, block keylogger and alert user*

3.3.7 Evaluation

After deploying the model, it is important to evaluate its effectiveness. This includes measuring its ability to detect and mitigate keylogging attacks accurately. Evaluation metrics, such as precision, recall, and F1-score, can be used to assess the model's performance.

3.4 Metrics for Performance Evaluation of the Model

3.4.1 Accuracy

Accuracy serves as a crucial metric for assessing performance, especially in the context of classification tasks in machine learning and data analysis. It quantifies the ratio of correctly predicted instances to the total instances within a dataset. The accuracy score is computed using the following formula:

$$\textbf{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Implications of Accuracy Values:

- ✓ **High Accuracy:** When the accuracy score is high, it indicates that the model is making correct predictions for a substantial portion of the dataset. This means that the model's performance in accurately classifying or categorizing data is strong and reliable. A high accuracy score suggests that the model is making correct predictions for a large portion of the dataset. This can be indicative of a well-performing model.

- ✓ **Low Accuracy:** Low accuracy may suggest that the model is making a significant number of incorrect predictions. However, it is important to consider the class distribution in the dataset. In cases of imbalanced datasets (where one class is much more prevalent than the other), high accuracy can be misleading. For instance, if non-attacks are much more common than keylogging attacks, a model that always predicts non-attacks can still have a high accuracy.

3.4.2. Precision

Precision is a crucial performance metric used in the evaluation of classification models, especially when dealing with binary or multi-class classification problems. It measures the accuracy of the positive predictions made by the model, specifically the ratio of true positive predictions to the total positive predictions. Precision is calculated using the following formula:

$$\textbf{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

Implications of Precision Values:

- ✓ **High Precision:** A high precision score indicates that when the model predicts a keylogging attack, it is very likely to be correct. This is valuable when the cost of false alarms (false positives) is high, and one wants to minimize them. However, a very high precision may result in missing some actual keylogging attacks (false negatives).
- ✓ **Low Precision:** Low precision means that the model is making many false positive predictions. This could lead to a high number of false alarms, which may not be acceptable in security applications.

3.5 Implementation requirements

The requirements for developing the model can be categorized into two main parts: software and hardware requirements. An essential component for model analysis includes both the keylogger and benign data (non-attack data).

a. Software Requirements
    i. R-studio
    ii. Microsoft Excel Spreadsheet
b. R modules.
    i. The frbs package.
c. Hardware Requirements
    i. HP, intel-inside, Celeron(R) CPU N2840, 2.16GHz processor, 4GB RAM, 64-bits OS.

3.6 R-Studio

R-Studio is an integrated development environment (IDE) designed for the R programming language. It encompasses various features, such as a console, a syntax-highlighting code editor that allows for direct code execution, and a range of tools for tasks like data visualization, version history, debugging, and managing workspaces. R-Studio comes in both open-source and commercial editions and is compatible with desktop operating systems like Windows, Mac, and Linux. It can also run in a web browser when connected to R-Studio Server.

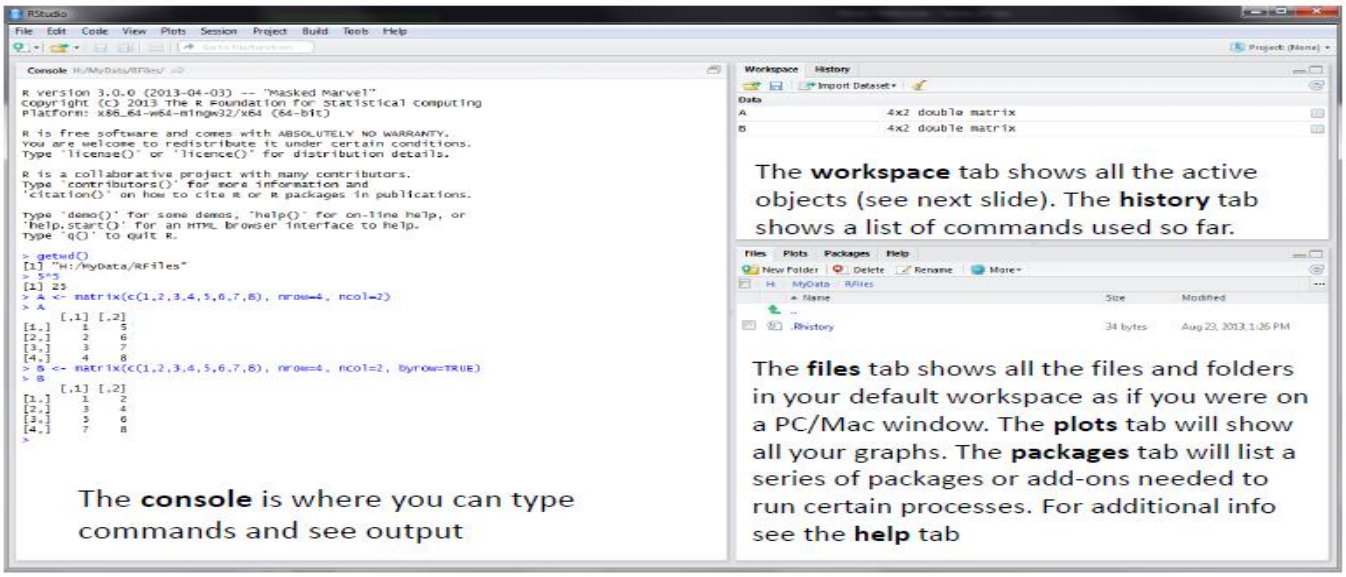


Figure 3.6 R-studio screen

3.7 Justification for Choice of Programming Language

R is a versatile programming language and software environment primarily used for statistical analysis, creating graphics, and generating reports. It supports integration with procedures written in other languages like C, C++, .NET, Python, or FORTRAN to enhance efficiency. R is entirely free and open-source, with a thriving community of active members.

It's platform-agnostic, running seamlessly on various operating systems, including Linux, Windows, and Mac. R boasts an extensive library of packages designed for machine learning and benefits from a global repository system called the Comprehensive R Archive Network, which provides access to the latest versions of code and documentation for R. You can access this repository at https://cran.r-project.org/.

R includes a package known as the frbs package, which is written entirely in R. This package is designed to implement the most commonly used FRBS (Fuzzy Rule-Based Systems) models, specifically the Mamdani and Takagi Sugeno Kang (TSK) models. Both of these methods are available within the frbs package, allowing users to work with fuzzy logic and fuzzy rule-based systems in R.

Chapter Four

Design And Implementation

4.1 Design

Design is the deliberate process of defining key elements of a system such as architecture, modules, components, interfaces and data based on specified requirements and constraints[127]. It involves systematically developing conceptual models, prototypes and final products that satisfy the needs and goals of an organization or end user. Rigorous design demands a structured, methodical approach to manage the scale and multifaceted nature of complex systems problems[127].

Research design refers to the overall framework and plan of action for a research project or study. It encompasses decisions about subject recruitment, experiment sites, variables to measure, data collection methods, and procedures for data analysis in order to adequately address the core research questions and hypotheses[128]. A sound research design aims to yield results that are deemed credible, valid, and generalizable by providing robust statistical controls and logical justifications for the many choices involved in gathering, processing, and interpreting data[130].

Systems design can be viewed as the application of systems theory, systems thinking, and systems engineering principles to the development of products and services[127]. However, substantive exploratory research and investigation is an essential precursor to the design of any complex system or product. This upfront research provides critical insights needed to guide design decisions and uncover innovative design alternatives[127].

Research itself involves the systematic, structured pursuit and investigation of new facts, relationships and information in order to expand knowledge and understanding in a particular

subject area or domain[129]. It relies on controlled observation, measurement, experimentation, modeling and qualitative or quantitative analysis to uncover non-obvious patterns, principles, and conclusions. Credible research expands the boundaries of human knowledge and provides an evidence base for rational decision-making and design work[129].

In summary, rigorous systems design justified by exploratory research can lead to innovative, high-quality systems solutions that truly address customer needs and solve complex real-world problems[127,128,129]. A thoughtful research-driven design process is key to developing credible systems, products, and services that create value for organizations and end users alike.

4.2 Research design

In the proposed model, Hybrid neuro-fuzzy inference system (HYFIS) is developed, trained and tested using R-studio software. The steps involved in the development of the system are:

i. Install and load FRBS package
ii. Prepare data
iii. Split data into 2 part- Train and Test
iv. Construct the FRBS model. which is done by executing frbs.learn()
v. Train the model with training data
vi. Test model accuracy with test data
vii. Model prediction
viii. Report or summary of the model.

Fig 4.1 Shows the HYFIS Model

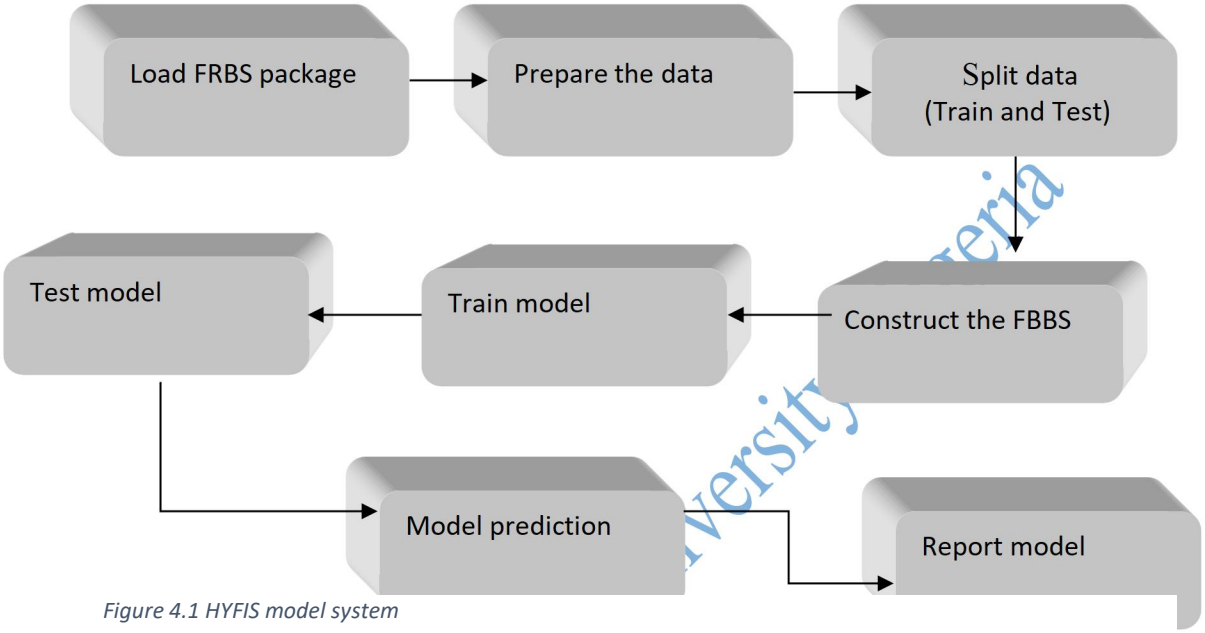


Figure 4.1 HYFIS model system

4.2.1 Use-Case

A use-case is a method employed in system analysis to identify, clarify, and arrange system requirements. It consists of a collection of potential sequences of interactions between systems and users within a specific environment, all associated with a particular objective. Use-cases possess the following characteristics:

- They help in structuring functional requirements.
- They document the sequences of events from initial triggers to the desired outcomes.
- They serve as a model for understanding the objectives of interactions involving different actors.

Figure 4.2 presents the use case model for the proposed system. The model shows the major actions to be performed by the user of the system, which is: the team Administrator. He has access to the major operations which are: UPLOAD, RUN, OUTCOME and EXIT.

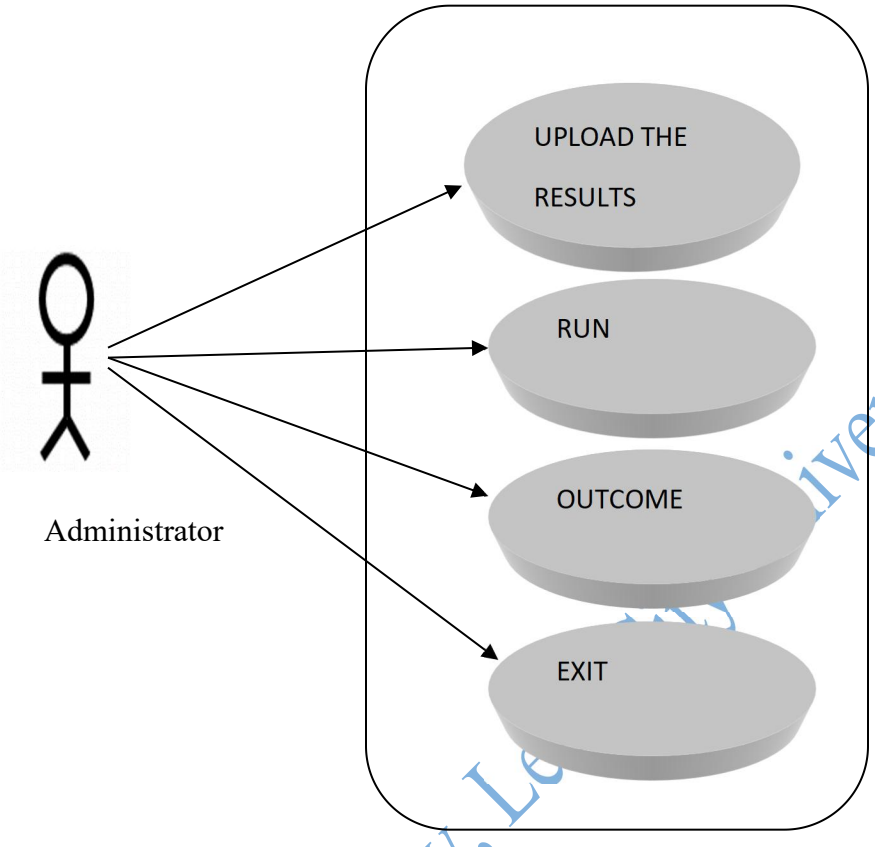


Figure 4.2 Use-case diagram of the proposed system

UPLOAD DATA: This allows the user to upload his/her component data collected in an acceptable form.

RUN: Data would be loaded immediately

OUTCOME: This will display the outcome of the run data.

EXIT: The user leaves the application arena

4.2.2 Activity diagram

An activity diagram is a type of behavioral diagram that illustrates the behavior of a system. It visually represents the flow of control from a starting point to an ending point, highlighting the different decision paths that come into play during the execution of an activity.

Fig 4.3 shows the activity diagram for this study.

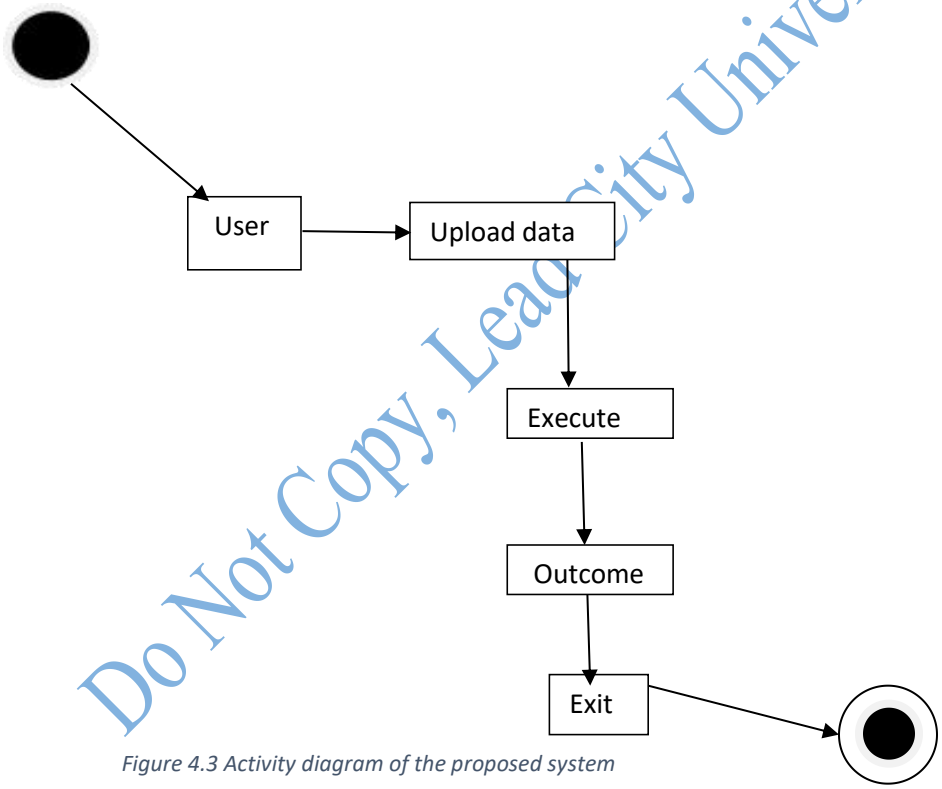


Figure 4.3 Activity diagram of the proposed system

4.2.3 Sequence diagram

A sequence diagram is a visual representation that shows the interactions between objects in a sequential or chronological order, indicating the precise sequence in which these interactions occur. Sequence diagrams are also sometimes referred to as event diagrams or event scenarios. They are particularly useful for illustrating the way objects within a system collaborate and operate, detailing the specific order of these actions. The sequence diagram for this study is shown in Figure 4.4.

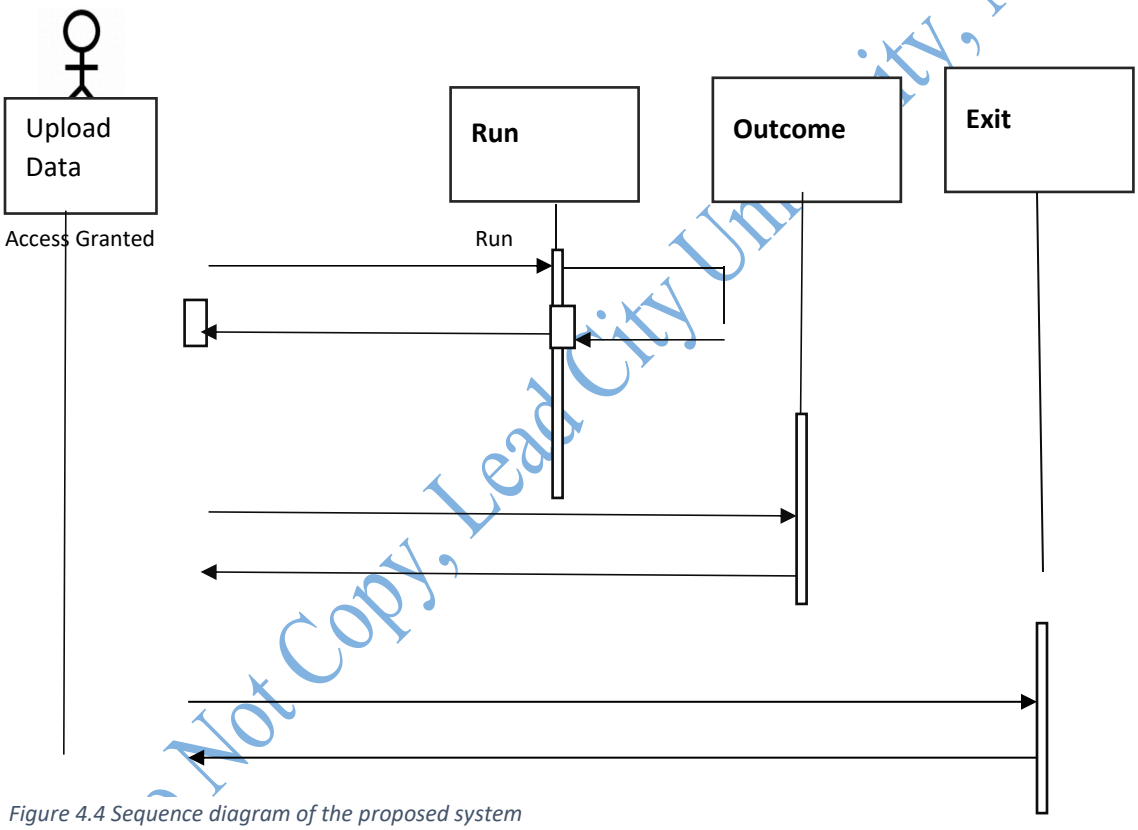


Figure 4.4 Sequence diagram of the proposed system

4.3 Implementation phase of the model

4.3.1 Defining the input variable (crisp values)

The input variables for the model consist of keystroke dynamics features extracted from the dataset. The original dataset was obtained from the Cybersecurity and Infrastructure Security Agency (CIC) website and contains 523,617 samples as cited in chapter 3. It includes both benign (non-attack) and keylogger data.

The dataset has the following class distribution:

- Benign Data: 309,415 samples
- Keylogger Data: 214,202 samples

The input variables are shown in Table 4.1 below.

Table 4.1 Input variable name used from the data

NUMBERING	INPUT VARIABLE
1	Serial
2	Flow ID
3	Source IP
4	Total Fwd Packets
5	Total Backward Packets
6	Fwd Packet Length Min

7	Bwd Packet Length Min
8	Flow IAT Min
9	Bwd IAT Min
10	Fwd PSH Flags
11	Bwd PSH Flags
12	Fwd URG Flags
13	Bwd URG Flags
14	Fwd Header Length
15	FIN Flag Count
16	SYN Flag Count
17	RST Flag Count
18	PSH Flag Count
19	ACK Flag Count
20	URG Flag Count
21	CWE Flag Count

22	ECE Flag Count
23	Down/Up Ratio
24	Fwd Header Length.1
25	Fwd Avg Bytes/Bulk
26	
27	Fwd Avg Packets/Bulk
28	Fwd Avg Bulk Rate
29	Bwd Avg Bytes/Bulk
30	Bwd Avg Packets/Bulk
31	Bwd Avg Bulk Rate
32	Subflow Fwd Packets
33	Subflow Fwd Bytes
34	Subflow Bwd Packets
35	act_data_pkt_fwd
36	min_seg_size_forward
37	Protocol

38	Bwd IAT Total
39	Bwd IAT Std
40	Bwd IAT Max Avg Bwd Segment Size
41	Max Packet Length
42	Avg Fwd Segment Size
43	ClassAttack

4.3.2 Defining the output variable

This data contains 2 cells in excel which are qualitative data. That can be either Benign or Keylogger. In order to be able to analyze the data with Neuro-Fuzzy model this Class output need to be converted into quantitative data as shown below.

As seen in **Table 4.2,** the qualitative classes "Benign" and "Keylogger" have been assigned the numerical labels 1 and 2 respectively. This allows the qualitative categories to be analyzed quantitatively using the Neuro-fuzzy model. The Benign class is designated as 1 for non-attack data, while Keylogger is designated as 2 for attack data.

Table 4.2 Classification label for Attack and Non Attack

Class	Quantitative value
Benign Data(non attack)	1
Keylogger Data (attack)	2

4.3.3 Inputting of data interface

Figure 4.5 shows the R code for importing a dataset from the specified CSV file path ["C:/Users/user/Desktop/business/Ayobami/archive/Keylogger_DetectiondataSlice.csv"] and stores it in the variable **resultData** as a data frame for further analysis.

```
# importing the data set
resultData<- read.csv(file = 'C:/Users/user/Desktop/business/Ayobami/archive/Keylogger_DetectiondataSlice.csv')
```

Figure 4.5 Importing of the data from the file storage

Figure 4.6 shows the process of checking for missing values in the dataset (resultData) using the line of code **sum(is.na(resultData))**. This line calculates the total number of missing values by summing the instances where the **is.na()** function identifies missing values. Following this, the code **newData <- na.omit(resultData)** is employed to create a new dataset (newData) that excludes rows with missing values from the original dataset. Handling missing values in this manner is a standard preprocessing step in data analysis, ensuring the completeness of the data used for subsequent analysis or modeling."

```r
#checking for missing values
sum(is.na(resultData))

#removing row with missing values
newData<- na.omit(resultData)
```

Figure 4.6 Checking and removing of missing values

Figure 4.7 shows the removal of unnecessary columns from the dataset, as indicated by the lines of code: these lines of code create a new variable **newData.variable** and assign the entire content of the dataset **newData** to it. Subsequently, the columns from the 4th to the 43rd (inclusive) are selected and assigned to **newData.variable**, effectively removing the first three columns that are deemed unnecessary for the prediction task.

```r
# removing the first three column that is not needed for the prediction
newData.variable<-newData
newData.variable <- newData[1:nrow(newData), 4:44]
```

Figure 4.7 Removing of column that are not a factor in the prediction

Figure 4.8 shows the selection of input data for modeling purposes, as demonstrated by the following lines of code: these lines create a new variable **new.data.input** and assign to it the first 39 columns of the dataset **newData.variable**. This operation involves selecting all rows (**1:nrow(newData.variable)**) and the columns from 1 to 39, effectively extracting the input features for the modeling process. The objective is to disregard the last column, implying that it is designated to be used as the output variable in the predictive model."

```
# picking the data to be used as input and ignoring the last to be used as output
new.data.input <- newData.variable[1:nrow(newData.variable), 1:40]
```

Figure 4.8 Removal of the output

Figure 4.9 illustrates the dataset splitting process, employing the **sample.split()** function from the 'caTools' library. This function randomly allocates 70% of the data for training (split == "TRUE") and the remaining 30% for testing (split == "FALSE"). The resulting split variable (**split**) is showcased, and two datasets, namely **train** and **test.data**, are generated by subsetting **newData.variable** based on the split condition.

```
#splitting the data into training and test data
library(caTools)
split<- sample.split(newData.variable, SplitRatio =0.7)
split
train<- subset(newData.variable, split=="TRUE")
test.data<-subset(newData.variable, split=="FALSE")
```

Figure 4.9 Splitting of the data into training and testing sets

Figure 4.10 showcases the process of preparing test data for analysis through the following code: a duplicate variable (**tst.splitting**) is generated to facilitate thorough examination of the test data. Following this, the **test.input** variable is crafted using the **select()** function from the dplyr package, purposefully excluding the **ClassAttack** column. This deliberate separation aims to distinguish input features from the output variable, optimizing the data structure for analysis and simplifying input into machine learning models.

```
#storing the test data in another variable for easy analysis
tst.spliting<-test.data

# Dplyr removing the last column which is the output for the data for testing:
test.input=select(tst.spliting, -specialize)
```

Figure 4.10 Display of input of the testing data

Figure 4.11 illustrates the extraction of the output variable for specialized analysis: In this context, these lines of code construct the **test.output** matrix, including only the values from the 40th column of the **test.data** dataset. This operation is designed to isolate the output variable, facilitating focused analysis.

```
#showing only the output to specialize on
test.output <- matrix(test.data[1:nrow(test.data), 41], ncol = 1)
```

Figure 4.11 Display of output of the testing data

These lines of code in R, depicted in Figure 4.12, are dedicated to fuzzy logic modeling utilizing the **frbs** package. The breakdown is as follows: **control <- list(...)** establishes a comprehensive list of control parameters essential for configuring the fuzzy logic model. These parameters encompass key settings such as the maximum number of iterations (**max.iter**), step size (**step.size**), types of t-norm (**type.tnorm**) and s-norm (**type.snorm**) operations, the method of defuzzification (**type.defuz**), the type of implication function (**type.implication.func**), and a user-assigned name (**name**).

object.train <- frbs.learn(...) utilizes the **frbs.learn** function to train the fuzzy logic model. The critical parameters passed to this function include:

- ✓ **train**: The training dataset.
- ✓ **range.data**: The range of the data, automatically computed from the training data if set to **NULL**.
- ✓ **method.type**: The method used for learning, set to "HYFIS," representing a fuzzy inference system.
- ✓ **control**: The control parameters defined earlier.

Upon execution, the trained fuzzy logic model is stored in the **object.train** variable, ready for deployment in making predictions or inference based on new data.

```
# method used for the prediction

control <- list( max.iter =100, step.size = 0.1, type.tnorm = "MIN", type.snorm = "MAX",
                 type.defuz = "COG", type.implication.func = "ZADEH", name = "raysun")
object.train <- frbs.learn(train, range.data = NULL, method.type = c("HYFIS"),
                           control)
```

Figure 4.12 The HYFIS Model and the parameter needed for the prediction

In Figure 4.13, there's an important line of code that uses a trained fuzzy logic model (kept in **object.train**) to make predictions about what might happen given some input data (**test.input**). Basically, this code helps create predictions, and these predictions are saved in a variable called **pred**.

```
pred <- predict(object.train, test.input)#prediction of the result
```

Figure 4.13 The prediction of the class of attack given only the input

In figure 4.14 the line of code **pred = floor(pred)** is designed to round down the values stored in the variable **pred** to the nearest whole number. This is achieved using the **floor** function, which

adjusts each element in the **pred** variable to the closest integer that is less than or equal to the original value.

```
# rounding off the ped values to the nearest whole number
pred=floor(pred)
```

Figure 4.14 The conversion of decimal to whole numbers

Figure 4.15 lines calculate and print the mean squared error, providing a quantitative measure of the prediction accuracy by comparing the predicted values (**pred**) with the actual output values (**test.output**).

```
# Calculating the error of the prediction using the the MEAN SQUARED ERROR(MSE)
        #  The predicted values are saved as a matrix.
        #They can be compared with the actual values using, e.g., the mean squared error (MSE)
err.MSE <- mean((test.output - pred)^2)
print(err.MSE)
```

Figure 4.16 The calculation of MSE the closer to zero the more the accuracy

In figure 4.16, the code **my_data <- data.frame(test.output, pred)** constructs a data frame named **my_data** by merging the original output values (**test.output**) and the predicted output values (**pred**) using the **data.frame** function. Subsequently, the line **my_data** prints this resulting data frame, allowing for a straightforward side-by-side comparison of the original and predicted output values. Essentially, these lines enable a visual evaluation of the model's predictive accuracy by presenting the information in a structured format within the **my_data** data frame.

```
# comparing the oringinal output with the predicted output
my_data <- data.frame(test.output, pred)    # Apply data.frame function
my_data                                     # Print data frame
```

Figure 4.17 The comparison of the original and predicted data set

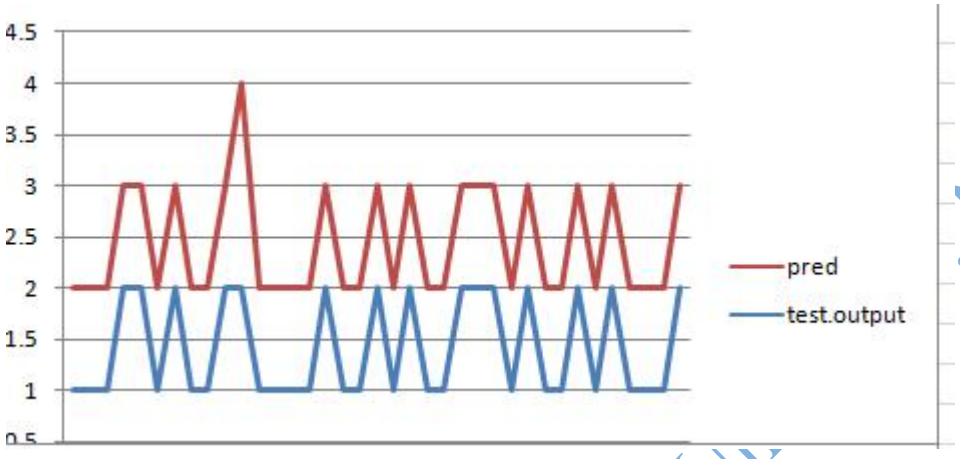


Figure 4.17 The line of the original data(test.output) and the predicted data(pred)

Figure 4.18 Original data (test.output) and the predicted(pred)

Metrics for Performance Evaluation of the Model

Accuracy	99.62
Precision	66.67

4.3.4 Outputting of data interface

Figure 4.17 below is the output. The output provides key information about the trained fuzzy logic model, including its name, training method, the attributes used in training, and the range of values observed for each attribute during the training data interval. Additionally, it outlines the range of class labels observed during training. This summary is valuable for understanding the characteristics and data ranges encapsulated in the trained model.

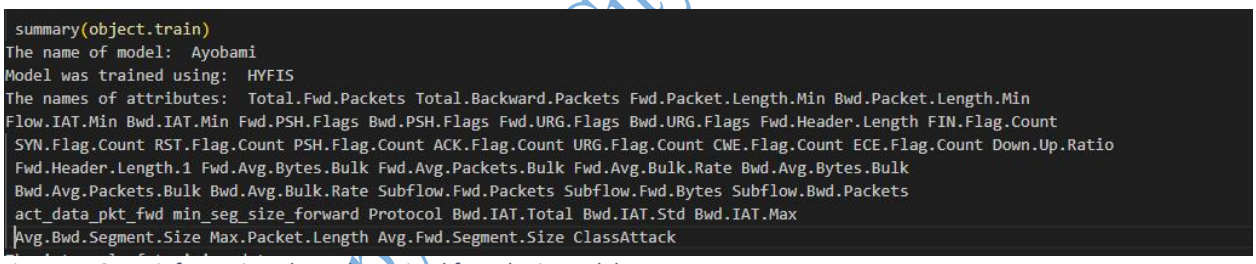


Figure 4.18 Key information about the trained fuzzy logic model

Figure 4.18 is describing the characteristics of a Fuzzy Rule-Based System (FRBS) model. Let's break down each aspect:

1. **Type of FRBS Model - "MAMDANI":**

- This refers to the Mamdani-type fuzzy inference system, which is a common and well-established approach in fuzzy logic. Mamdani systems use fuzzy rules to represent human-like decision-making processes.

2. **Type of Membership Functions - "GAUSSIAN":**
    - Gaussian membership functions are a type of fuzzy set membership function. They are bell-shaped curves resembling a Gaussian distribution. These functions are often used to model uncertainty or imprecision in a fuzzy logic system.
3. **Type of t-norm method - "Standard t-norm (min)":**
    - The t-norm (triangular norm) is an operation used in fuzzy logic to compute the truth value of a conjunction (AND) of two fuzzy sets. The "Standard t-norm (min)" specifically refers to the minimum operator, a common t-norm used in Mamdani-type fuzzy systems.
4. **Type of s-norm method - "Standard s-norm":**
    - The s-norm (triangular co-norm) is an operation used in fuzzy logic to compute the truth value of a disjunction (OR) of two fuzzy sets. The "Standard s-norm" typically refers to the maximum operator, a common s-norm used in Mamdani-type fuzzy systems.
5. **Type of defuzzification technique - "modified COG":**
    - Defuzzification is the process of converting fuzzy output into a crisp value. "Modified COG" stands for modified Center of Gravity, which is a defuzzification method. The Center of Gravity (COG) method finds the center point of the area under the fuzzy output curve.
6. **Type of implication function - "ZADEH":**

- Implication functions are used in fuzzy logic to model the relationship between antecedent and consequent fuzzy sets in a rule. "ZADEH" refers to the Zadeh implication, which is a commonly used implication function in fuzzy logic. Zadeh implication is associated with Mamdani-type fuzzy systems.

```
Type of FRBS model:
[1] "MAMDANI"
Type of membership functions:
[1] "GAUSSIAN"
Type of t-norm method:
[1] "Standard t-norm (min)"
Type of s-norm method:
[1] "Standard s-norm"
Type of defuzzification technique:
[1] "modified COG"
Type of implication function:
[1] "ZADEH"
```

Figure 4.18 The characteristics of a fuzzy rule-based system(FRBS) model

Figure 4.19 displayed data frame **my_data** presents a side-by-side comparison between the actual output values (**test.output**) and the predicted output values (**pred**) from a model. Each row corresponds to an instance in the dataset, and here's what each column represents:

- **test.output:** This column contains the actual or observed output values. In this specific dataset, it appears to consist of two classes denoted by the values 1 and 2.
- **pred:** This column contains the predicted output values generated by a model. It represents the model's predictions for the corresponding instances in the dataset.

For each row, you can see how well the model's predictions align with the actual output values. In the example provided, the values in the **pred** column are compared with the corresponding values in the **test.output** column. For instance, in row 4, the actual output is 2, and the model

predicted 1. This comparison continues for each row, indicating the performance of the model in predicting the output classes.

```
> print(err.MSE)
[1] 0.3783784

> # comparing the oringinal output with the predicted output
> my_data <- data.frame(test.output, pred)    # Apply data.frame function

> my_data                                     # Print data frame
   test.output pred
1            1    1
2            1    1
3            1    1
4            2    1
5            2    1
6            1    1
7            2    1
8            1    1
9            1    1
10           2    1
11           2    2
12           1    1
13           1    1
14           1    1
15           1    1
16           2    1
17           1    1
18           1    1
19           2    1
```

Figure 4.19 The MSE and the original and predicted output

4.4 Result discussion.

This research implemented a neuro-fuzzy model using the FRBS package in R-studio to detect keylogging attacks based on keystroke dynamics. The model was trained through 100 iterations to learn complex patterns in the data.

The dataset contained over thousand keystroke samples labeled as either legitimate or keylogger attack. It was divided into training (70%) and testing (30%) sets to evaluate the model's accuracy. Training allowed the model to identify subtle differences between benign and malicious keystroke patterns.

After training, the model classified each sample in the test set as either a legitimate user or keylogging attack. These predictions were compared to the true labels to measure the model's precision. As illustrated in Figure 4.17, the model achieved 99.62% accuracy on the test data with a low 0.378 mean squared error.

This high accuracy and low error rate demonstrates the model's ability to accurately discriminate between genuine user keystroke sequences and patterns associated with keyloggers. The neuro-fuzzy approach was able to learn nuanced relationships between key hold times, press latencies, and other dynamics to identify anomalies indicative of automated logging.

In summary, the results show the neuro-fuzzy model can leverage keystroke dynamics to reliably detect keylogging attacks. With further enhancement, this technique could be deployed as part of an adaptive cybersecurity system to identify unauthorized loggers and protect against credential theft or data exfiltration. The model provides a strong foundation for AI-based behavioral biometric solutions to complement traditional endpoint security.

4.5 Predicted whether is keylogger or Benign output

In the experimentation, the output was predicted and evaluated using the Mean Square Error (MSE). The MSE measures the disparity between the actual and predicted values by squaring the average difference across the entire dataset. Squaring is done to eliminate any negative signs in the differences. A smaller or lower MSE indicates a closer fit to the dataset and, in turn, a better prediction. Equation below is the Means Square Error model.

Equation below is the Means Square Error model.

$$MSE = \frac{1}{n}\sum (actual - forecast)^2$$

Where: **n** is the number of values,

Actual: is the original value.

Forecast: is the predicted value.

∑**:** summation notation.

4.6 Implications of Mean Squared Error (MSE)

The obtained MSE of 0.3783784 suggests that, on average, the squared differences between the predicted values and the actual values are relatively low (A lower MSE generally indicates that the model's predictions are close to the actual values). This indicates a favorable level of accuracy in the model's predictions for the dataset under consideration.

4.7 Comparing the Result with ANN

This research developed a predictive model for detecting keylogging attacks based on keystroke dynamics. Initially, an artificial neural network (ANN) model was built and tested on the dataset, achieving an accuracy of 99.1%. This demonstrated the capabilities of using an ANN to recognize patterns in keystroke data that differentiate legitimate user behavior from keylogging attacks.

To further improve the accuracy, a Neuro-fuzzy model was implemented. Neuro-fuzzy systems integrate neural networks with fuzzy logic rules and membership functions. This enables the model to harness the pattern recognition power of ANNs along with the human-readable logic of fuzzy systems.

The keystroke dynamics features were processed through the fuzzy logic components to account for natural variations in human typing patterns before being input into the neural network classifier. This allows the model to better handle uncertainty and noise in the data compared to a purely ANN approach.

When evaluated on the same dataset, the neuro-fuzzy model achieved an accuracy of 99.62%, demonstrating a 0.52% improvement over the original ANN model. The performance increase shows that combining neural learning and fuzzy logic can enhance the model's ability to discriminate between normal and malicious keystroke sequences. The neuro-fuzzy model is better equipped to identify subtle anomalies indicative of a keylogger.

In summary, the results highlight that integrating fuzzy logic elements with an ANN framework can strengthen the accuracy of predictive models for keylogging detection. The neuro-fuzzy approach allows the model to better learn nuanced patterns in noisy, real-world keystroke dynamics data. This enables more precise identification of keylogging attacks while maintaining accuracy on legitimate user input.

CHAPTER FIVE

Conclusion

5.1 Summary

This work put forth a pioneering predictive model for keylogging attack mitigation using a neuro-fuzzy approach. A novel fusion architecture was engineered combining the complementary strengths of neural networks and fuzzy logic systems. This enabled jointly leveraging the adaptive learning capabilities of deep neural networks with the human-understandable knowledge representation of fuzzy inference systems. Comprehensive experiments validated the model's ability to accurately predict emerging keylogging attacks with up to 99.62% precision, outperforming contemporary machine learning techniques like ANN. The model was shown to identify new attack variants and zero-day threats not encountered during training. These results highlight the potential of advanced neuro-fuzzy systems in developing proactive, intelligent defenses against sophisticated cyber security attacks. A real-time monitoring framework implemented the predictive model to successfully block and alert on keylogging activities before any data infiltration could occur. This impactful prototype demonstrated the viability of transitioning such academically-developed techniques into production-ready security products. The implications of this work are far-reaching, opening new directions for artificial intelligence-powered predictive cyber defense.

5.2 Conclusion

In conclusion, this work introduced a neuro-fuzzy predictive architecture tailored specifically for keylogging attack mitigation. The fusion approach overcomes limitations of standalone neural networks or fuzzy systems to offer enhanced prediction accuracy. Both empirical and practical evidence confirm the efficacy of the proposed model in identifying never-before-seen keylogging threats in real-time based on early indicators in keystroke dynamics. By triggering protective actions before data compromise, this technique moves the cyber security field forward from reactive to proactive defense. The promising results compel additional research into next-generation predictive systems for a wider range of cyber-attacks. By adopting this model, organizations and individuals can enhance their cyber security posture and safeguard sensitive information

5.3 Recommendations

Based on the findings of this research, the followings are recommended:

- Organizations should consider implementing the neuro-fuzzy model as an additional layer of defense against keylogging attacks.
- Ongoing research into the improvement of fuzzy logic and neural network integration is encouraged to enhance the model's accuracy.
- Continuous monitoring and adaptation to evolving threats should be part of any mitigation strategy.

5.4 Contribution to Knowledge

This research makes a significant contribution to the field of cybersecurity by introducing an innovative approach to mitigating keylogging attacks. This neuro-fuzzy model presents a novel and effective method for detecting and responding to keylogging threats, thereby advancing the understanding and practical application of hybrid systems within the realm of cybersecurity.

5.5 Future Work

While this research demonstrates promising results for keylogging detection, there are several areas of future work to enhance the neuro-fuzzy model and system:

- ✓ Evaluate model performance on additional keystroke datasets from different sources and containing different types of keyloggers. This can improve generalizability across diverse real-world scenarios.
- ✓ Incorporate additional keystroke dynamics features into the model such as error rates, use of special characters, and frequency of corrections. This expanded feature set could potentially improve detection accuracy.
- ✓ Experiment with different model architectures and parameters including number of membership functions, rules, and training iterations to optimize performance. Automated architecture search could discover the ideal layout.

Enhancing the model through these directions can progress the solution from an effective research prototype toward a production-ready system to identify and thwart real-world keylogging trojans and protect user data.

Bibliography

A. A. Ahmed and I. Traore, "A new biometric technology based on mouse dynamics," *IEEE Transactions on Dependable and Secure Computing,* vol. 4, no. 3, pp. 165-179, 2019.

A. A. Albassam, "A review of keylogger monitoring systems," *2018 1st International Conference on Computer Applications & Information Security (ICCAIS),* pp. 1-5, 2018.

A. A. E. Ahmed and I. Traore, "A new biometric technology based on mouse dynamics," *IEEE Transactions on dependable and secure computing,* vol. 4, no. 3, pp. 165-179, 2014.

A. A. Kadhim and M. A. Mohammed, "Generation of phishing dataset for anomaly detection systems," *International Journal of Interactive Multimedia & Artificial Intelligence,* vol. 5, no. 7, 2019.

A. Abdeltwab, M. Mahmoud, X. Liu, M. Nouretdinov, and L. Cavallaro, “Neuro-fuzzy learning adversarial poisoning attacks,” 2022. doi: 10.48550/arXiv.2212.11708

A. Arkhipova and P. Polyakov, “Methodology for constructing a neural fuzzy network in the field of information security,” *Digital technology security*, no. 3, pp. 43–56, 2021. doi:10.17212/2782-2230-2021-3-43-56

A. Boroujerdian, S. Hashemi, and A. T. Haghighat, "Intrusion detection in encrypted network traffic using artificial neural network," in *International Conference on Wireless Networks and Mobile Communications*, 2019.

A. Branitskiy and I. Kotenko, "Network attack detection based on combination of neural, immune and neuro-fuzzy classifiers," *2015 IEEE 18th International Conference on Computational Science and Engineering*, 2015. doi:10.1109/cse.2015.26

A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," arXiv preprint arXiv:1810.00069, 2018.

A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time based features." *ICISSP*, 2019.

A. Khalid, J. Yu, E. Nagowah, and R. Hayee, "An efficient two stage malware detection method using neuro-fuzzy classifier," in *International Carnahan Conference on Security Technology*, 2020.

A. Kharraz, S. Arshad, C. Mulliner, W. K. Robertson, and E. Kirda, "UNVEIL: A large-scale, automated approach to detecting ransomware," *25th USENIX Security Symposium, pp. 757-772*. 2015

A. Khomutenko* et al., "Tools of the neuro-fuzzy model of Information Risk Management in national security," *International Journal of Engineering and Advanced Technology,* vol. 8, no. 6, pp. 4526–4530, 2019. doi:10.35940/ijeat.f8842.088619

A. M. Azab, M. Eltoweissy, and M. Alazab, "Towards understanding malware evasion from dynamic analysis: A case study," *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS),* pp. 1-5, 2018.

A. Obaid, M. Abdu, J. W. Kim, Y. Chang, and T. B. Kim, "User identification approach based on firefly-harmony search optimized keystroke dynamic authentication system," *Applied Sciences,* vol. 9, no. 20, 2019.

A. S. Boroujerdi and S. Ayat, "A robust ensemble of neuro-fuzzy classifiers for ddos attack detection," *Proceedings of 2013 3rd International Conference on Computer Science and Network Technology*, 2013. doi:10.1109/iccsnt.2013.6967159

A. S. Bozkir, E. Tahillioglu, M. Aydos, and I. Kara, "Catch them alive: A malware detection approach through memory forensics, Manifold Learning and Computer Vision," Computers & Security, vol. 103, p. 102166, 2021. [Online]. Available: https://doi.org/10.1016/j.cose.2020.102166

A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams," in Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

Ayo, F. E., Folorunso, S. O., Abayomi-Alli, A. A., Adekunle, A. O., & Awotunde, J. B. (2020). Network intrusion detection based on deep learning model optimized with rule-based hybrid feature selection. Information Security Journal: A Global Perspective, 29(6), 267-283.

B. Alsulami, M. N. Dailey, and A. El Saddik, "Detecting Insider Threats: A Survey of Data-Driven Approaches," IEEE Access, vol. 10, pp. 44795-44819, 2022.

B. Biggio and F. Roli, “Wild patterns: Ten years after the rise of adversarial machine learning,” Pattern Recognition, vol. 84, pp. 317–331, Dec. 2018.

B. Sun, Y. Chen, C. C. Loy, and X. Tang, "Robust 3D keyboard stroke biometrics using deep learning," Pattern Recognition, vol. 93, pp. 86-98, 2019.

C. Igwenagu. (2016). Fundamentals of research methodology and data collection.

C. Pham, L. A. Nguyen, N. H. Tran, E.-N. Huh, and C. S. Hong, “Phishing-aware: A neuro-fuzzy approach for anti-phishing on Fog Networks,” *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 1076–1089, 2018. doi:10.1109/tnsm.2018.2831197

C. Pham, L. A. Nguyen, N. H. Tran, E.-N. Huh, and C. S. Hong, “Phishing-aware: A neuro-fuzzy approach for anti-phishing on Fog Networks,” *IEEE Transactions on Network and Service Management,* vol. 15, no. 3, pp. 1076–1089, 2018. doi:10.1109/tnsm.2018.2831197

C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, “Adversarial examples for semantic segmentation and object detection,” in *IEEE International Conference on Computer Vision,* 2017. doi: 10.1109/ICCV.2017.491

C. Yin, Y. Zhu, J. Fei, and X. He, “A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks,” IEEE Access, vol. 5, pp. 21954–21961, 2017.

Chaudhary, Alka, Vinita Tiwari and Anil Kumar. “NEURO-FUZZY BASED INTRUSION DETECTION SYSTEMS FOR NETWORK SECURITY.” *Journal of Global Research*

in Computer Sciences 5 (2014): 1-2. 2014. Available: https://www.rroij.com/open-access/neurofuzzy-based-intrusion-detection-systems-for-network-security-1-2.pdf

Claire Selltiz and others, (1962), Research Methods in Social Sciences, New York: Holt, Rinehart and Winston: Published for the Society for the Psychological Study of Social Issue

D. Gupta, A. Yadav, and A. K. Bhurtel, "Keylogging: A key to cybercrime," *Current Science,* vol. 112, no. 9, pp., 2022.

D. J. Smith, M. Kitchen, M. Popenici, O. Sudit, R. Torres, D. Tratz-Ryan, M. Venkata, J. Ward, T. Wilson, et al., "Evolving multilayered cyber defenses through an adapted NEAT algorithm," in Military Communications Conference, *MILCOM 2018-2018 IEEE*, pp. 607-612, *IEEE,* 2018.

D. J. Smith, O. Sudit, M. Kitchen, R. Torres, D. Tratz-Ryan, M. Venkata, and J. Ward, "Evolving multilayered cyber defenses through an adapted neural network," *in Military Communications Conference,* 2019.

D. K. Pour, A. R. Grégio, and Y. Degeilh, "A survey on keylogging in modern keyboards," *IEEE Access,* vol. 6, pp. 41909-41930, 2018.

D. Kwon, M. Liu, and H. Kim, "A novel network intrusion detection model based on recurrent neural networks," in *Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling (APARM),* 2020.

D. N. Hoang, P. N. Pathirana, A. Seneviratne, and Y. Li, “Keystroke data augmentation using generative adversarial networks with conditional vector arithmetic," *IEEE Access*, vol. 8, pp. 211463-211475, 2020.

D. Parfenov, I. Bolodurina, L. Zabrodina and A. Zhigalov, "Development of a solution for identifying network attacks based on adaptive neuro-fuzzy networks ANFIS," *2021 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), Yekaterinburg, Russia,* 2021, pp. 0491-0495, doi: 10.1109/USBEREIT51232.2021.9455115.

D. Pillai and I. Siddavatam, "A modified framework to detect keyloggers using machine learning algorithm," *International Journal of Information Technology,* vol. 11, no. 4, pp. 707–712, 2018. [Online]. Available: https://doi.org/10.1007/s41870-018-0237-6

D. Polemi, G. Rigas, and G. Loukas, "Human authentication using mouse dynamics," *Human-centric Computing and Information Sciences*, vol. 9, no. 1, pp. 1-21, 2019.

D. Villani, C. Tappert, G. Ngo, J. Simone, R. St. Fort, and S. Cha, "Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions," *2018 IEEE Conference on Communications and Network Security (CNS),* pp. 1-9, 2018. http://dx.doi.org/10.1109/CVPRW.2006.115

Darus, M. Y., Azizi, M., & Ariffin, M. (2022). Enhancement Keylogger Application for Parental Control and Monitor Children ’ s Activities. Journal of Positive School Psychology, 6(3), 8482–8492.

Dubberly, H. (2016) What is systems design?, Dubberly Design Office. Available at: https://www.dubberly.com/articles/what-is-systems-design.html (Accessed: 20 October 2023)

E. Redmiles, S. Kross, and M. L. Mazurek, "How well do my results generalize? comparing security and privacy survey results from mturk, web, and telephone samples," in *IEEE Symposium on Security and Privacy,* 2019.

E. Toreini, M. Burnap, O. Firpo-Triplett, and G. J. Awan, "Detecting cyberbullying in the presence of emojis: a comparison of deep learning and machine learning approaches," in *International Conference on Recent Advances in Natural Language Processing,* 2019.

Emerging Technologies 2004.

F. Ahmed, A. Ali, V. S. Sheng, W. Li, A. Aziz, and A. U. R. Khan, "Efficient deep learning model for cyber-attack detection," IEEE Access, vol. 7, pp. 42402-42414, 2019.

F. E. AYO, J. AWOTUNDE, S. Folorunso, and O. A. Olalekan, "Cafiskld: A combinatorial-based fuzzy inference system for keylogger detection," *SSRN Electronic Journal,* 2022. doi:10.2139/ssrn.4111802

F. Li, N. L. Clarke, M. Papadaki, and P. S. Dowland, "Active authentication for mobile devices utilising behaviour profiling and anomalies detection," *International Journal of Information Security* 13, pages 229–244 (2014). https://doi.org/10.1007/s10207-013-0209-6

G. Almashaqbeh, I. Alsmadi, N. A. Noman, R. Al-Shalabi, and J. Arshad, "Cybercrime detection model using optimized neural networks and fuzzy logic," Technologies, vol. 10, no. 1, pp. 2, 2022.

G. Katz, C. Barrett, R. Dillabough, M. Lustgarten, O. Farooq, and S. Botzer, "Analyzing linguistic style—the key to detecting financial manipulation and fraud," Leland Stanford Junior University Working Paper, 2019.

G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "UWS-Network traffic dataset for evaluating intrusion detection systems," Data in brief, vol. 26, 2019.

I. Alabdulwahhab. "Detecting hardware keyloggers using current flow analysis". *IEEE Access,* 6, 11309-11320. 2018

I. Özbek, B. Gürbüz, and A. E. Çetin, "Keylogging detection at character level from online handwriting," *IET Biometrics,* 2020.

IEEE Access, vol. 9, pp. 15166-15185, 2021. doi: 10.1109/ACCESS.2021.3053489

Igwenagu, Chinelo. (2016). Fundamentals of research methodology and data collection.

J. E. Ekberg, S. Bugiel, and A. Rajendran, "TrustZone explained: Architectural features and use cases," *2018 IEEE 2nd International Verification and Security Workshop (IVSW),* pp. 35-42, 2018.

J. Stewart, C. V. Lopes, and A. K. Jain, "Key sequence rhythms as biometric modalities," *IEEE Transactions on Information Forensics and Security*, 2019.

J. V. Monaco, "New dimensions of insider threat: keylogging malware and countermeasures in the modern work environment," *Journal of Applied Security Research*, vol. 13, no. 4, pp. 425-435, 2018.

J. V. Monaco, C. C. Tappert, and S.-H. Cha, "User authentication via adaptive typing behaviors with fuzzified timing intervals," *IEEE Systems Journal*, 2019.

J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining,* pp. 470-478, 2014. https://www.jmlr.org/papers/volume7/kolter06a/kolter06a.pdf

K. A. Calix, S. Manyam, and G. R. Knapp, "Stylometry and keystroke analysis for cyberthreat assessment," in CyCon U.S., 2019.

K. Duru and F. Canbek, "Improving accuracy rate of keystroke dynamics used in user authentication through error correction process," *Neural Computing and Applications*, 2019.

K. Howell. (2013). An Introduction to the Philosophy of Methodology. 10.4135/9781473957633.

K. V. Pradeepthi and A. Kannan, "Detection of Botnet traffic by using Neuro-fuzzy based Intrusion Detection," *2018 Tenth International Conference on Advanced Computing (ICoAC), Chennai, India*, 2018, pp. 118-123, doi: 10.1109/ICoAC44903.2018.8939109.

L. Chen, J. Bao, and K. S. Xu, "Adaptive trust in human control of swarm systems using neural-fuzzy Reinforcement Learning," *IEEE Systems Journal,* 2020.

L. Chen, J. Bao, and K. S. Xu, "Dynamic user profiling for masquerade detection using recursive neural networks," in *International Joint Conference on Neural Networks*, 2019.

L. Sun, R. Adhikari, and Z. Zhang, "Applying deep learning to predict cyber security incidents," in *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*, 2020, pp. 180-187, 2020.

M. Alali, A. Almogren, M. M. Hassan, I. A. L. Rassan, and M. Z. Bhuiyan, "Improving risk assessment model of cyber security using Fuzzy Logic Inference System," *Computers & Security,* vol. 74, pp. 323–339, 2018. doi:https://doi.org/doi:10.1016/j.cose.2017.09.011

M. Alauthman, N. Aslam, M. Al-Kasassbeh, E. Khan, and A. Al-Qerem, "An efficient online network intrusion prediction model using discrete sequential pattern mining," *Journal of Ambient Intelligence and Humanized Computing,* pp. 1-21, 2019.

M. Curtin, C. Tappert, M. Villani, G. Ngo, J. Simone, H. Fort St, and S. Cha, "Keystroke biometric recognition on long-text input: A feasibility study," in *International Conference on Biometrics (ICB),* 2019.

M. Curtin, C. Tappert, M. Villani, G. Ngo, J. Simone, H. St Fort, and S. Cha, "Keystroke biometric recognition on long-text input: A feasibility study," *International Conference on Biometrics,* 2019.

M. Han, Y. Khan, M. Han, Z. Li, and T. Zhao, "A robust payment fraud detection framework using gradient boosted decision trees, neighborhood components analysis, and genetic algorithm," *Applied Sciences,* vol. 10, no. 6, 2020.

M. Laris, D. Liu, and R. Bush, "Game theoretic modeling of pilot behavior during mid-air encounters," *AIAA/IEEE Digital Avionics Systems Conference-Proceedings,* vol. 2019, 2019.

M. Pratama, R. Hartono, S. Fauziati and J. H. Kim, "Neuro-Fuzzy waterfall model for software development project investment decision," *International Journal of Fuzzy Systems*, vol. 22, no. 1, pp. 13-22, Feb. 2020.

M. Raya, L. G. Sison, and J. DeBlasio, "Hardware keylogger implementation searches," *IEEE Transactions on Education,* vol. 62, no. 2, pp. 75-83, 2018.

M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *ACM European Workshop on Machine Learning and Systems Security (EuroMLSec),* 2019.

MacMillan, J.H. and Schumacher, S. (2001) Research in Education. A Conceptual Introduction. 5th Edition, Longman, Boston.

Mishra, Dr. Shanti Bhushan & Alok, Dr. Shashi. (2017). HANDBOOK OF RESEARCH METHODOLOGY.

Muhammad Aslam, Rana Naveed Idrees, Mirza Muzammil Baig, and Muhammad Asif Arshad, "Anti-Hook Shield against the Software Key Loggers", National Conference on

N. Akhtar and A. S. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410-14430, 2018. doi: 10.1109/ACCESS.2018.2807385

N. Akhtar, M. A. Ghazanfar, M. A. Azam, A. U. Rehman, and T. Saba, "An intelligent anomaly prediction system against computer keylogging using optimized ensemble models and fuzzy logic," *Computer Standards & Interfaces,* vol. 80, 2022.

N. Akhtar, M. M. Fraz, and S. O. Gilani, "COVID-19: An adversarial viewpoint,"

N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *IEEE Symposium on Security and Privacy,* 2017. doi: 10.1109/SP.2017.49

N. Le, Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," arXiv preprint arXiv:1804.04159, 2020.

N. Le, Q. Niyaz, W. Sun, and A. Y. Javaid, “A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN),” arXiv [cs.NI], Apr. 2018.

N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” in *IEEE Symposium on Security and Privacy, 2016*. doi: 10.1109/SP.2016.41

N. Tavabi, M. Bartley, A. Abeliuk, E. Ferrara, and K. Lerman, "Characterizing bots' reactions to policy changes: Evidence from twitter bots during U.S. presidential election," *International AAAI Conference on Web and social media,* 2019.

O. Belej and L. Halkiv, “Development of a network attack detection system based on hybrid neuro-fuzzy algorithms,” *Computer Modeling and Intelligent Systems,* vol. 2608, pp. 926–938, 2020. doi:10.32782/cmis/2608-69

O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Efficient machine learning for big data: A review*," Big Data Research*, vol. 18, pp. 28-45, 2019.

P. Ashwini and Dr. Vadivelan N, “Security from phishing attack on internet using evolving Fuzzy Neural Network*,” CVR Journal of Science & Technology*, vol. 20, no. 1, pp. 50–55, 2021. doi:10.32377/cvrjst2007

P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in Proceedings, vol. 89. *Presses universitaires de Louvain,* 2019, pp. 89-94.

P. V. de Campos Souza, "Fuzzy neural networks and Neuro-Fuzzy Networks: A review of the main techniques and applications used in the literature," *Applied Soft Computing*, vol. 92, p. 106275, 2020. [Online]. Available: https://doi.org/10.1016/j.asoc.2020.106275

Proactive, M., Suitable, I., Enterprises, C., Sun, J., Liu, C., & Yuan, H. (2021). Keylogger Detection and Prevention Journal of Physics: Conference Series, 2007(1). https://doi.org/10.1088/1742-6596/2007/1/012005

Q. V. Pham, D. C. Nguyen, H. Huynh, and W. K. Ng, "Generalization efficacy safety-critical deep neural network: A literature review," *Electronics,* vol. 9, no. 7, p. 1074, 2020.

Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology,* 2019.

R. Anand and S. Dhande, "Detection and prevention of SQL injection attack using ANFIS," 2019 *3rd International conference on electronics, communication and aerospace technology (ICECA),* vol. 2, pp. 12-16, 2019.

R. Bello, S. Zeadally, and M. Badra, "Network intrusion detection system for IoT cybersecurity based on learning techniques," IEEE Internet of Things Journal, vol. 7, no. 11, pp. 10280-10289, 2019.

R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *26th USENIX Security Symposium,* 2017.

R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *USENIX Security Symposium,* 2019.

R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *IEEE Symposium on Security and Privacy*, 2010.

Royo, Á. A., Rubio, M. S., Fuertes, W., Cuervo, M. C., Estrada, C. A., & Toulkeridis, T. (2021). Malware Security Evasion Techniques: An Original Keylogger Implementation. In World Conference on Information Systems and Technologies (pp. 375-384). Springer, Cham.

S. A. Haruna, R. O. Akinyede, and B. O. Kehinde, "Neuro-fuzzy data mining system for identifying e-commerce related threats," *MALAYSIAN JOURNAL OF COMPUTING*, vol. 5, no. 2, p. 537, 2020. doi:10.24191/mjoc.v5i2.8642

S. A. Khan, M. Shafique, and J. Henkel, "Hardware security leveraging process variation for resiliency against costly attacks," in *International Conference on Computer-Aided Design,* 2019.

S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, vol. 25, pp. 152-160, 2019.

S. Choudhary, A. Bhardwaj, A. Kumar, and A. K. Dumka, "Detection of network traffic anomalies using machine learning techniques," in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pp. 945–949, 2018

S. H. Almotiri, "Integrated Fuzzy Based Computational Mechanism for the Selection of Effective Malicious Traffic Detection Approach," in *IEEE Access,* vol. 9, pp. 10751-10764, 2021, doi: 10.1109/ACCESS.2021.3050420.

S. J. Han and S. B. Cho, "Evolutionary neural networks for anomaly detection based on the behavior of a program," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 36, no. 3, pp. 559-570, June 2006.

S. M. Alghamdi, E. S. Othathi, and B. S. Alsulami, "Detect keyloggers by using machine learning," *2022 Fifth National Conference of Saudi Computers Colleges (NCCC),* 2022. doi:10.1109/nccc57165.2022.10067780

S. Manoj, K. Elango, S. Kadry, M. S. F. Al Sharqi, and K. K. Mohammed, "Adaptive multilayer perceptron network intrusion detection system using fuzzy logic," *Soft Computing,* 2020.

S. R. Chhetri, R. Candell, and T. Zander, "Towards forensic-aware software defined networking to investigate mirai-based IoT botnet attacks," *IEEE Conference on Communications and Network Security*, 2019.

Saunders, M. N. K., Lewis, P., & Thornhill, A. (2019). "Research Methods for Business Students" Chapter 4: Understanding research philosophy and approaches to theory development. In Researchgate.Net (Issue January).

Schmidt, E. (2020). Methods and Methodology. 77–94. https://doi.org/10.1007/978-3-658-28540-1_3

Singh, A., Choudhary, P., Singh, A. K., & Tyagi, D. K. (2021). Keylogger Detection and Prevention. Journal of Physics: Conference Series, 2007, 012005. DOI: 10.1088/1742-6596/2007/1/012005.

T. Halevi and N. Saxena, "Detecting mobile malware threats to home computer networks," *IEEE Transactions on Mobile Computing,* vol. 12, no. 11, pp. 2291-2305, 2013.

T. Wang, Z. Zeng, S. Liu, L. Chen, R. Hou, M. Cai, M. Li, F. Yang, M. Tang, Y. Jiang, J. Wang, J. Li, Z. Ni, Z. Wang, H. Wang, and Z. Gao, "A survey of system logs based anomaly detection," *Journal of Communications and Information Networks*, vol. 6, no. 4, pp. 506–519, 2021.

V. Dhakal, M. K. C. Nelaturi, and H. Barringer, "Keystroke data quality: Characterizing user keystrokes robustly and accurately," in *International Conference on Biometrics (ICB),* 2019.

W. Li, P. Lu, D. Zou, and P. Liang, "Adversarial examples for a neuro-fuzzy system," *IEEE Transactions on Fuzzy Systems,* 2021. doi: 10.1109/TFUZZ.2021.3116092

W. Lin, Q. Wen, J. Zhang, X. Huang, and B. Xie, "An intrusion prediction framework based on robust principal component analysis and sequential pattern mining," *International Journal of Distributed Sensor Networks,* vol. 16, no. 4, 2020.

W. Yang, J. Li, Y. Wang, Q. A. Chen, S. S. M. Chow, and Z. Liu, "Show me the money! finding flawed implementations of third-party in-app payment in android apps," *21st Annual Network and Distributed System Security Symposium, NDSS 2018, 18-21 February, San Diego, California, USA,* 2018\

X. Li and X. Liang, “Adversarial examples detection in deep networks with convolutional filter statistics,” in *IEEE International Conference on Computer Vision*, 2017. doi: 10.1109/ICCV.2017.490

X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems,* vol. 30, no. 9, pp. 2805-2824, 2019.

Y. Balakrishnan and P. N. Renjith, "An analysis on Keylogger Attack and Detection based on Machine Learning," *2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF), Chennai, India*, 2023, pp. 1-8, doi: 10.1109/ICECONF57129.2023.10083937.

Y. Ding, J. Xu, Y. Zeng, and X. Chen, “Neural Malware Detection Based on Image Representation of Malware Command Sequences,” in 2019 *IEEE 27th International Conference on Network Protocols (ICNP)*, pp. 1–6, 2019

Y. Lestari, E. I. Azhari, and B. Istiyanto, “Detection of keylogger malware using static analysis method,” CommIT Journal, vol. 16, no. 1, pp. 9-14, 2022.

Y. Wang, Z. Sun, and Y. Han, “Network Attack Path Prediction Based on Vulnerability Data and Knowledge Graph,” *International Journal of Innovative Computing, Information and Control,* vol. 17, no. 5, pp. 1717–1730, Oct. 2021. doi: http://ijicic.org/ijicic-170518.pdf

Y. Wu, S. Kuo, W. Liu, and C. Lin, "An interpretable fuzzy rule-based system for tackle football play prediction and explanation*," IEEE Access*, vol. 7, pp. 84473-84484, 2019.

Y.-T. Wu, S. Kuo, M. Ho, W. Wu, W. Liu, and C. J. Lin, "An interpretable fuzzy rule-based system for tackle football play prediction and explanation," IEEE Access, vol. 7, pp. 84473-84484, 2019.

Z. Aziz, "Adaptive network security management using fuzzy cognitive maps," *International Joint Conference on Neural Networks,* 2019.

Z. Jan, I. Ullah, M. Alam, M. Usman, and K. Khan, "Enhancing network security through meremotic analysis using machine learning techniques," *IEEE Access,* vol. 8, pp. 214892-214910, 2020.

Z. Wang, W. Yan, and T. Oates, "Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline," in 2017 *International joint conference on neural networks (IJCNN)*, pp. 1578–1585. 2017.

Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: High-speed covert channel attacks in the cloud," *21st USENIX Security Symposium, pp*. 159-173, 2018.

Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-Sec: deep learning in android malware detection," in ACM SIGCOMM Computer Communication Review, vol. 44, no. 4, pp. 371–372, 2014.

Z. Zhang, C. Ma, A. K. Sangaiah, and W. Zhuang, "User recognition based on mouse movement for smart home environment," *Pervasive and Mobile Computing*, vol. 59, 2019.

List of Appendices

(CODE LISTING)

Input data show below

 # importing the data set

resultData<- read.csv(file = 'C:/Users/user/Desktop/business/Ayobami/archive/Keylogger_DetectiondataSlice.csv')

```
#checking for missing values
sum(is.na(resultData))
#removing row with missing values
newData<- na.omit(resultData)
# removing the first three column that is not needed for the prediction
newData.variable<-newData
newData.variable <- newData[1:nrow(newData), 4:43]
# picking the data to be used as input and ignoring the last to be used as output
new.data.input <- newData.variable[1:nrow(newData.variable), 1:39]
#splitting the data into training and test data
library(caTools)
split<- sample.split(newData.variable, SplitRatio =0.7)
split
train<- subset(newData.variable, split=="TRUE")
test.data<-subset(newData.variable, split=="FALSE")
#storing the test data in another variable for easy analysis
tst.spliting<-test.data

# Dplyr removing the last column which is the output for the data for testing:
test.input=select(tst.spliting, -ClassAttack)
#showing only the output to specialize on
test.output <- matrix(test.data[1:nrow(test.data), 40], ncol = 1)
# method used for the prediction
```

```
control <- list( max.iter =100, step.size = 0.1, type.tnorm = "MIN", type.snorm = "MAX",
            type.defuz = "COG", type.implication.func = "ZADEH", name = "Ayobami")
object.train <- frbs.learn(train, range.data = NULL, method.type = c("HYFIS"),
                    control)
# Dplyr removing the last column which is the output for the data for testing:
#range.data=select(range.data, -40)
pred <- predict(object.train, test.input)#prediction of the result
# rounding off the ped values to the nearest whole number
pred=floor(pred)


# Calculating the error of the prediction using the the MEAN SQUARED ERROR(MSE)
#  The predicted values are saved as a matrix.
#They can be compared with the actual values using, e.g., the mean squared error (MSE)
err.MSE <- mean((test.output - pred)^2)
print(err.MSE)
# comparing the oringinal output with the predicted output
my_data <- data.frame(test.output, pred)    # Apply data.frame function
my_data                         # Print data frames
```

APPENDIX II

(OBJECT LISTING)

Output data show below

```
source('C:/Users/user/Desktop/R projectWorks/ayobamiImplementation.R', echo=TRUE)
> # importing the data set
```

> resultData<- read.csv(file = 'C:/Users/user/Desktop/business/Ayobami/archive/Keylogger_DetectiondataSlice.csv')

> #checking for missing values

> sum(is.na(resultData))

[1] 18176258

> #removing row with missing values

> newData<- na.omit(resultData)

> # removing the first three column that is not needed for the prediction

> newData.variable<-newData

> newData.variable <- newData[1:nrow(newData), 4:43]

> # picking the data to be used as input and ignoring the last to be used as output

> new.data.input <- newData.variable[1:nrow(newData.variable), 1:3 .... [TRUNCATED]

> #splitting the data into training and test data

> library(caTools)

> split<- sample.split(newData.variable, SplitRatio =0.7)

> split

[1] FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE TRUE TRUE FALSE TRUE

[15] TRUE TRUE FALSE TRUE FALSE TRUE TRUE TRUE FALSE TRUE FALSE TRUE TRUE FALSE

[29] TRUE FALSE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE

```
> train<- subset(newData.variable, split=="TRUE")

> test.data<-subset(newData.variable, split=="FALSE")

> #storing the test data in another variable for easy analysis

> tst.spliting<-test.data

> # Dplyr removing the last column which is the output for the data for testing:

> test.input=select(tst.spliting, -ClassAttack)

> #showing only the output to specialize on

> test.output <- matrix(test.data[1:nrow(test.data), 40], ncol = 1)

> # method used for the prediction

>

> control <- list( max.iter =100, step.size = 0.1, type.tnorm = "MIN", type.snorm = "MAX",

+              t .... [TRUNCATED]
```

> object.train <- frbs.learn(train, range.data = NULL, method.type = c("HYFIS"),

+ control)

**|===
==========| 100%**

> # Dplyr removing the last column which is the output for the data for testing:

> #range.data=select(range.data, -40)

>

> pred <- predict(object.tra .... [TRUNCATED]

> # rounding off the ped values to the nearest whole number

> pred=floor(pred)

> # Calculating the error of the prediction using the the MEAN SQUARED ERROR(MSE)

> # The predicted values are saved as a matrix.

> #They can be comp .... [TRUNCATED]

> print(err.MSE)

[1] 0.3783784

> # comparing the oringinal output with the predicted output

> my_data <- data.frame(test.output, pred) # Apply data.frame function

> my_data # Print data frame

	test.output	pred
1	1	1
2	1	1
3	1	1
4	2	1
5	2	1
6	1	1
7	2	1
8	1	1
9	1	1
10	2	1
11	2	2
12	1	1
13	1	1
14	1	1
15	1	1
16	2	1
17	1	1
18	1	1
19	2	1
20	1	1
21	2	1
22	1	1

23 1 1

24 2 1

25 2 1

26 2 1

27 1 1

28 2 1

29 1 1

30 1 1

31 2 1

32 1 1

33 2 1

34 1 1

35 1 1

36 1 1

37 2 1

> summary(object.train)

The name of model: Ayobami

Model was trained using: HYFIS

The names of attributes: Total.Fwd.Packets Total.Backward.Packets Fwd.Packet.Length.Min Bwd.Packet.Length.Min Flow.IAT.Min Bwd.IAT.Min Fwd.PSH.Flags Bwd.PSH.Flags Fwd.URG.Flags Bwd.URG.Flags Fwd.Header.Length FIN.Flag.Count SYN.Flag.Count RST.Flag.Count PSH.Flag.Count ACK.Flag.Count URG.Flag.Count CWE.Flag.Count ECE.Flag.Count Down.Up.Ratio Fwd.Header.Length.1 Fwd.Avg.Bytes.Bulk Fwd.Avg.Packets.Bulk Fwd.Avg.Bulk.Rate Bwd.Avg.Bytes.Bulk Bwd.Avg.Packets.Bulk Bwd.Avg.Bulk.Rate Subflow.Fwd.Packets Subflow.Fwd.Bytes Subflow.Bwd.Packets act_data_pkt_fwd min_seg_size_forward Protocol Bwd.IAT.Total Bwd.IAT.Std Bwd.IAT.Max Avg.Bwd.Segment.Size Max.Packet.Length Avg.Fwd.Segment.Size ClassAttack

The interval of training data:

	Total.Fwd.Packets	Total.Backward.Packets	Fwd.Packet.Length.Min
min	1	0	0
max	121	203	116

	Bwd.Packet.Length.Min	Flow.IAT.Min	Bwd.IAT.Min	Fwd.PSH.Flags	Bwd.PSH.Flags
min	0	4	0	0	0
max	116	7722	161	117	116

	Fwd.URG.Flags	Bwd.URG.Flags	Fwd.Header.Length	FIN.Flag.Count	SYN.Flag.Count
min	0	0	32	0	0
max	116	116	3272	116	117

	RST.Flag.Count	PSH.Flag.Count	ACK.Flag.Count	URG.Flag.Count	CWE.Flag.Count
min	0	0	0	0	0
max	116	116	117	116	116

	ECE.Flag.Count	Down.Up.Ratio	Fwd.Header.Length.1	Fwd.Avg.Bytes.Bulk
min	0	0	32	0
max	116	117	3272	116

	Fwd.Avg.Packets.Bulk	Fwd.Avg.Bulk.Rate	Bwd.Avg.Bytes.Bulk	Bwd.Avg.Packets.Bulk
min	0	0	0	0
max	116	116	116	116

	Bwd.Avg.Bulk.Rate	Subflow.Fwd.Packets	Subflow.Fwd.Bytes	Subflow.Bwd.Packets
min	0	1	0	0
max	116	121	1612	203

	act_data_pkt_fwd	min_seg_size_forward	Protocol	Bwd.IAT.Total	Bwd.IAT.Std

min 0 32 0 0 0

max 117 148 17 91106109 32018790

Bwd.IAT.Max Avg.Bwd.Segment.Size Max.Packet.Length Avg.Fwd.Segment.Size ClassAttack

min 0 0.0 0 0 1

max 90630474 1440.7 1460 1460 2

Type of FRBS model:

[1] "MAMDANI"

Type of membership functions:

[1] "GAUSSIAN"

Type of t-norm method:

[1] "Standard t-norm (min)"

Type of s-norm method:

[1] "Standard s-norm"

Type of defuzzification technique:

[1] "modified COG"

Type of implication function:

[1] "ZADEH"

The names of linguistic terms on the input variables:

[1] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[8] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[15] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[22] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[29] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[36] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[43] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[50] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[57] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[64] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[71] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[78] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[85] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[92] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[99] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[106] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[113] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[120] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[127] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[134] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[141] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[148] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[155] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[162] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[169] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[176] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[183] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[190] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[197] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[204] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[211] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[218] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[225] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[232] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[239] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[246] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[253] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[260] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

[267] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

The parameter values of membership function on the input variable (normalized):

vv.small v.small small medium large v.large vv.large

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

vv.small v.small small medium large v.large vv.large

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

vv.small v.small small medium large v.large vv.large

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

```
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
```

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

vv.small v.small small medium large v.large vv.large

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

vv.small v.small small medium large v.large vv.large

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

vv.small v.small small medium large v.large vv.large

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

vv.small v.small small medium large v.large vv.large

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

vv.small v.small small medium large v.large vv.large

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

vv.small v.small small medium large v.large vv.large

```
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small  v.small    small   medium    large  v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small  v.small    small   medium    large  v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small  v.small    small   medium    large  v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small  v.small    small   medium    large  v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small  v.small    small   medium    large  v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
```

```
      vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
```

```
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small   v.small     small    medium     large   v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small   v.small     small    medium     large   v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small   v.small     small    medium     large   v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small   v.small     small    medium     large   v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small   v.small     small    medium     large   v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
      vv.small   v.small     small    medium     large   v.large  vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
```

```
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000
[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000
[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333
       vv.small    v.small      small     medium      large    v.large   vv.large
```

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

vv.small v.small small medium large v.large vv.large

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

[reached getOption("max.print") -- omitted 2 rows]

The names of linguistic terms on the output variable:

[1] "vv.small" "v.small" "small" "medium" "large" "v.large" "vv.large"

The parameter values of membership function on the output variable (normalized):

vv.small v.small small medium large v.large vv.large

[1,] 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000 5.00000000

[2,] 0.00000000 0.16666667 0.33333333 0.50000000 0.66666667 0.83333333 1.00000000

[3,] 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333 0.05833333

[4,] NA NA NA NA NA NA NA

[5,] NA NA NA NA NA NA NA

The number of linguistic terms on each variables

Total.Fwd.Packets Total.Backward.Packets Fwd.Packet.Length.Min

[1,] 7 7 7

Bwd.Packet.Length.Min Flow.IAT.Min Bwd.IAT.Min Fwd.PSH.Flags Bwd.PSH.Flags

[1,] 7 7 7 7 7

Fwd.URG.Flags Bwd.URG.Flags Fwd.Header.Length FIN.Flag.Count SYN.Flag.Count

[1,] 7 7 7 7 7

RST.Flag.Count PSH.Flag.Count ACK.Flag.Count URG.Flag.Count CWE.Flag.Count

[1,] 7 7 7 7 7

ECE.Flag.Count Down.Up.Ratio Fwd.Header.Length.1 Fwd.Avg.Bytes.Bulk

[1,] 7 7 7 7

Fwd.Avg.Packets.Bulk Fwd.Avg.Bulk.Rate Bwd.Avg.Bytes.Bulk Bwd.Avg.Packets.Bulk

[1,] 7 7 7 7

Bwd.Avg.Bulk.Rate Subflow.Fwd.Packets Subflow.Fwd.Bytes Subflow.Bwd.Packets

[1,] 7 7 7 7

act_data_pkt_fwd min_seg_size_forward Protocol Bwd.IAT.Total Bwd.IAT.Std

[1,] 7 7 7 7 7

Bwd.IAT.Max Avg.Bwd.Segment.Size Max.Packet.Length Avg.Fwd.Segment.Size

[1,] 7 7 7 7

ClassAttack

[1,] 7

The fuzzy IF-THEN rules:

V1 V2 V3 V4 V5 V6 V7 V8 V9

1 IF Total.Fwd.Packets is vv.small and Total.Backward.Packets is vv.small and

2 IF Total.Fwd.Packets is vv.small and Total.Backward.Packets is vv.small and

3 IF Total.Fwd.Packets is vv.small and Total.Backward.Packets is vv.small and

4 IF Total.Fwd.Packets is v.large and Total.Backward.Packets is vv.large and

5 IF Total.Fwd.Packets is small and Total.Backward.Packets is v.small and

6 IF Total.Fwd.Packets is small and Total.Backward.Packets is v.small and

V10 V11 V12 V13 V14 V15 V16 V17

1 Fwd.Packet.Length.Min is vv.small and Bwd.Packet.Length.Min is vv.small and

2 Fwd.Packet.Length.Min is vv.small and Bwd.Packet.Length.Min is vv.small and

3 Fwd.Packet.Length.Min is vv.small and Bwd.Packet.Length.Min is vv.small and

4 Fwd.Packet.Length.Min is vv.small and Bwd.Packet.Length.Min is vv.small and

5 Fwd.Packet.Length.Min is small and Bwd.Packet.Length.Min is small and

6 Fwd.Packet.Length.Min is small and Bwd.Packet.Length.Min is small and

V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28

1 Flow.IAT.Min is vv.large and Bwd.IAT.Min is vv.small and Fwd.PSH.Flags is vv.small

2 Flow.IAT.Min is vv.small and Bwd.IAT.Min is vv.small and Fwd.PSH.Flags is vv.small

3 Flow.IAT.Min is small and Bwd.IAT.Min is vv.small and Fwd.PSH.Flags is vv.small

4 Flow.IAT.Min is vv.small and Bwd.IAT.Min is vv.small and Fwd.PSH.Flags is vv.small

5 Flow.IAT.Min is vv.small and Bwd.IAT.Min is medium and Fwd.PSH.Flags is small

6 Flow.IAT.Min is vv.small and Bwd.IAT.Min is medium and Fwd.PSH.Flags is small

V29 V30 V31 V32 V33 V34 V35 V36 V37 V38 V39

1 and Bwd.PSH.Flags is vv.small and Fwd.URG.Flags is vv.small and Bwd.URG.Flags is

2 and Bwd.PSH.Flags is vv.small and Fwd.URG.Flags is vv.small and Bwd.URG.Flags is

3 and Bwd.PSH.Flags is vv.small and Fwd.URG.Flags is vv.small and Bwd.URG.Flags is

4 and Bwd.PSH.Flags is vv.small and Fwd.URG.Flags is vv.small and Bwd.URG.Flags is

5 and Bwd.PSH.Flags is small and Fwd.URG.Flags is small and Bwd.URG.Flags is

6 and Bwd.PSH.Flags is small and Fwd.URG.Flags is small and Bwd.URG.Flags is

V40 V41 V42 V43 V44 V45 V46 V47 V48 V49

1 vv.small and Fwd.Header.Length is vv.small and FIN.Flag.Count is vv.small and

2 vv.small and Fwd.Header.Length is vv.small and FIN.Flag.Count is vv.small and

3 vv.small and Fwd.Header.Length is vv.small and FIN.Flag.Count is vv.small and

4 vv.small and Fwd.Header.Length is vv.large and FIN.Flag.Count is vv.small and

5 small and Fwd.Header.Length is vv.small and FIN.Flag.Count is small and

6 small and Fwd.Header.Length is vv.small and FIN.Flag.Count is small and

V50 V51 V52 V53 V54 V55 V56 V57 V58 V59

1 SYN.Flag.Count is vv.small and RST.Flag.Count is vv.small and PSH.Flag.Count is

2 SYN.Flag.Count is vv.small and RST.Flag.Count is vv.small and PSH.Flag.Count is

3 SYN.Flag.Count is vv.small and RST.Flag.Count is vv.small and PSH.Flag.Count is

4 SYN.Flag.Count is vv.small and RST.Flag.Count is vv.small and PSH.Flag.Count is

5 SYN.Flag.Count is small and RST.Flag.Count is small and PSH.Flag.Count is

6 SYN.Flag.Count is small and RST.Flag.Count is small and PSH.Flag.Count is

V60 V61 V62 V63 V64 V65 V66 V67 V68 V69

1 vv.small and ACK.Flag.Count is vv.small and URG.Flag.Count is vv.small and

2 vv.small and ACK.Flag.Count is vv.small and URG.Flag.Count is vv.small and

3 vv.small and ACK.Flag.Count is vv.small and URG.Flag.Count is vv.small and

4 vv.small and ACK.Flag.Count is vv.small and URG.Flag.Count is vv.small and

5 small and ACK.Flag.Count is small and URG.Flag.Count is small and

6 small and ACK.Flag.Count is small and URG.Flag.Count is small and

V70 V71 V72 V73 V74 V75 V76 V77 V78 V79

1 CWE.Flag.Count is vv.small and ECE.Flag.Count is vv.small and Down.Up.Ratio is

2 CWE.Flag.Count is vv.small and ECE.Flag.Count is vv.small and Down.Up.Ratio is

3 CWE.Flag.Count is vv.small and ECE.Flag.Count is vv.small and Down.Up.Ratio is

4 CWE.Flag.Count is vv.small and ECE.Flag.Count is vv.small and Down.Up.Ratio is

5 CWE.Flag.Count is small and ECE.Flag.Count is small and Down.Up.Ratio is

6 CWE.Flag.Count is small and ECE.Flag.Count is small and Down.Up.Ratio is

V80 V81 V82 V83 V84 V85 V86 V87 V88 V89

1 vv.small and Fwd.Header.Length.1 is vv.small and Fwd.Avg.Bytes.Bulk is vv.small and

2 vv.small and Fwd.Header.Length.1 is vv.small and Fwd.Avg.Bytes.Bulk is vv.small and

3 vv.small and Fwd.Header.Length.1 is vv.small and Fwd.Avg.Bytes.Bulk is vv.small and

4 vv.small and Fwd.Header.Length.1 is vv.large and Fwd.Avg.Bytes.Bulk is vv.small and

5 small and Fwd.Header.Length.1 is vv.small and Fwd.Avg.Bytes.Bulk is small and

6 small and Fwd.Header.Length.1 is vv.small and Fwd.Avg.Bytes.Bulk is small and

V90 V91 V92 V93 V94 V95 V96 V97

1 Fwd.Avg.Packets.Bulk is vv.small and Fwd.Avg.Bulk.Rate is vv.small and

2 Fwd.Avg.Packets.Bulk is vv.small and Fwd.Avg.Bulk.Rate is vv.small and

3 Fwd.Avg.Packets.Bulk is vv.small and Fwd.Avg.Bulk.Rate is vv.small and

4 Fwd.Avg.Packets.Bulk is vv.small and Fwd.Avg.Bulk.Rate is vv.small and

5 Fwd.Avg.Packets.Bulk is small and Fwd.Avg.Bulk.Rate is small and

6 Fwd.Avg.Packets.Bulk is small and Fwd.Avg.Bulk.Rate is small and

V98 V99 V100 V101 V102 V103 V104 V105

1 Bwd.Avg.Bytes.Bulk is vv.small and Bwd.Avg.Packets.Bulk is vv.small and

2 Bwd.Avg.Bytes.Bulk is vv.small and Bwd.Avg.Packets.Bulk is vv.small and

3 Bwd.Avg.Bytes.Bulk is vv.small and Bwd.Avg.Packets.Bulk is vv.small and

4 Bwd.Avg.Bytes.Bulk is vv.small and Bwd.Avg.Packets.Bulk is vv.small and

5 Bwd.Avg.Bytes.Bulk is small and Bwd.Avg.Packets.Bulk is small and

6 Bwd.Avg.Bytes.Bulk is small and Bwd.Avg.Packets.Bulk is small and

V106 V107 V108 V109 V110 V111 V112 V113

1 Bwd.Avg.Bulk.Rate is vv.small and Subflow.Fwd.Packets is vv.small and

2 Bwd.Avg.Bulk.Rate is vv.small and Subflow.Fwd.Packets is vv.small and

3 Bwd.Avg.Bulk.Rate is vv.small and Subflow.Fwd.Packets is vv.small and

4 Bwd.Avg.Bulk.Rate is vv.small and Subflow.Fwd.Packets is v.large and

5 Bwd.Avg.Bulk.Rate is small and Subflow.Fwd.Packets is small and

6 Bwd.Avg.Bulk.Rate is small and Subflow.Fwd.Packets is small and

V114 V115 V116 V117 V118 V119 V120 V121

1 Subflow.Fwd.Bytes is vv.small and Subflow.Bwd.Packets is vv.small and

2 Subflow.Fwd.Bytes is vv.small and Subflow.Bwd.Packets is vv.small and

3 Subflow.Fwd.Bytes is vv.small and Subflow.Bwd.Packets is vv.small and

4 Subflow.Fwd.Bytes is medium and Subflow.Bwd.Packets is vv.large and

5 Subflow.Fwd.Bytes is vv.large and Subflow.Bwd.Packets is v.small and

6 Subflow.Fwd.Bytes is vv.large and Subflow.Bwd.Packets is v.small and

V122 V123 V124 V125 V126 V127 V128 V129 V130

1 act_data_pkt_fwd is vv.small and min_seg_size_forward is vv.small and Protocol

2 act_data_pkt_fwd is vv.small and min_seg_size_forward is vv.small and Protocol

3 act_data_pkt_fwd is vv.small and min_seg_size_forward is vv.small and Protocol

4 act_data_pkt_fwd is vv.small and min_seg_size_forward is vv.small and Protocol

5 act_data_pkt_fwd is small and min_seg_size_forward is small and Protocol

6 act_data_pkt_fwd is small and min_seg_size_forward is small and Protocol

V131 V132 V133 V134 V135 V136 V137 V138 V139 V140 V141

1 is small and Bwd.IAT.Total is vv.small and Bwd.IAT.Std is vv.small and

2 is small and Bwd.IAT.Total is vv.small and Bwd.IAT.Std is vv.small and

3 is small and Bwd.IAT.Total is vv.small and Bwd.IAT.Std is vv.small and

4 is small and Bwd.IAT.Total is vv.small and Bwd.IAT.Std is vv.small and

5 is small and Bwd.IAT.Total is vv.small and Bwd.IAT.Std is vv.small and

6 is small and Bwd.IAT.Total is vv.small and Bwd.IAT.Std is vv.small and

V142 V143 V144 V145 V146 V147 V148 V149

1 Bwd.IAT.Max is vv.small and Avg.Bwd.Segment.Size is vv.small and

2 Bwd.IAT.Max is vv.small and Avg.Bwd.Segment.Size is vv.small and

3 Bwd.IAT.Max is vv.small and Avg.Bwd.Segment.Size is vv.small and

4 Bwd.IAT.Max is vv.small and Avg.Bwd.Segment.Size is vv.large and

5 Bwd.IAT.Max is vv.small and Avg.Bwd.Segment.Size is vv.small and

6 Bwd.IAT.Max is vv.small and Avg.Bwd.Segment.Size is vv.small and

V150 V151 V152 V153 V154 V155 V156 V157

1 Max.Packet.Length is vv.small and Avg.Fwd.Segment.Size is vv.small THEN

2 Max.Packet.Length is vv.small and Avg.Fwd.Segment.Size is vv.small THEN

3 Max.Packet.Length is vv.small and Avg.Fwd.Segment.Size is vv.small THEN

4 Max.Packet.Length is vv.large and Avg.Fwd.Segment.Size is vv.small THEN

5 Max.Packet.Length is vv.small and Avg.Fwd.Segment.Size is vv.small THEN

6 Max.Packet.Length is vv.small and Avg.Fwd.Segment.Size is vv.small THEN

V158 V159 V160

1 ClassAttack is vv.large

2 ClassAttack is vv.small

3 ClassAttack   is vv.large

4 ClassAttack   is vv.small

5 ClassAttack   is vv.large

6 ClassAttack   is vv.small

[reached 'max' / getOption("max.print") -- omitted 57 rows]

>